M

Maritime Inventory Routing Problems

MARIELLE CHRISTIANSEN[1], KJETIL FAGERHOLT[1,2]
[1] Section of Managerial Economics, Finance and Operations Research, Norwegian University of Science and Technology, Trondheim, Norway
[2] Norwegian Marine Technology Research Institute (MARINTEK), Trondheim, Norway

Article Outline

Abstract

Maritime transportation is a heavily utilized mode when large quantities of bulk products need to be transported over long distances. Often, inventories exist at the loading and/or unloading ports of the sailing legs. When the ship operator has the responsibility for both the transportation of the fleet and the inventories at the ports, the underlying planning problem is a maritime inventory routing problem. Here we introduce the reader to various applications within maritime inventory routing and present some examples of research contributions. First we consider and present a mathematical model for the basic problem where a single product is transported and denote this problem the *inventory ship routing problem*. There exist a lot of extensions and variants of the problem. These include, among others, problems with inventories at only one end, variable production/consumption rates, multiple products, use of spot charters and problems that combine inventory routing with other planning aspects. Maritime inventory routing problems are very complex and to the authors' knowledge there exist no commercial optimization-based systems for the shipping industry yet. However, it is probably just a question of time before they become available.

Introduction

In order to survive in a tough global market, many companies have been forced to change their focus from competition between companies to competition between supply chains. Supply chains of companies with foreign sources of raw materials or with overseas customers most often include maritime transportation. Supply chain management and optimization are active fields of research, and we can see applications in almost all industries. So far the focus of such applications has usually not been much on maritime transportation, so there is a great potential and need for research in the area.

A maritime inventory routing problem is defined here as a combined ship routing and scheduling problem and an inventory management problem. The basic *inventory ship routing problem* (ISRP) concerns the transportation of a single product. The product is produced and stored in inventories at given loading ports and is transported by sea to inventories at unloading or consumption ports. Inventory capacities are defined in all ports. Further, we assume that information about production and consumption rates is given in all ports. To transport the product between the given production and consumption ports, the planners control a heterogeneous fleet of ships. The planning problem is to find routes and schedules for the fleet that minimize the transportation costs without interrupting production or consumption at the storages. Depending on the segment the fleet is operating in, the typical planning period spans from 1 to 2 weeks up to several months.

Most ship scheduling problems studied in the literature are so-called *cargo routing problems* [1]. In cargo routing problems, each cargo is specified by a given loading and unloading port. The quantity of the cargo is given and normally there exist time windows for loading and/or unloading. When planning routes and schedules, the shipping company either seeks to minimize the transportation costs for carrying all contracted cargoes or in addition to maximize profit for optional spot cargoes that may be available. We refer to [4] for a survey on maritime cargo routing problems. The cargo routing problems deviate from the ISPRs in a number of ways. The number of calls at a given port during the planning horizon is not predetermined in the ISRP, neither is the quantity to be loaded or unloaded at each port call. There is also no predefined pickup and delivery pair in the ISRP. The combination of the inventory management and the ship routing and scheduling makes the ISRP a very complex problem.

The inventory routing problem has been focused on in the literature for a couple of decades. Dror and Ball [8] defined the problem as a distribution problem in which each customer maintains a local inventory of a product such as heating oil and consumes a certain amount of that product each day. Given a central supplier (depot), the objective is to minimize the annual delivery costs while attempting to ensure that no customer runs out of the product at any time. The asymmetry between each type of inventory (production and consumption) with only one central supply node (depot) will often be found in road-based inventory routing problems, and more seldom in maritime transportation (ISRP). In the road-based inventory routing problem, the amount unloaded at each customer is often small compared to the total capacity of the vehicle. This is also in contrast with the ISRP, where the ship is often fully loaded and unloaded.

The objective of this article is to introduce the reader to various real planning problems within maritime inventory routing. The purpose is not to give a comprehensive overview of such problems, but rather to present examples of applications and research in the area.

The rest of the article is organized as follows: The first section defines the basic inventory ship routing problem and the underlying mathematical model. Extensions of the basic ISRP are addressed next. Finally, concluding remarks and future research follow.

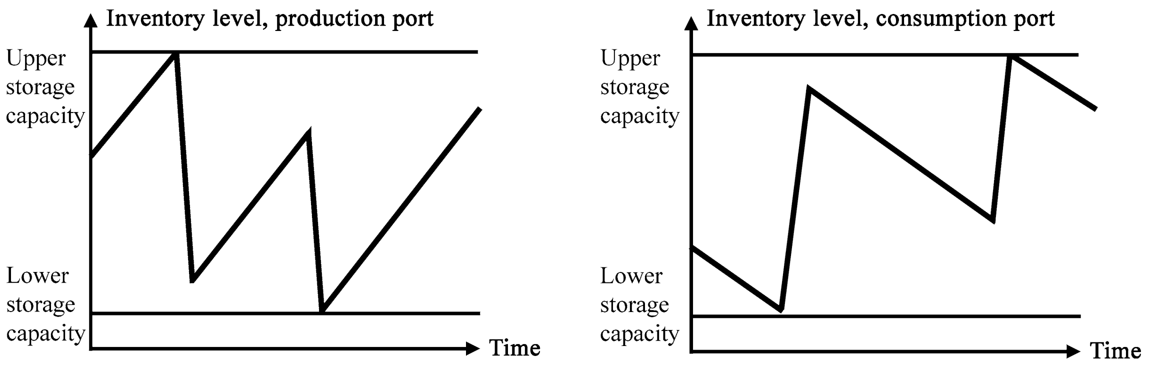
The Basic ISRP

In order to give an introduction to the various real planning problems within maritime inventory routing, we will start with a basic ISRP. First we describe the planning problem. Then we present an arc-flow formulation of the problem . The final section is devoted to real applications of the basic ISRP.

Problem Description

The products transported in maritime inventory routing problems are usually bulk products, where large quantities are transported and there are inventories at both the loading and the unloading ports. In these problems, the ship operators have a twofold responsibility: transportation and inventory management at the production and consumption sites. In such planning situations, the routing and scheduling of the fleet have to be synchronized with the inventory management at both production and consumption sites.

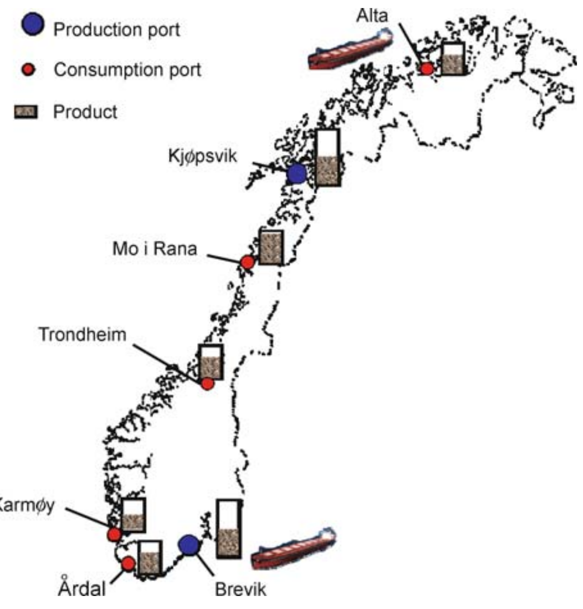
In the basic ISRP a single (homogeneous) product is transported. The product is produced at the sources, called loading ports, and consumed at the destinations, called unloading ports. Inventory storage capacities are given in all ports in addition to the production or consumption rate of the product. Here, the rate is assumed constant during the planning horizon.



Maritime Inventory Routing Problems, Figure 1
Inventory levels during a planning period for a production and a consumption port

The number of calls at a given port during the planning period is not predetermined, nor is the quantity to be loaded or unloaded at each port call. Figure 1 shows an example of a production/consumption and loading/unloading pattern. For both port types, the port is called at twice. However, the quantities loaded or unloaded differ at each call. The reason for this might be that the ports are visited by ships with different capacities loading/unloading full loads, or due to partial loading/unloading. In loading ports, it is important to ensure that the inventory level is not above the maximum inventory level when loading starts and not under the minimum inventory level when the loading has finished. In unloading ports, the opposite has to be ensured. The inventory level at the beginning of the planning period can be at any level, as indicated in Fig. 1.

Therefore, the planning problem is to design routes and schedules that minimize the transportation cost without interrupting production or consumption. We assume no inventory costs because the shipper owns both the producing sources and the consuming destinations. The ship operator controls a heterogeneous fleet of ships. We assume that partial loading and unloading is allowed, such that two ports of the same type (loading or unloading) may be called at in succession. The ship is not necessarily empty at the beginning of the planning horizon, but might have some load onboard. The ship can be either at a port or at sea at the beginning of the planning horizon. Figure 2 shows a simplified illustration of the planning problem for a cement producer in Norway with two production factories and five consumption ports with inventories. The fleet consists of two ships. Each port can be called at several times during the planning period by the same ship or different ships.



Maritime Inventory Routing Problems, Figure 2
A simplified planning situation with seven ports and two ships

Mathematical Model

The model of the ISRP will be presented in a compact and simplified way. In Sect. "Routing," we describe the flow network and the objective function. Then, the conditions for the loading and unloading, the time aspects and the inventories are described in Sect. "Loading and Unloading", "Scheduling" and "Inventory Management," respectively. We base our notation and model on those of Christiansen et al. [3].

In the upcoming formulation, we have assumed that the ship may be partially loaded/unloaded, meaning that multiple cargoes may be onboard a ship simultaneously. The model could have been simplified if we had assumed full loads only or sailing between different port types (from loading to unloading and vice versa).

Routing In the mathematical description of the network each port is represented by an index i and the set of ports is given by $\mathcal{N}$. Let $\mathcal{V}$, indexed by v, be the set of available ships to be routed and scheduled. Not all ships can visit all ports, and $\mathcal{N}_v$ = {feasible ports for ship v} $\cup\{o(v),d(v)\}$ is the set of ports that can be visited by ship v. The terms $o(v)$ and $d(v)$ represent the artificial origin port and artificial destination port of ship v, respectively. Each port can be visited several times during the planning horizon, and $\mathcal{M}_i$ is the set of possible calls at port i, while $\mathcal{M}_{iv}$ is the set of calls at port i that can be made by ship v. The port call number is represented by an index m, and M_i is the last possible call at port i within the planning period. The set of nodes in the flow network represents the set of port calls, and each port call is specified by (i, m), $i \in \mathcal{N}, m \in \mathcal{M}_i$. In addition, we specify flow networks for each ship v with nodes (i, m), $i \in \mathcal{N}_v, m \in \mathcal{M}_{iv}$. $\mathcal{A}_v$ contains all feasible arcs for ship v, which is a subset of $\{i \in \mathcal{N}_v, m \in \mathcal{M}_{iv}\} \times \{i \in \mathcal{N}_v, m \in \mathcal{M}_{iv}\}$. Finally, C_{ijv} represents the variable costs for sailing between port i and port j with ship v. This includes port, channel and fuel costs.

In the network flow part of the formulation we use the following types of variables: the binary flow variable x_{imjnv}, $v \in \mathcal{V}$, $(i, m, j, n) \in \mathcal{A}_v$ equals 1, if ship v sails from node (i, m) directly to node (j, n), and 0 otherwise, and the slack variable w_{im}, $i \in \mathcal{N}$, $m \in \mathcal{M}_i$ is equal to 1 if no ship takes port call (i, m), and 0 otherwise. The routing formulation including the objective function is as follows:

$$\min \sum_{v\in\mathcal{V}} \sum_{(i,m,j,n)\in\mathcal{A}_v} C_{ijv} x_{imjnv}, \tag{1}$$

subject to

$$\sum_{v\in\mathcal{V}} \sum_{j\in\mathcal{N}_v} \sum_{n\in\mathcal{M}_{jv}} x_{imjnv} + w_{im} = 1, \; \forall i \in \mathcal{N}, m \in \mathcal{M}_i, \tag{2}$$

$$\sum_{j\in\mathcal{N}_v} \sum_{n\in\mathcal{M}_{jv}} x_{o(v)1jnv} = 1, \; \forall v \in \mathcal{V}, \tag{3}$$

$$\sum_{i\in\mathcal{N}_v} \sum_{m\in\mathcal{M}_{iv}} x_{imjnv} - \sum_{i\in\mathcal{N}_v} \sum_{m\in\mathcal{M}_{iv}} x_{jnimv} = 0, \quad \forall v \in \mathcal{V}, j \in \mathcal{N}_v \backslash \{o(v), d(v)\}, n \in \mathcal{M}_{jv}, \tag{4}$$

$$\sum_{i\in\mathcal{N}_v} \sum_{m\in\mathcal{M}_{iv}} x_{imd(v)1v} = 1, \; \forall v \in \mathcal{V}, \tag{5}$$

$$w_{im} - w_{i(m-1)} \geq 0, \; \forall i \in \mathcal{N}, m \in \mathcal{M}_i, \tag{6}$$

$$x_{imjnv} \in \{0,1\}, \; \forall v \in \mathcal{V}, (i,m,j,n) \in \mathcal{A}_v, \tag{7}$$

$$w_{im} \in \{0,1\}, \; \forall i \in \mathcal{N}, m \in \mathcal{M}_i. \tag{8}$$

The objective function (1) minimizes the total costs. Constraints (2) ensure that each port call is visited at most once. Constraints (3)–(5) describe the flow on the sailing route used by ship v. One or several of the calls in a specified port can be made by a dummy ship, and the highest call numbers will be assigned to dummy ships in constraints (6). These constraints reduce the number of symmetrical solutions in the solution approach. For the calls made by a dummy ship, we get artificial starting times and artificial inventory levels within the defined upper and lower limits. Finally, the formulation involves binary requirements (7) and (8) on the flow variables and port call slack variables, respectively.

Loading and Unloading The capacity of ship v is given by V_{CAPv}. Variable l_{imv}, $v \in V$, $i \in \mathcal{N}_v \backslash \{d(v)\}$, $m \in \mathcal{M}_{iv}$ gives the total load onboard ship v just after the service is completed at node (i, m), while variable q_{imv}, $v \in V$, $i \in \mathcal{N}_v \backslash \{d(v)\}, m \in \mathcal{M}_{iv}$ represents the quantity loaded or unloaded at port call (i, m), when ship v visits (i, m). It is assumed that nothing is loaded or unloaded at the artificial origin $o(v)$; $q_{o(v)1v} = 0$. However, the ships may have cargo onboard, L_{0v}, at the beginning of the planning horizon; $l_{o(v)1v} = L_{0v}$. Further, constant I_i is equal to 1 if port i is a loading port, -1 if port i is an unloading port and 0 if port i is $o(v)$ or $d(v)$. Constraints related to the quantity onboard a ship can be formulated as follows:

$$x_{imjnv}(l_{imv} + I_j q_{jnv} - l_{jnv}) = 0, \quad \forall v \in \mathcal{V}, (i,m,j,n) \in \mathcal{A}_v | j \neq d(v), \tag{9}$$

$$q_{imv} \leq l_{imv} \leq \sum_{j \in \mathcal{N}_v} \sum_{n \in \mathcal{M}_{jv}} V_{CAPv} x_{imjnv}, \quad \forall v \in \mathcal{V}, i \in \mathcal{N}_v, m \in \mathcal{M}_{iv} | I_i = 1, \tag{10}$$

$$0 \leq l_{imv} \leq \sum_{j \in \mathcal{N}_v} \sum_{n \in \mathcal{M}_{jv}} V_{CAPv} x_{imjnv} - q_{imv}, \quad \forall v \in \mathcal{V}, i \in \mathcal{N}_v, m \in \mathcal{M}_{iv} | I_i = -1. \tag{11}$$

Constraints (9) give the relationship between the binary flow variables and the ship load at each port call. Constraints (10) and (11) give the ship capacity intervals at the port calls for loading and unloading ports, respectively.

Scheduling The time required to load or unload the ship may constitute a major part of the total time in many maritime transportation applications. It is therefore usual to calculate this as a function of the quantity loaded/unloaded. The time spent loading/unloading one unit of a cargo at port i is given by T_{Qi}. The term T_{Sijv} represents the sailing time from port i to port j with ship v. In some ports, there is a minimum required time, T_{Bi}, between the departure of one ship and the arrival of the next ship, due to small port area or narrow channels from the port to the pilot station. The time variable t_{im}, $(i \in \mathcal{N}, m \in \mathcal{M}_i) \cup (i \in o(v), \forall v, m = 1)$ represents the time at which service begins at node (i, m). It is assumed that the ship arrives at $o(v)$ at a given fixed time; $t_{o(v)1} = T_{0v}$. Finally, let T denote the planning horizon. The scheduling constraints can now be written as follows:

$$x_{imjnv}(t_{im} + T_{Qi} q_{imv} + T_{Sijv} - t_{jn}) \leq 0, \quad \forall v \in \mathcal{V}, (i,m,j,n) \in \mathcal{A}_v | j \neq d(v), \tag{12}$$

$$t_{im} - t_{i(m-1)} - \sum_{v \in \mathcal{V}} T_{Qi} q_{i(m-1)v} + T_{Bi} w_{im} \geq T_{Bi}, \quad \forall i \in \mathcal{N}, m \in \mathcal{M}_i \backslash \{1\}. \tag{13}$$

Constraints (12) take into account the timing or scheduling on the route. Note that waiting at a port is allowed. Constraints (13) prevent service overlap in the ports and ensure the order of real calls at the same port. A ship must complete its service before the next ship starts its service at the same port. If port i does not have constraints regarding the minimum time between departure of one ship and arrival of the next, $T_{Bi} = 0$. If port i also allows the service of several ships simultaneously, constraints (13) will simply be $t_{im} - t_{i(m-1)} \geq 0$, to ensure the order of calls at the port.

Inventory Management The levels of the inventory have to be within a given interval at each port $[S_{MNi}, S_{MXi}]$. The production rate R_i is positive if port i is producing the product, and negative if port i is consuming the product. At the beginning of the planning horizon, the inventory level at each port i is S_{0i} . Finally, s_{im}, $i \in \mathcal{N}, m \in \mathcal{M}_i$ represents the inventory level when service starts at port call (i, m). The inventory constraints of the formulation become

$$s_{i1} - R_i t_{i1} = S_{0i}, \; \forall i \in \mathcal{N}, \tag{14}$$

$$s_{i(m-1)} - \sum_{v \in \mathcal{V}} I_i q_{i(m-1)v} + R_i(t_{im} - t_{i(m-1)}) - s_{im} = 0, \quad \forall i \in \mathcal{N}, m \in \mathcal{M}_i \backslash \{1\}, \tag{15}$$

$$S_{MNi} \leq s_{im} \leq S_{MXi}, \; \forall i \in \mathcal{N}, m \in \mathcal{M}_i, \tag{16}$$

$$S_{MNi} \leq s_{im} - \sum_{v \in \mathcal{V}} I_i q_{imv} + R_i(T - t_{im}) \leq S_{MXi}, \quad \forall i \in \mathcal{N}, m = M_i. \tag{17}$$

The inventory level at the first call at each port is calculated in constraints (14). From constraints (15), we find the inventory level at any port call (i, m) from the inventory level upon arrival at the port in the previous call (i,m-1), adjusted for the loaded/unloaded quantity at the port call and the production/consumption between the two arrivals. The general inventory limit constraints at each port call are given in (16). Constraints (17) ensure that the level of inventory at the end of the planning horizon is within its limits. It can easily be shown by substitution that constraints (17) ensure that the inventory at time T will be within the bounds even if ports are not visited at their last calls.

A Real Application An application that is close to the ISRP is a real ship planning problem for ammonia transportation. Norsk Hydro Agri (now named Yara) produces and consumes ammonia in its factories worldwide. The planners at the company are responsible for keeping the inventory levels within the predefined upper and lower limits at all Norsk Hydro Agri factories around the world where they produce and consume ammonia. This requires the planners to design routes and schedules for their fleet of heterogeneous ships transporting ammonia from production ports to consumption ports. The problem is described in detail in Christiansen [2]. The overall solution approach is based on a column generation approach with columns for both the ship routes and the inventory management sequences [5], where subproblems are solved by dynamic programming for each port and each ship [6]. Another solution approach to the same problem was developed by Flatberg et al. [9]. They used an iterative improvement heuristic combined with an linear program (LP) solver to solve this problem. The heuristic is used to solve the combinatorial problem of finding the ship routes, and an LP model is used to find the starting time of service at each call and the loading/unloading quantities.

Extensions of the ISRP

Most of the real applications of maritime inventory routing problems have a more complex structure than the basic ISRP. We present here various extensions of the ISRP that are described in the literature or have been experienced in our research group. In many maritime applications, several of the extensions are combined.

One Central Supplier or Consumer

As mentioned in the introduction, the road-based inventory routing problem often has a vehicle routing problem (VRP) structure, where a central supplier (or depot) serves a set of customers with a local inventory and a consumption rate. We can imagine a lot of real planning problems with such a structure, for instance, in the gasoline business, delivering gasoline to gas stations from a refinery or central storage. Milk collection at farms for transport to a dairy has the opposite structure, where the customers are producers and the depot consumes the milk.

In the maritime sector, we can also find this VRP structure for ship operators dealing with maritime inventory routing problems. The Norwegian oil company Statoil will start its production of natural gas from Snøhvit, Melkøya, north of Norway in 2008. Most of the gas will be cooled down and transported as liquefied natural gas (LNG) by LNG tankers. At the moment the planning problem concerns one source producing the gas and several consumption ports. Frich and Horgen [10] presented a mixed integer program (MIP) model of the planning problem where this special VRP structure is exploited.

Similar maritime inventory routing problems can be found, for instance, with the Arabian Gulf as the source for the transportation of both LNG and heavier oil products.

Inventory Constraints in Either Production Ports or Consumption Ports

For the ISRP, the inventory management is considered at both the loading and the unloading ports. However, many real planning problems concern the design of routes and schedules for a fleet of ships with inventory constraints at just one of the port types. There exist for instance ship operators engaged in vendor managed inventory (VMI) contracts. Here, the ship operator monitors its customers' inventories and must ensure that these are kept within predefined limits. Often, the customers are concerned about inventories at only the unloading ports, while the ship operator has entered into a contract to supply the product with given quantities and time windows from the loading ports. The opposite might also be the case, where the customers have inventories at only loading ports. Then, the ship operator must also engage in contracts to deliver these volumes with given quantities and time windows.

Variable Production or Consumption Rate

The production and consumption rates are assumed constant for all port inventories during the planning period in the ISRP. However, for many real planning problems this assumption is too coarse, and the production and consumption that may vary from day to day have to be taken into account in the modeling. Including this aspect into the basic ISRP model would result in a more complicated model and it would become harder to solve.

A maritime inventory routing problem for the LNG business was considered by Grønhaug et al. [11]. Here the production of LNG at the liquefaction plants and the consumption of LNG at the regasification terminals have to be regarded as variable. In order to overcome these complicating factors, a time discretized model was developed, and it was solved by a column generation approach.

Also Ronen [16] used a time discretized model with a variable production and consumption rate for an inventory routing problem for refinery products. The model focuses on the inventory and not the routing part of the problem, as the model solution suggests shipment sizes that are assumed to be an input for a cargo routing problem at a later stage.

Multiple Products

Here we extend the ISRP to the multiproduct case. In the ISRP several cargoes may be transported simultaneously in one ship, but the product is assumed to be the same. This means that the product does not need to be transported in separated compartments onboard the ship or stored in separate stores at the ports.

The problem with multiple products is frequently encountered by chemical and oil product transport companies. Al-Khayyal and Hwang [1] gave a mathematical formulation for such a problem where the products are assumed to require dedicated compartments in the ship. For this problem there exist inventory limits and production/consumption rates for each product in each port. Hwang [13] used a combined Lagrangian relaxation and heuristic approach to solve test instances of the problem.

The problem described in Ronen [16] also includes multiple products. Sometimes the stowage onboard the ship must also be considered in the inventory routing problem; see, for instance, Haugen and Lund [12] for the transportation of cement products.

Use of Spot Charters

In some cases the dedicated fleet of ships has insufficient transportation capacity to provide continuous production at all sources and consumption at all destinations. In such a case some of the loads can be serviced by spot charters, which are ships chartered for a single voyage.

The cement company described by Haugen and Lund [12] is faced with limited vessel capacity. In some periods the company makes use of spot charters, while in peak periods additional road-based transportation is necessary. In their solution approach, the consumption inventories are sorted according to their importance and their location regarding what the cost effects for additional trucks will be. It is ensured that the inventories with highest priority are served by the fleet of ships.

Combined Inventory Routing and Cargo Routing

The cargo routing problem was introduced in the introduction. For this problem, there exist predefined cargoes with specified quantities and time windows. The cargoes may be contracted or optional spot ones. Often the companies facing an ISRP trade cargoes with other operators in order to better utilize the fleet and to ensure there is product balance at their own plants.

In the real problem described by Christiansen [2], the shipper trades ammonia with other operators. These traded volumes are determined by negotiations. The ship operator undertakes to load or unload ammonia within a determined quantity interval and to arrive at a particular port within a given time window. For these external ports, no inventory management problem exists.

There also exist shipping companies that have VMI contracts with some customers, but apart from that are involved in ordinary cargo routing. This will give these shipping companies a combined inventory and cargo routing planning problem.

Combining Inventory Routing with Other Planning Aspects

The ISRP concerns parts of a supply chain and focuses on sea transportation and the inventories at both ends of the sailing leg. In many real planning situations, it is sensible to consider larger parts of the supply chain. Persson and Göthe-Lundgren [15] studied a planning problem that integrates both the shipment planning of petroleum products from refineries to depots and the production scheduling at the refineries. Shih [17] and Liu and Sherali [14] presented two other maritime supply chain applications where coal is transported.

Rather than considering a larger part of the supply chain, the ISRP may be combined with other planning aspects. In Sect. "Multiple Products," we referred to the combined ISRP and stowage of different cement products in various compartments onboard the ships. See Haugen and Lund [12] for more information about the case and solution approach.

Concluding Remarks

We have described the maritime inventory routing problem, which is a combined inventory management and a ship routing and scheduling problem. The so-called basic ISRP and several extensions to the ISRP were presented. In practice, planners are more often faced with extensions of the ISRP and also the extensions described in combination with each other.

As far as we know, no generic commercial optimization-based decision support system exists for solving maritime inventory routing problems. However, the shipping industry is experiencing an increased need for such systems owing to extended planning responsibility and increased fleet sizes. We expect that such systems will be available on the market in the years to come.

The basic VRP is computationally very hard. The maritime inventory routing problem is even more demanding owing to the additional degrees of freedom. Many of the extensions discussed in this article are barely touched on in the operations research community. This means that there exist a lot of research challenges, in the development of both exact methods and heuristic solution methods.

Maritime transportation is faced with higher uncertainty in its operations compared with many other modes of transportation. This is due to greater dependence on weather conditions and technology. For the maritime inventory routing problem, we have also uncertainties in the production and consumption at the inventories. The consideration of these uncertainty aspects is another interesting topic of research.

Acknowledgements

This work was carried out with financial support from the Research Council of Norway through the INSUMAR project (Integrated supply chain and maritime transportation planning), the OPTIMAR project (Optimization in maritime transportation and logistics) and the DOMinant project (Discrete optimization methods in maritime and road-based transportation).

References

1. Al-Khayyal F, Hwang S-J (2007) Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk, Part I: Applications and model. Eur J Oper Res 176:106–130
2. Christiansen M (1999) Decomposition of a combined inventory and time constrained ship routing problem. Transp Sci 33(1):3–16
3. Christiansen M, Fagerholt K, Nygreen B, Ronen D (2007) Maritime Transportation. In: Barnhart C, Laporte G (eds) Handbooks in Operations Research and Management Science: 14 Transportation. North-Holland, Amsterdam, pp 189–284
4. Christiansen M, Fagerholt K, Ronen D (2004) Ship routing and scheduling: Status and perspectives. Transp Sci 38(1):1–18
5. Christiansen M, Nygreen B (1998) A method for solving ship routing problems with inventory constraints. Ann Oper Res 81:357–378
6. Christiansen M, Nygreen B (1998) Modeling path flows for a combined ship routing and inventory management problem. Ann Oper Res 82:391–412
7. Christiansen M, Nygreen B (2005) Robust inventory ship routing by column generation. In: Desaulniers G, Desrosiers J, Solomon MM (eds) Chapter 7, Column generation. Springer, pp 197–224
8. Dror M, Ball M (1987) Inventory/Routing: Reduction from an Annual to a Short-Period Problem. Nav Res Logist 34: 891–905
9. Flatberg T, Haavardtun H, Kloster O, Løkketangen A (2000) Combining exact and heuristic methods for solving a vessel routing problem with inventory constraints and time windows. Ric Oper 29(91):55–68
10. Frich OT, Horgen R (2004) Supply chain optimization: Snøhvit LNG distribution. Master thesis, Norwegian University of Science and Technology, Trondheim, pp 144
11. Grønhaug R, Christiansen M, Desaulniers G, Desrosiers J (2008) A branch-and-price-and-cut method for a liquefied natural gas inventory routing problem. Working paper, Norwegian University of Science and Technology, Trondheim
12. Haugen Ø, Lund EH (2006) Optimization-based decision support for planning of cement distribution in maritime supply chains. Master thesis, Norwegian University of Science and Technology, Trondheim, pp 137
13. Hwang S-J (2005) Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk. Ph.d. Thesis, Georgia Institute of Technology, Atlanta
14. Liu C-M, Sherali HD (2000) A coal shipping and blending problem for an electric utility company. OMEGA 28:433–444
15. Persson JA, Göthe-Lundgren M (2005) Shipment planning at oil refineries using column generation and valid inequalities. Eur J Oper Res 163:631–652

16. Ronen D (2002) Marine inventory routing: Shipments planning. J Oper Res Soc 53:108–114
17. Shih L-H (1997) Planning of fuel coal imports using a mixed integer programming method. Int J Prod Econ 51:243–249

Mathematical Programming for Data Mining

Ioannis P. Androulakis[1],
W. Art Chaovalitwongse[2]
[1] Department of Biomedical Engineering, Rutgers University, Piscataway, USA
[2] Department of Industrial and Systems Engineering, Rutgers University, Piscataway, USA

Article Outline

Keywords

Mathematical programming; Data mining; Optimization; Clustering; Classification

Introduction

Progress in digital data acquisition and storage technology has resulted in the growth of huge databases. This has occurred in a variety of scientific and engineering research applications [8] as well as medical domain [19,20]. Making sense out of these rapidly growing massive data sets gave birth to a "new" scientific discipline often referred to as *Data Mining*. Defining a discipline is, however, always a controversial task. The following working definition of the area was recently proposed [9]: Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner.

Clearly the term data mining if often used as a synonym for the process of extracting useful information from databases. However, the overall knowledge discovery from databases (KDD) process is far more complicated and convoluted and involves a number of additional pre and post-processing steps [6]. Therefore, in our definition data mining refers to the ensemble of new, and existing, specific algorithms for extracting structure from data [8]. The exact definition of the knowledge extraction process and the expected outcomes are very difficult to characterize. However, a number of specific tasks can be identified and, by and large, define the key subset of deliverables from a data mining activity. Two such critical activities are classification and clustering.

A number of variants for these tasks can be identified and, furthermore, the specific structure of the data involved greatly impacts the methods and algorithms that are to be employed. Before we proceed with the exact definition of the tasks we need to provide working definitions of the nature and structure of the data.

Basic Definitions

For the purposes of our analysis we will assume that the data are expressed in the form of n-dimensional feature vectors $x \in X \subseteq \Re^n$. Appropriate pre-processing of the data may be required to transform the data into this form. Although in many cases this transformations can be trivial, in other cases transforming the data into a "workable" form is a highly non-trivial task. The goal of data mining is to estimate an explicit, or implicit, function that maps points of the feature vector from the input space, $X \subseteq \Re^n$, to an output space, C, given a finite sample. The concept of the finite sample is important because, in general, what we are given is a finite representative subset of the original space (training set) and we wish to make predictions on new elements of the set (testing set). The data mining tasks can thus de defined based on the nature of the mapping C and the extent to which the train set is characterized.

If the predicted quantity is a categorical value and if we know the value that corresponds to each elements of the training set then the question becomes how to identify the mapping that connects the feature vector and the corresponding categorical value (class). This problem is known as the classification problem (supervised learning). If the class assignment is not known and we seek to: (*a*) identify whether a small, yet unknown, number of classes exist; (*b*) define the mapping assigning the features to classes then we have a clustering problem (unsupervised learning).

A related problem associated with superfluous information in the feature vector is the so-called feature selection problem. This is a problem closely related to over-fitting in regression. Having a minimal number of features leads to simpler models, better generalization and easier interpretation. One of the fundamental issues in data mining is therefore to identify the least number of features, sub-set of the original set of features, that best address the two issues previously defined. The concept of parsimony (Occam's razor) is often invoked to bias the search [1]: never do with more what can be done with fewer.

Although numerous methods exist for addressing these problems they will not be reviewed here. Nice reviews of classification and were recently presented in [8,9]. In this short introduction we will concentrate on solution methodologies based on reformulating the clustering, and classification questions as optimization problems.

Mathematical Programming Formulations

Classification and clustering, and for that matter most of the data mining tasks, are fundamentally optimization problems. Mathematical programming methodologies formalize the problem definition and make use of recent advances in optimization theory and applications for the efficient solution of the corresponding formulations. In fact, mathematical programming approaches, particularly linear programming, have long been used in data mining tasks.

The pioneering work presented in [13,14] demonstrated how to formulate the problem of constructing planes to separate linearly separable sets of points.

In this summary we will follow the formalism put forth in [2] since it presented one of the most comprehensive approaches to this problem. One of the major advantages of a formulation based on mathematical programming is the ease in incorporating explicit problem specific constraints. This will be discussed in greater detail later in this summary.

Classification

As discussed earlier the main goal in classification is to predict a categorical variable (class) based on the values of the feature vector. The general families of methods for addressing this problem include [9]:

i) Estimation of the conditional probability of observing class C given the feature vector x.

ii) Analysis of various proximity metrics and based the decision of class assignment based on proximity.

iii) Recursive input space partitioning to maximize a score of class purity (tree-based methods).

The two-class classification problem can be formulated as the search of a function that assigns a given input vector x into two disjoint point sets A and B. The data are represented in the form of matrices. Assuming that the set A has m elements and the set B has k elements, then $A \in \Re^{m \times n}, B \in \Re^{k \times n}$, describe the two sets respectively. The discrimination in based on the derivation of hyperplane

$$P = \{x | x \in \Re^n, x^T \omega = \gamma\}$$

with normal and distance from the origin $\frac{|\gamma|}{||\omega||_2}$. The optimization problem then becomes to determine ω and γ such that the separating hyperplane P defines two open half spaces

$$\{x | x \in \Re^n, x^T \omega < \gamma\}$$
$$\{x | x \in \Re^n, x^T \omega > \gamma\}$$

containing mostly points in A and B respectively. Unless A and B are disjoint the separation can only be satisfied within some error. Minimization of the average violations provides a possible approximation of the separating hyperplane [2]:

$$\min_{\omega,\gamma} \frac{1}{m} \|(-A\omega + e\gamma + e)_+\|_1 + \frac{1}{k} \|(-B\omega + e\gamma + e)_+\|_1$$

In [2] a number of linear programming reformulations are discussed exploring the properties of the structure of the optimization problem. In particular an effective robust linear programming (RLP) reformulation was suggested making possible the solution of large-scale problems:

$$\begin{aligned}\min_{\omega,\gamma,y,z} \ & \frac{e^T y}{m} + \frac{e^T z}{k} \\ \text{s.t.} & -A\omega + e\gamma + e \leq y \\ & B\omega - e\gamma + e \leq z \\ & y, z \geq 0.\end{aligned}$$

In [17] it was demonstrated how the above formulation can be applied repeatedly to produce complex space partitions similar to those obtained by the application of standard decision tree methods such as C4.5 [21] or CART [4].

Clustering

The goal of clustering is the segmentation of the raw data into groups that share a common, yet unknown, characteristic property. Similarity is therefore a key property in any clustering task. The difficulty arises from the fact that the process is unsupervised. That is neither the property nor the expected number of groups (clusters) are known ahead of time. The search for the optimal number of clusters is parametric in nature and the optimal point in an "error" vs. "number of clusters" curve is usually identified by a combined objective the weighs appropriately accuracy and number of clusters. Conceptually a number of approaches can be developed for addressing clustering problems:

- **i)** Distance-based methods, by far the most commonly used, that attempt to identify the best k-way partition of the data by minimizing the distance of the points assigned to cluster k from the center of the cluster.
- **ii)** Model-based methods assume the functional form of a model that describes each of the clusters and then search for the best parameter fit that models each cluster by minimizing some appropriate likelihood measure.

There are two different types of clustering: (1) hard clustering; (2) fuzzy clustering. The former assigns a data point to *exactly* one cluster while the latter assigns a data point to one of more clusters along with the likelihood of the data point belonging to one of those clusters.

The standard formulation of the hard clustering problem is:

$$\min_{c} \sum_{i=1}^{m} \min_{l} \|x^i - c^l\|_n$$

That is given m points, x, in an n-dimensional space, and a fixed number of cluster, k, determine the centers of the cluster, c, such that the sum of the distances of each point to a nearest cluster center is minimized. It was shown in [3] that this general non convex problem can be reformulated such that we minimize a bilinear functions over a polyhedral set by introducing a selection variable t_{il}:

$$\begin{aligned}\min_{c,d,t} \ & \sum_{i=1}^{m}\sum_{i=1}^{k} t_{il}(e^T d_{il}) \\ \text{s.t.} & -d_{il} \leq x^i - c^l \leq d_{il} \\ & \sum_{l=1}^{k} t_{il} = 1 \\ & t_{il} \geq 0 \\ & i = 1, \dots, m, l = 1, \dots, k.\end{aligned}$$

d is a dummy variable used to bound the components of the difference $x - c$. In the above formulation the 1-norm is selected [3].

The fuzzy clustering problem can be formulated as follows [5]:

$$\begin{aligned}\min_{w} \ & \sum_{i=1}^{m}\sum_{l=1}^{k} w_{il}^2 \|x^i - c^l\|^2 \\ \text{s.t.} & \sum_{l=1}^{k} w_{il} = 1 \\ & w_{il} \geq 1,\end{aligned}$$

where x^i, $i = 1, \dots, m$ is the location descriptor for the data point, c^l, $l = 1, \dots, k$ is the center of the cluster, w_{il} is the likelihood of a data point i being assigned to cluster l.

Support Vector Machines

This optimization formalism bares significance resemblance to the Support Vector Machines (SVM) framework [25]. SVM incorporate the concept of structural risk minimization by determining a separating hyperplane that maximizes not only a quantity measuring the misclassification error but also maximizing the margin separating the two classes. This can be achieved by augmenting the objective of the RLP formulation earlier presented by an appropriately weighted measure of the separation between the two classes as $(1-\lambda)(e^T y + e^T z) + \frac{\lambda}{2}\|\omega\|_2^2$.

In [6] the concept of SVM is extended by introducing the Proximal support vector machines which classify points based on proximity to one of two parallel planes that are pushed as far apart as possible. Nonlinear transformations were also introduced in [6] to enable the derivation of non-linear boundaries in classifiers.

Multi-Class Support Vector Machines

Support vector machines were originally designed for binary classification. Extending to multi-class problems is still an open research area [10].

The earliest multi-class implementation is the *one against all* [22] by constructing k SVM models, where k is the number of classes. The ith SVM is classifies the examples of class i against all the other samples in all other classes. Another alternative builds *one against one* [12] classifiers by building $\frac{k(k-1)}{2}$ models where each is trained on data from two classes. The emphasis of current research is on novel methods for generating all the decision functions through the solution of a single, but much larger, optimization problem [10].

Data Mining in the Presence of Constraints

Prior knowledge about a system is often omitted in data mining applications because most algorithms do not have adequate provisions for incorporating explicitly such types of constrains. Prior knowledge can either encodes explicit and/or implicit relations among the features or models the existence of "obstacles" in the feature space [24].

One of the major advantages of a mathematical programming framework for performing data mining tasks is that prior knowledge can be incorporated in the definition of the various tasks in the form of (non)linear constraints. Efficient incorporation of prior knowledge in the form of nonlinear inequalities within the SVM framework was recently proposed by [15]. Reformulations of the original linear and nonlinear SVM classifiers to accommodate prior knowledge about the problem were presented in [7] in the context of approximation and in [16] in the context of classifiers.

Data Mining and Integer Optimization

Data mining tasks involve, fundamentally, discrete decisions:

- How many clusters are there?
- Which class does a record belong to?
- Which features are most informative?
- Which samples capture the essential information?

Implicit enumeration techniques such as branch-and-bound were used early on to address the problem of feature selection [18].

Mathematical programming inspired by algorithms for addressing various data mining problems are now being revisited and cast as integer optimization problems. Representative formulations include feature selection using Mixed-Integer Linear Programs [11] and in [23] integer optimization models are used to address the problem of classification and regression.

Research Challenges

Numerous issues can of course be raised. However, we would like to focus on three critical aspects

i) Scalability and the curse of dimensionality. Databases are growing extremely fast and problems of practical interest are routinely composed of millions of records and thousands of features. The computational complexity is therefore expected to grow beyond what is currently reasonable and tractable. Hardware advances alone will not address this problem either as the increase in computational complexity outgrows the increase in computational speed. The challenge is therefore two-fold: either improve the algorithms and the implementation of the algorithms or explore sampling and dimensionality reduction techniques.

ii) Noise and infrequent events. Noise and uncertainty in the data is a given. Therefore, data mining algorithms in general and mathematical programming formulations in particular have to account for the presence of noise. Issues from robustness and uncertainty propagation have to be incorporated.

However, an interesting issue emerges: how do we distinguish between noise and an infrequent, albeit interesting observation? This in fact maybe a question with no answer.

iii) Interpretation and visualization. The ultimate goal of data mining is understanding the data and developing actionable strategies based on the conclusions. We need to improve not only the interpretation of the derived models but also the knowledge delivery methods based on the derived models. Optimization and mathematical programming needs to provide not just the optimal solution but also some way of interpreting the implications of a particular solution including the quantification of potential crucial sensitivities.

References

1. Blumer A, Ehrenfeucht A, Haussler D, Warmuth MK (1987) Occam's razor. Inf Process Lett 24:377–380
2. Bradley PS, Fayyad U, Mangasarian OL (1999) Mathematical programming for data mining: Formulations and challenges. INFORMS J Comput 11:217–238
3. Bradley PS, Mangasarian OL, Street WN (1997) Clustering via concave minimization. In: Mozer MC, Jordan MI, Petsche T (eds) Advances in Neural Information Processing Systems. MIT Press, Cambridge, pp 368–374
4. Breiman L, Friedman J, Olsen R, Stone C (1993) Classification and Regression Trees. Wadsworth Inc., Boca Raton
5. Dunn JC (1973) A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. J Cybern 3:32–57
6. Fayyad U, Piatetsky-Shapiro G, Smyth P (1996) From data mining to knowledge discovery: An overview. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthursamy R (eds) Advances in knowledge discovery and data mining. AAAI Press, Cambridge, pp 229–248
7. Fung GM, Mangasarian OL (2001) Proximal Support Vector Machine Classifiers In: Provost F, Srikant R (eds) Proceedings KDD-2001: Knowledge Discovery and Data Mining, San Francisco, August 26–29 2001, Asscociation for Computing Machinery, pp 77–86, ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-02.ps. Accessed 2004
8. Grossman RL, Kamath C, Kegelmeyer P, Kumar V, Namburu EE (2001) Data mining for scientific and engineering applications. Kluwer, Dordrecht
9. Hand DJ, Mannila H, Smyth P (2001) Principle of data mining. Bradford Books, Cambridge
10. Hsu C-W, Lin C-J (2002) A comparison of methods multiclass support vector machines. IEEE Trans Neural Netw 13:415–425
11. Iannatilli FJ, Rubin PA (2003) Feature selection for multiclass discrimination via mized-integer linear programming. IEEE Trans Pattern Anal Mach Learn 25:779–783
12. Krebel U (1999) Pairwise classification and support vector machines: Advances in Kernel Methods – Support Vector Learning. MIT Press, Cambridge
13. Mangasarian OL (1965) Linear and nonlinear separation of pattern by linear programming. Oper Res 31:445–453
14. Mangasarian OL (1968) Multisurface method for pattern separation. IEEE Trans Inf Theory IT-14:801–807
15. Mangasarian OL, Shavlik JW, Wild EW (2003) Knowledge-based kernel approximations. Tech. rep., Data Mining Institute, University of Wisconsin, Madison
16. Mangasarian OL, Shavlik JW, Wild EW (2004) Knowledge-based kernel approximation. J Mach Learn Res 5:1127–1141
17. Mangasarian OL, Street WN, Wolberg WH (1995) Breast cancer diagnosis and prognosis via Linear Programming. Oper Res 43:570–577
18. Narendra P, Fukunaga K (1977) A branch and bound algorithm for feature subset selection. IEEE Trans Comput 26:917–922
19. Pardalos PM, Boginski V, Vazakopoulos A (2007) Data Mining in Biomedicine. Springer, Berlin
20. Pardalos PM, Principe J (2002) Biocomputing, Kluwer, Dordrecht
21. Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco
22. Scholkopf B, Burges C, Vapnik V (1995) Extracting support data for a given task: Proc. First International Conference on Knowledge Discovery and Data Mining. AAAI Press, Cambridge, pp 252–257
23. Shioda R (2003) Integer Optimization in Data Mining. PhD thesis, MIT, Cambridge
24. Tung AK, Hou J, Han J (2001) Spatial clustering in the presence of obstacles In: Proceedings ICDE-2001: 17th International Conference on Data Engineering. Heidelberg, pp 359–367
25. Vapnik VN (1995) The nature of statistical learning. Springer, Berlin

Mathematical Programming Methods in Supply Chain Management

LAURA DI GIACOMO
Dipartimento di Statistica, Probabilità e Statistiche Applicate, Università di Roma "La Sapienza", Rome, Italy

Article Outline

Introduction

Supply chain management (SCM) is the integration of key business processes from end users through the original suppliers to the customers that provides products, services and information that add value to all parties [13,14,17]. It is therefore concerned with the organization, the planning and the qualitative and quantitative determination of material and information flows both in and between facilities (vendors, plants, sites and distribution centres) and between these and the final consumers. It is a set of important activities in all producing facilities and in many organizations [6].

For some restricted production problems, such as determining an optimal control to a chemical plant, suitable experimental designs can be enacted, such as **EV**olutionary **OP**eration (**EVOP**) [4], Taguchi methods [19], or more complex experimental designs such as Latin squares, Greek squares and block designs [21].

In general, verification procedures, based on experimental replication and design, cannot be used in the applied sciences, as non-reversible and unpredictable changes in the environment occur [18], and the outcome of the plans cannot be imputed to the effect of the decision taken rather than to an environmental change, so there can be no evaluation of the relevance of a formulated supply chain plan.

Thus more complex methodologies than those based on experimental verification, such as intuition experimental design or anecdotal evidence, must be posited. The solution of any SCM problem must be undertaken with respect to a set of principles and procedures to ensure the formulation of expectationally valid plans, i. e. robust valid feasible policies are determined.

To enable management to formulate good SCM plans, the methodology proposed should be analysed for its logical consistency, its statistical correctness and its adequacy. Essentially, it must be shown that from acceptable premises or axioms, by suitable deductions a policy is formulated (syntactically correctness). Since this policy cannot be tested, but only applied, it must also be shown that in many other historical derivations the policies that were formulated by this methodology turned out to be applicable (semantically adequate).

A dynamic non-linear stochastic system formulation of an SCM model must be estimated and applied. Thus an optimization algorithm must be specified and solved which determines simultaneously the adequate functional form, its parameterization and the optimal control [6].

Definitions

In this section some fundamental definitions will be given.

A dynamical system is a precise mathematical object [16], and given the flows of the activities of the phenomenon, the input-output relationships must be determined by appropriate estimation methods.

Not every relationship can be modelled by mathematical system theory, since a representation which is non-anticipatory is required [16], while the condition that the functionals be sufficiently smooth which was previously required may be waived.

Dynamical systems have been defined at a high level of generality to refine concepts and perceive unity in a diversity of applications, and by appropriate modelling whole hierarchies of phenomena can be represented as systems defined at different levels.

Definition 1 ([16]) A dynamical system is a composite mathematical object defined by the following axioms:

1. There is a given time set T, a state set X, a set of input values U, a set of acceptable input functions $\Omega = \omega\colon \Omega \to U$, a set of output values Y and a set of output functions $\Gamma = \gamma\colon \Gamma \to Y$.
2. (Direction of time). T is an ordered subset of the reals.
3. The input space Ω satisfies the following conditions:
   (a) (Non-triviality). Ω is non-empty.

   (b) (Concatenation of inputs). An input segment $\omega_{(t_1,t_2]}$, $\omega \in \Omega$ restricted to $(t_1, t_2] \cap T$. If $\omega, \omega' \in \Omega$ and $t_1 < t_2 < t_3$, there is an $\omega'' \in \Omega$ such that $\omega''_{(t_1,t_2]} = \omega_{(t_1,t_2]}$ and $\omega''_{(t_2,t_3]} = \omega'_{(t_2,t_3]}$.
4. There is a state transition function $\varphi\colon T \times T \times X \times \Omega \rightarrow X$ whose value is the state $x(t) = \varphi(t; \tau, x, \omega) \in X$ resulting at time $t \in T$ from the initial state $x = x(\tau) \in X$ at the initial time $\tau \in T$ under the action of the input $\omega \in \Omega$. ϕ has the following properties:
   (a) (Direction of time). ϕ is defined for all $t \geq \tau$, but not necessarily for all $t < \tau$.
   (b) (Consistency). $\varphi(t; t, x, \omega) = x$ for all $t \in T$, all $x \in X$ and all $\omega \in \Omega$.
   (c) (Composition property). For any $t_1 < t_2 < t_3$ there results:

$$\varphi(t_3; t_1, x, \omega) = \varphi(t_3; t_2, \varphi(t_2; t_1, x, \omega), \omega)$$

   for all $x \in X$ and all $\omega \in \Omega$.
   (d) (Causality). If $\omega, \omega' \in \Omega$ and $\omega_{(\tau,t]} = \omega'_{(\tau,t]}$ then $\varphi(t; \tau, x, \omega) = \varphi(t; \tau, x, \omega')$.
5. There is a given readout map $\eta\colon T \times X \rightarrow Y$ which defines the output $y(t) = \eta(t, x(t))$. The map $(\tau, t] \rightarrow Y$ given by $\sigma \mapsto \eta(\sigma, \varphi(\sigma, \tau, x, \omega))$, $\sigma \in (\tau, t]$, is an output segment, that is the restriction $\gamma_{(\tau,t]}$ of some $\gamma \in \Gamma$ to $(\tau, t]$.

The following mathematical structures in Definition 1 will be indicated by:
- The pair (t, x), $t \in T, x \in X\ \forall t$ is called an event;
- The state transition function $\varphi(x_t, u_t)$ is called a trajectory.

Phenomena may also be modelled through dynamical systems in the input/output sense, which reflect an experimental design or a simulative approach, long applied in science.

Definition 2 A dynamical system in an input/output sense is a composite mathematical object defined as follows:
1. There are given sets T, U, Ω, Y and Γ satisfying all the properties required by Definition 1.
2. There is a set A indexing a family of functions

$$\mathcal{F} = \{f_\alpha\colon T \times \Omega \rightarrow Y, \alpha \in A\};$$

   each member of $\mathcal{F}$ is written explicitly as $f_\alpha(t, \omega) = y(t)$, which is the output resulting at time t from the input ω under the experiment α. Each f_α is called an input/output function and has the following properties:
   (a) (Direction of time). There is a map $\iota\colon A \rightarrow T$ such that $f_\alpha(t, \omega)$ is defined for all $t \geq \iota(\alpha)$.
   (b) (Causality). Let $\tau, t \in T$ and $\tau < t$. If $\omega, \omega' \in \Omega$ and $\omega_{(\tau,t]} = \omega'_{(\tau,t]}$, then $f_\alpha(t, \omega) = f_\alpha(t, \omega')$ for all α such that $\tau = \iota(\alpha)$.

While the input/output approach may determine a family of functions, which generally vary over the time interval of realization and across instances, the state-space approach represents the trajectories in the way indicated, through a unique function. The latter approach is intuitively more appealing, especially in applications.

The representations are equivalent. It is easy to transform a given system from a state space formulation into an input/output formulation and vice versa [2,16], so each may be used as convenience suggests.

It cannot be assumed generally that a dynamical system satisfies the conditions of smoothness, nor that it will meet the necessary and sufficient conditions for an optimal control to exist. Thus in general, the dynamical systems to be dealt with may have an awkward structure, but through the combined estimation and optimization approach a sufficiently good approximation may be obtained with the required characteristics [6].

A sufficiently general representation of a dynamical system may be formulated by applying Definition 1, recalling the equivalence of an input/output system and a system in state form:

$$x_{t+1} = \varphi(x_t, u_t)\,, \tag{1}$$

$$y_t = \eta(x_t)\,, \tag{2}$$

where $x_t \in X \subseteq R^r$ may simply be taken as an r-dimensional vector in a Euclidean space X, indicating the state of the system at time t, $u_t \in U \subseteq R^q$ may be taken as a q-dimensional vector in a Euclidean subspace U of control variables and $y_t \in Y \subseteq R^p$ is a p-dimensional vector in a Euclidean space Y of output variables, in line with Definitions 1 and 2.

The definition of a dynamical system is based on defining an intermediary set of states and a transition function or a family of functions. Neither of these constructions is unique, so if it is desired to represent a SCM system by such structures, equivalence of the possible structures must be shown.

Definition 3 Given two states x_{t_0} and $\hat{x}_{t_0}$ belonging to systems S and $\hat{S}$ which may not be identical but have a common input space Ω and output space Y, the two states are said to be equivalent if and only if for all input segments $\omega_{[t_0,t)} \in \Omega$ the response segment of S starting in state x_{t_0} is identical with the response segment of $\hat{S}$ starting in state $\hat{x}_{t_0}$; that is

$$x_{t_0} \cong \hat{x}_{t_0} \Leftrightarrow \eta(t, \varphi(x_{t_0}, \omega_{[t_0,t)})) = \hat{\eta}(t, \hat{\varphi}(\hat{x}_{t_0}, \omega_{[t_0,t)})) \quad \forall t \in T, t_0 \leq t, \forall \omega_{[t_0,t)} \in S, \hat{S}\,. \tag{3}$$

Systems S and $\hat{S}$ may be two models of a SCM system solved with different control policies, or they may be various alternative models of the phenomenon.

Definition 4 A system is in reduced form if there are no distinct states in its state space which are equivalent to each other.

Definition 5 Systems S and $\hat{S}$ are equivalent $S \equiv \hat{S}$ if and only if to every state in the state space of S there corresponds an equivalent state in the state space of $\hat{S}$ and vice versa.

Some important conditions are required to make the representation of the SCM adequate.

The conditions of the system are:

- Reachability
- Controllability
- Observability
- Stability

These conditions are very important since they allow trajectories to be defined, the initial point of trajectories to be determined and their stability properties to be derived. Moreover they can be applied at any moment in time to determine if the goals of the SCM are still attainable and at what cost. Reachability, controllability and stability are seldom formally examined and yet at every period exogenous events can arise to nullify even the best formulated plan, so these are important instruments for SCM [6].

An important property which distinguishes dynamical systems from their counterparts derived in comparative statics is the distinction between systems which are simply equivalent and those which are multiply equivalent [6]. This distinction is crucial if dynamical systems are considered, while with comparative static models the distinction does not apply. This is one of the many reasons that one should insist on solving SCM dynamic estimation problems with a data-driven formulation [6].

The dynamical system representation of a SCM system permits one to verify its specification, whether the optimal control which determines the final event is reachable, if the system is controllable throughout the sequence of events comprising the trajectory, if the system is observable and finally if the given solution is stable, so that small perturbations will not give rise to explosive perturbations or to chaotic behaviour. In so doing crucial questions which are important to management can be answered.

If these conditions are not verified, this will suggest strategic changes to the SCM system or profound modification of policies, aspects which are difficult to determine in advance.

Computationally, these aspects are handled by adding appropriate constraints in the mathematical program [6].

Formulation

Consider the monitoring of a set of activities in time of a supply chain at a given level of aggregation, which may be at the department, plant or firm level, or a hierarchical system developed through all these organizational structures. Although the accuracy of the representation may depend on the sampling strategy and the time interval, these aspects will not be considered here.

Thus a given finite-dimensional estimation and optimization problem will be considered which may well be non-linear and dynamic.

Consider the data set of a phenomenon consisting of measurements (y^t, x^t, u^t) over $(t = 1, 2, \dots, T)$ periods, where it is assumed that $y_t \in R^p$ is a p-dimensional vector, while $x_t \in R^r$ is an r-dimensional vector of explanatory or state variables of the dynamic process of dimension. Also, u_t is a q-dimensional vector of control variables. It is desired to determine functional forms $\varphi\colon R^{r+q} \to R^r$ and $\eta\colon R^r \to R^p$ and a set of suitable coefficients $\Theta \in R^m$ such that:

$$\text{Min} \quad J = \sum_{t=T+1}^{\mathcal{T}} c(x_t, u_t, y_t), \tag{4}$$

$$x^{t+1} = \varphi(x^t, u^t, y^t, w^t) \quad \forall t = T+1, \ldots, \mathcal{T}-1, \quad (5)$$

$$y^t = \eta(x^t, u^t, v^t) \quad \forall t = T+1, \ldots, \mathcal{T}, \quad (6)$$

where w^t and v^t are stochastic processes also to be determined.

Equation (4) is the objective function for the supply chain and (5) and (6) are the system equations in state space formulation and a similar representation may be adopted for the the input-output formulation [16,20].

The system (4)–(6) could be estimated by a maximum likelihood method so as to minimize the random errors, indicated by $w^t \in R^r$ and $v^t \in R^p$, such that they will have minimum variance and zero mean value, and then on the quantified model the optimal control problem could be solved, usually through an appropriate optimization problem.

However, for this type of model with serially correlated disturbances, which are also correlated with the control variables, its estimation will be biased and the necessary least-squares properties to ensure an asymptotically correct estimate may only be fulfilled in exceptional cases. Thus the two-stage approach, indicated above, is inappropriate [15].

It is important to apply a suitable data-driven statistical method to determine the most appropriate statistical form and the most precise values of the parameters, as when implemented correctly with regard to an accurately specified functional form. Such a method will provide estimates of parameters that satisfy the statistical properties [1,18].

Suppose that all the statistical properties that a given estimate must fulfil are set up as constraints to the maximum likelihood problem to be solved; then the parameters are defined implicitly by this optimization problem, which can be inserted into the optimal control system for policy determination, so that statistically correct estimates will always result. Thus the solution yielding the best policy can be chosen, where $T+1, \ldots, \mathcal{T}$ is the forecast period, by solving an optimization formulation of this complex problem. By recursing on the specifications, i. e. by changing the functional form, increasingly better fits can be obtained. At each iteration, the best combination of parameterization and policy is obtained.

The unknowns to be determined are the input and output variables considered and the parameters of the functional form specified in the current iteration, indicated as $\Theta = \{\theta_1, \theta_2\} \subset R^m$, respectively for (5) and (6). Note that m may be much larger than $2r + q + p + 1$, the number of variables present in each system, since the system is non-linear.

The mathematical program will be formulated with respect to the residual variables, but it is immediate that for a given functional form, the unknown parameters will be specified and thus the unknowns of the problem will also be defined and available. Thus the mathematical program is fully specified for each functional form to be considered.

Using the notation given above, the residual terms are given from Eqs. (5) and (6) as:

$$w_i = \hat{x}_{i+1} - \varphi(\hat{x}_i, \hat{u}_i, \hat{y}_i : \theta_1) \quad i = 1, 2, \ldots, N, \quad (7)$$

$$v_i = \hat{y}_{i+1} \eta(x_i, u_i, v_i : \theta_2) \quad i = 1, 2, \ldots, N, \quad (8)$$

where $\hat{\cdot}$, as usual, indicates the historical values of a variable, and thus suitable values of θ_1 and θ_2 must be determined by the mathematical program such that all the constraints expressed in terms of $w_i, v_i \forall i$ are specified.

Methods and Applications

Given an experimental data set obtained as a set of measurements of the operation of a phenomenon, it is desired to determine a suitable representation of it in the form of a model, so as to determine a suitable control law for the model which can then be extended to the phenomenon and thus obtain a better performance [3].

Except in some simple cases, the representation assumed by the model and the data that have been collected will condition the results obtainable by enacting the control law. For models that are non-linear in the parameters, the interaction between the estimation of these and the determination of an optimal control is much more complex than the linear case requiring the solution of constrained optimization problems which will determine simultaneously the best estimates and the optimal control.

Consider the availability of a given data set containing a number of sets of time series data or cross-sectional data. To determine from these data a suitable model, a functional form must be selected and a set of suitable parameters must be estimated which will satisfy all the conditions on the model and permit the determination of a suitable set of control variables, which will define an optimal control with respect to a predefined merit function.

Thus from the data set it is desired to derive a sufficiently accurate model of the phenomenon, which can then be used in control and in prediction.

Statistical estimation methods are important because, when implemented correctly with regard to an accurately specified functional form, they will provide estimates of parameters that have the following properties [1,18]:

1. The parameter estimates are unbiased.
   - As the size of the data set grows larger, the estimated parameters tend to their true values.
2. The parameter estimates are consistent, which will then satisfy the following conditions:
   - The estimated parameters are aymptotically unbiased.
   - The variance of the parameter estimate must tend to zero as the data set tends to infinity.
3. The parameter estimates are asymptotically efficient.
   - The estimated parameters are consistent.
   - The estimated parameters have smaller asymptotic variance as compared to any other consistent estimator.
4. The residuals have minimum variance, which is ensured by the following factors:
   - The variance of the residuals must be minimum.
   - The residuals must be homoscedastic.
   - The residuals must not be serially correlated.
5. The residuals are unbiased (have zero mean).
6. The residuals have a non-informative distribution (usually, a Gaussian distribution).
   - If the distribution of the residuals is informative, the extra information could somehow be obtained, reducing the variance of the residuals, their bias etc., with the result that better estimates are obtained.

In short, through correct implementation of statistical estimation techniques the estimates are as close as possible to their true values, all the information that is available is applied and the uncertainty surrounding the estimates and the data fit is reduced to the maximum extent possible. Thus the estimates of the parameters, which satisfy all these conditions, are the 'best' possible in a 'technical' sense [1].

To ensure that all the statistical properties which the given estimates of the residuals must fulfil are satisfied at every iteration, instead of solving an unconstrained maximum likelihood or least-squares problem [15], the required statistical properties of the estimates are set up as constraints, together with the specification of the model of the phenomenon, and this global optimization problem is solved for all the undetermined variables.

The parameters of this model to be estimated are defined implicitly through those constraints which define the statistical conditions. On solving the global optimization problem, the parameter estimates that result will be defined for the optimal control system for the policy determination so that statistically correct estimates will always result.

The procedure adopted can be specified easily by using the same notation as above and by adding an additional set of constraints which express the statistical conditions that must be satisfied by the estimates.

Let

$$\gamma(x_{i+1}, x_i, u_i, y_{i+1}, y_i, w_i, v_i, \theta_1, \theta_2) \geq 0 \qquad i = 1, 2, \ldots, N, N+1, \ldots, \mathcal{T} \tag{9}$$

be the set of conditions to be satisfied to obtain estimates, if they exist, which satisfy the statistical properties indicated above. Then the optimization problem to be solved is:

$$\text{Min} \quad J = \sum_{i=N+1}^{\mathcal{T}} c(x_i, u_i, y_i), \tag{10}$$

$$x_{i+1} = \varphi(x_i, u_i, y_i, w_i : \theta_1) \quad i = 1, 2, \ldots, \mathcal{T}, \tag{11}$$

$$y_{i+1} = \eta(x_i, u_i, v_i : \theta_2) \quad i = 1, 2, \ldots, \mathcal{T}, \tag{12}$$

$$0 \leq \gamma(x_{i+1}, x_i, u_i, y_{i+1}, y_i, w_i, v_i, \theta_1, \theta_2) \qquad i = 1, 2, \ldots, \mathcal{T}. \tag{13}$$

Thus the solution yielding the best policy can be chosen by solving an optimization formulation of this complex problem. By recursing on the specifications, i. e. by changing the functional form, and increasing the number of independent variables considered, increasingly better fits can be obtained, with regard to both the historical data and the predicted optimal control policy.

Models

Many models of industrial, extractive and financial activities require the integration of key processes, but the most essential aspect is to formulate precise information where it is most needed [11,12].

Many optimization models solve satisfactorily supply chain problems, but apparently no model except this one integrates information and allocation of goods and operations dynamically.

This algorithm instead solves the combined problem, as has been shown elsewhere [6], while a theory-driven modelling approach to the problem, using models consisting of two stages, an identification stage and an optimization stage, can be shown to be dominated by this data-driven approach.

At present, this seems to be the only viable approach to solving such complex problems.

Cases

Some non-typical SCM problems are indicated here: dynamical supply chain management problems for perforation oil wells and for finance. Industrial SCM models are given elsewhere [3,8,9,10].

Dynamic Supply Chain Management Problem for Extraction Activities

The perforation of oil wells consists of a number of operations to drive the bit head lower and lower while ensuring normal functions on the equipment and the operations. To this end complex measurements are executed by software systems indicated as mudlogging systems. These measurements are designed to assist the operator in controlling the perforation rate of the bit head by monitoring a number of crucial operations periodically.

The settings of some of these operations affect the rate of perforation, and therefore it is considered extremely useful to dispose of measurements of these variables and have predictions over the next few periods of the possible advancement of the bit head, or of the rate of perforation, and so enable an optimal control of the process to be formulated [5].

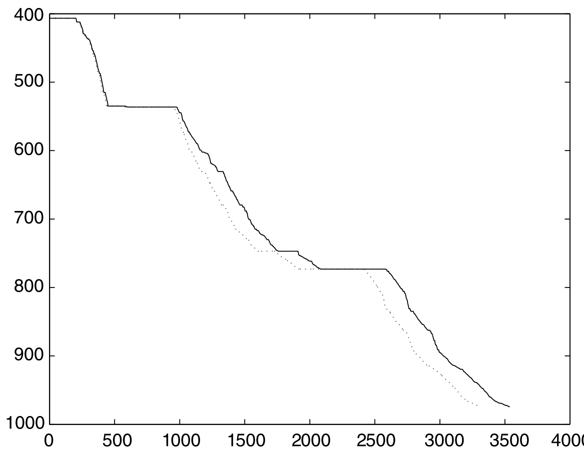
It should be mentioned that periodically the drilling process must be halted so that the boring can be lined with suitable materials. Also, one of the most important elements of the process is to keep circulating around the bit-head assembly a concentration of mud lubricants, indicated as mud, which gives the name to the measuring process. Recall that all these flows and operations occur in time, so it is considered crucial to specify dynamic models, unless it is desired to determine the steady-state rates of the eventual process.

In fact oil drilling processes can be considered as complex supply chain systems with many phases and many operations.

The determination of optimal control policies in processes for the extraction of oil from underground require that they be formulated as formal procedures, which are syntactically correct and semantically adequate, so as to permit management to make the necessary investments, not on hearsay or clever promotional activities, but on the basis of rational knowledge and confidence in the application.

Figure 1 shows an optimal SCM plan compared to the actual historical plan implemented. The predicted trajectory is superimposed on the actual time path of the perforation process, thus respecting all the interruptions and periods of halting.

In Table 1, six instances to determine optimal controls are indicated, and each entry reflects the drilling experience of the given well for that week with regard to the given period. From the active perforation intervals an intial period was selected randomly and the optimal



Mathematical Programming Methods in Supply Chain Management, Figure 1
Example of drilling for oil: real-time path (*continuous*) and optimal control path (*dashed*) for the well

Mathematical Programming Methods in Supply Chain Management, Table 1
Optimal predicted versus actual increment for 6 oil wells over 192 periods (8 h) in metres

Well and week	Optimal increment	Real increment	% difference
FT02D 9	114.0	73.3	35.70
FT02D 16	116.7	83.8	28.19
FT02D 23	73.7	13.45	81.75
GX01 3	94.8	72.2	23.84
GX01 11	57.9	18.8	67.53

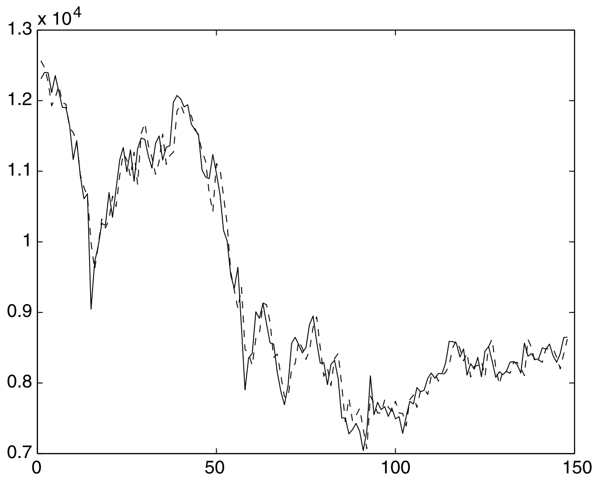
control was defined for the next 192 periods (8 h). The average predicted increment in depth attainable over the actual one was more than 30% on average.

Dynamic Supply Chain Management Problem for Finance

The prediction of future quotations on stock exchange indices is important and consists of the basic instrument to handle financial supply chain management systems. A financial supply chain system must consider many types of financial intermediaries, many types of stocks and stock indices and many types of operations. Further, there are many possibilities for managing the monetary holdings, so that a full SCM system is envisaged as defined above [7].

Consider the Dow-Jones Industrial Average (DJIA) stock exchange index over a period of 3 years starting in April 2001, as shown in Fig. 2, where the continuous line indicates the actual quotations, week by week over the period, while the 1-week-ahead predictions are given by the dashed line. As can be easily seen, the two curves almost coincide, which implies that the predictions 1 week ahead are very good.

Instead, in Table 2 a period of 5 weeks is considered from April 16, 2004 to May 14, 2004. The quotations are given every Friday evening at closing time, while the predictions are made on Fridays just after closing time. Thus on April 9 predictions were made for the next 5 weeks, as indicated in the second row of the table. After closing on April 16, 2004, predictions were made for 4 weeks only and are depicted in the third row of the table and so on for the subsequent weeks. Finally, in the last row the closing quotations for the week are given.



Mathematical Programming Methods in Supply Chain Management, Figure 2
Weekly time series of the Dow-Jones Industrial Average

Mathematical Programming Methods in Supply Chain Management, Table 2
Results for prediction of the Dow-Jones Industrial Average, 147 periods

Period	16/4	23/4	30/4	7/5	14/5
9/4	8652.86	8568.44	9304.81	13306.5	9958.15
16/4	..	8646.28	8552.54	11514.3	11000.3
23/4	..	..	8820.73	8806.51	4700.47
30/5	..	..	..	8518.12	8361.19
7/5	..	..	..	..	8343.35
Index	8712.88	8855.03	8538.03	8505.54	8432.25

This table allows one to determine with the appropriate portfolio model suitable financial policies to formulate optimal financial SCM plans [7].

Conclusions

Optimal dynamic SCM policies may be obtained by a correct application of statistical inference and mathematical programming techniques.

It has been indicated that these policies are expectationally valid, which implies that they are syntactically correct and semantically adequate.

Computational evidence has been presented and indicated in the references.

See also

- ► Generalizations of Interior Point Methods for the Linear Complementarity Problem
- ► Simultaneous Estimation and Optimization of Nonlinear Problems

References

1. Amemiya T (1985) Advanced Econometrics. Blackwell, Oxford
2. Aoki M (1976) Optimal Control and System Theory in Dynamic Economic Analysis. North-Holland, New York
3. Bilardo U, Di Giacomo L, Patrizi G (2007) Dynamic management of petroleum drilling operations. Submitted, June 2007, pp 1–35
4. Box GEP, Draper NR (1969) Evolutionary Operation: A Statistical Method for Process Improvement. Wiley, New York
5. Bradley HB (1987) Petroleum Engineering Handbook. Society of Petroleum Engineers, Richardson, TX
6. Di Giacomo L, Patrizi G (2006) Dynamic nonlinear modelization of operational supply chain systems. J Global Optim 34:503–534
7. Di Giacomo L, Patrizi G (2006) Optimal dynamic nonlinear prediction methods for management of financial instruments. Technical report, Dipartimento di Statistica, Probabilitá e Statistiche Applicate, Universitá di Roma, La Sapienza, Rome
8. Di Giacomo L, Patrizi G (2007) Methodological analysis of supply chain management applications: towards a normative theory of supply chain management. Submitted for publication, copy at: http://banach.sta.uniroma1.it/patrizi/, pp 1–41
9. Di Giacomo L, Patrizi G (2007) Optimal control of well drilling operations. Submitted, pp 1–31
10. Di Giacomo L, Patrizi G, Di Lena E, Pomaranzi L, Sensi F (2008) C.A.s.S.a.n.D.r. A computerized analysis for supply chain distribution activity. In: Bertazzi L, Speranza MG, van Nunen J (eds) Lectures Notes in Economics ans Mathematical Systems. Springer, Berlin
11. Eksioglu SD, Romeijn HE, Pardalos PM (2006) Cross-facility management of production and transportation planning problem. Comput Oper Res 33:3231–3251
12. Geunes J, Akçali E, Pardalos P, Romeijn H, Shen Z-J (2005) Applications of Supply Chain Management and E-Commerce Research. Springer, New York
13. Geunes J, Pardalos P (2005) Supply Chain Optimization. Springer, New York
14. Geunes J, Pardalos P, Romejin H (2005) Supply Chain Management: Models, Applications, and Research Directions. Springer, New York
15. Jennrich RI (1969) Asymptotic properties of non-linear least squares estimators. Ann Math Statist 40:633–643
16. Kalman RE, Falb PL, Arbib MA (1969) Topics in Mathematical System Theory. McGraw-Hill, New York
17. Lambert DM, Cooper MC, Pagh JD (1998) Supply chain management: Implementation issues and research opportunities. Int J Logist Manag 9:1–17
18. Malinvaud E (1978) Méthodes Statistiques de l'économétrie. Dunod, Paris
19. Ross PJ (1988) Taguchi techniques for quality engineering: loss function, orthogonal experiments, parameter and tolerance design. McGraw-Hill, New York
20. Söderström T, Stoica P (1989) System Identification. Prentice-Hall, New York
21. Vajda S (1967) Mathematics of Experimental Design: incomplete block designs and Latin squares. Griffin, London

Matrix Completion Problems

MONIQUE LAURENT
CWI, Amsterdam, The Netherlands

MSC2000: 05C50, 15A48, 15A57, 90C25

Article Outline

Keywords

Partial matrix; Completion of matrices

Matrix completion problems are concerned with determining whether partially specified matrices can be completed to fully specified matrices satisfying certain prescribed properties. In this article we survey some results and provide references about these problems for the following matrix properties: positive semidefinite matrices, Euclidean distance matrices, completely positive matrices, contraction matrices, and matrices of given rank. We treat mainly optimization and combinatorial aspects.

Introduction

A *partial matrix* is a matrix whose entries are specified only on a subset of its positions; a *completion* of a partial matrix is simply a specification of the unspecified entries. *Matrix completion problems* are concerned with determining whether or not a completion of a partial matrix exists which satisfies some prescribed property. We consider here the following matrix properties: positive (semi) definite matrices, distance matrices, completely positive matrices, contraction matrices, and matrices of given rank; definitions are recalled below.

In what follows, x^*, A^* denote the conjugate transpose (in the complex case) or transpose (in the real case) of vector x and matrix A. A square real symmetric or complex Hermitian matrix A is *positive semidefinite* (psd) if $x^*Ax \geq 0$ for all vectors x and *positive definite* (pd) if $x^*Ax > 0$ for all vectors $x \neq 0$; then we write: $X \succcurlyeq 0$ ($X \succ 0$). Equivalently, A is psd (respectively, pd) if and only if all its eigenvalues are nonnegative (respectively, positive) and A is psd if and only if $A = BB^{\mathsf{T}}$ for some matrix B. A matrix A is said to be *completely positive* if $A = BB^{\mathsf{T}}$ for some nonnegative matrix B. An $n \times n$ real symmetric matrix $D = (d_{ij})$ is a *Euclidean distance matrix* (abbreviated as *distance matrix*) if there exist vectors $v_1, \ldots, v_n \in \mathbf{R}^k$ (for some $k \geq 1$) such that, for all $i, j = 1, \ldots, n$, d_{ij} is equal to the square of the Euclidean distance between v_i and v_j. Finally, a (rectangular) matrix A is a *contraction matrix* if all its singular values (that is, the eigenvalues of A^*A) are less than or equal to 1.

The set of positions corresponding to the specified entries of a partial matrix A is known as the *pattern* of A. If A is an $n \times m$ partial matrix, its pattern can be represented by a bipartite graph with node bipartition $[1, n] \cup [1, m]$ having an edge between nodes $i \in [1, n]$ and $j \in [1, m]$ if and only if entry a_{ij} is specified.

When asking about existence of a psd completion of a partial $n \times n$ matrix A, it is commonly assumed that all diagonal entries of A are specified (which is no loss of generality if we ask for a pd completion); moreover, it can obviously be assumed that A is *partial Hermitian*, which means that entry a_{ji} is specified and equal to $a_{ij}*$ whenever a_{ij} is specified. Hence, in this case, complete information about the pattern of A is given by the graph $G = ([1, n], E)$ with node set $[1, n]$ and whose edge set E consists of the pairs ij $(1 \leq i < j \leq n)$ for which a_{ij} is a specified entry of A. The same holds when dealing with distance matrix completions (in which case diagonal entries can obviously be assumed to be equal to zero).

An important common feature of the above matrix properties is that they possess an 'inheritance structure'. Indeed, if a partial matrix A has a psd (pd, completely positive, distance matrix) completion, then every principal specified submatrix of A is psd (pd, completely positive, a distance matrix); similarly, if a partial matrix A admits a completion of rank $\leq k$, then every specified submatrix of A has rank $\leq k$. Hence, having a completion of a certain kind imposes certain 'obvious' necessary conditions. This leads to asking which are the patterns for the specified entries that insure that if the obvious necessary conditions are met, then there will be a completion of the desired type; therefore, this introduces a combinatorial aspect into matrix completion problems, as opposed to their analytical nature.

In this article we survey some results and provide references for the various matrix completion problems mentioned above, concerning optimization and combinatorial aspects of the problems. See [32,47] for more detailed surveys on some of the topics treated here.

Positive Semidefinite Completion Problem

We consider here the following *positive (semi) definite completion problem* (PSD): Given a partial Hermitian matrix $A = (a_{ij})_{ij \in S}$ whose entries are specified on a subset S of the positions, determine whether A has a psd (or pd) completion; if, yes, find such a completion. (Here, S is generally assumed to contain all diagonal positions.)

This problem belongs to the most studied matrix completion problems. This is due, in particular, to its many applications, e. g., in probability and statistics, systems engineering, geophysics, etc., and also to the fact that positive semidefiniteness is a basic property which is closely related to other matrix properties like being a contraction or distance matrix. Equivalently, (PSD) is the problem of testing feasibility of the following system (in variable $X = (x_{ij})$):

$$X \succeq 0, \qquad x_{ij} = a_{ij} \quad (ij \in S). \tag{1}$$

Therefore, (PSD) is an instance of the following *semidefinite programming problem* (P): Given Hermitian matrices $A_1, \ldots, A_m$ and scalars $b_1, \ldots, b_m$, decide whether the following system is feasible:

$$X \succeq 0, \qquad A_j \cdot X = b_j \quad (j = 1, \ldots, m) \qquad (2)$$

(where $A \cdot X := \sum_{i,j=1}^{n} a_{ij} * x_{ij}$ for two Hermitian $(n \times n)$-matrices A and X).

The exact complexity status of problems (PSD) and (P) is not known; in particular, it is not known whether they belong to the complexity class *NP*. However, it is shown in [60] that (P) is neither *NP*-complete nor co-*NP*-complete if $NP \neq$ co-NP. However, the semidefinite programming problem and, thus, problem (PSD) can be solved with an arbitrary precision in polynomial time. This can be done using the ellipsoid method (since one can test in polynomial time whether a rational matrix A is positive semidefinite and, if not, find a vector x such that $x^*Ax < 0$; cf. [24]), or interior point methods (cf. [3,27,56]). There has been a growing interest in semidefinite programming in the recent years (1994), which is due, in particular, to its successful application to the approximation of hard combinatorial optimization problems (cf. the survey [20]). This has prompted active research on developing interior point algorithms for solving semidefinite programming problems; the literature is quite large, see [64,65] for extensive information. Numerical tests are reported in [34] where an interior point algorithm is proposed for the approximate psd completion problem; it permits to find exact completions for random instances up to size 110.

Moreover, it is shown in [59] that problem (P) can be solved in polynomial time (for rational input data A_j, b_j) if either the number m of constraints, or the order n of the matrices X, A_j in (2) is fixed (cf. also [9]). Moreover, under the same assumption, one can test in polynomial time the existence of an integer solution and find one if it exists [39].

Call a partial Hermitian matrix A *partial psd* (respectively, *partial pd*) if every principal specified submatrix of A is psd (respectively, pd). As mentioned in the Introduction, being partial psd (pd) is an obvious necessary condition for A to have a psd (pd) completion. In general, this condition is not sufficient; for instance, the partial matrix:

$$A = \begin{pmatrix} 1 & 1 & ? & 0 \\ 1 & 1 & 1 & ? \\ ? & 1 & 1 & 1 \\ 0 & ? & 1 & 1 \end{pmatrix}$$

('?' indicates an unspecified entry) is partial psd, yet no psd completion exists; note that the pattern of A is a circuit of length 4. Call a graph *chordal* if it does not contain any circuit of length ≥ 4 as an induced subgraph; chordal graphs occur in particular in connection with the Gaussian elimination process for sparse pd matrices (cf. [21,61]). (An *induced subgraph* of a graph $G = (V, E)$ being of the form $H = (U, F)$ where $U \subseteq V$ and $F := \{ij \in E\colon i, j \in U\}$.) It is shown in [23] that every partial psd matrix with pattern G has a psd completion if and only if G is a chordal graph; the same holds for pd completions. This extends an earlier result from [16] which dealt with 'block-banded' partial matrices; in the Toeplitz case (all entries equal along a band), one finds the classical Carathéodory–Fejér theorem from function theory.

The proof from [23] is constructive and can be turned into an algorithm with a polynomial running time [48]. Moreover, it is shown in [48] that (PSD) can be solved in polynomial time when restricted to partial rational matrices whose pattern is a graph having a fixed minimum fill-in; the *minimum fill-in* of a graph being the minimum number of edges needed to be added in order to obtain a chordal graph. This result is based on the above mentioned results from [39,59] concerning the polynomial time solvability of (integer) semidefinite programming with a fixed number m of linear constraints in (2).

The result from [23] on psd completions of partial matrices with a chordal pattern has been generalized in various directions; for instance, considering general inertia possibilities for the completions ([17,35]), or considering completions with entries in a function ring [37].

If A is a partial matrix having a pd completion, then A has a unique pd completion with maximum determinant (this unique completion being characterized by the fact that its inverse has zero entries at all unspecified positions of A) [23]. In the case when the pattern of A is chordal, explicit formulas for this maximum determinant are given in [7]. The paper [52] considers the more general problem of finding a maximum determinant psd completion satisfying some additional linear constraints.

Further necessary conditions are known for the existence of psd completions. Namely, it is shown in [8] that if a partial matrix $A = (a_{ij})$ with pattern G and diagonal entries equal to 1 is completable to a psd matrix, then the associated vector $x := (\arccos(a_{ij})/\pi)_{ij \in E}$ satisfies the inequalities:

$$\sum_{e \in F} x_e - \sum_{e \in C \setminus F} x_e \leq |F| - 1$$
$$\text{for all } F \subseteq C, \quad C \text{ circuit in } G, \ |F| \text{ odd.} \quad (3)$$

Moreover, any partial matrix with pattern G satisfying (3) is completable to a psd matrix if and only if G does not contain a homeomorph of K_4 as an induced subgraph (then, G is also known as *series-parallel graph*) [44]. (Here, K_4 denotes the complete graph on 4 nodes and a *homeomorph* of K_4 is obtained by replacing the edges of K_4 by paths of arbitrary length.) The patterns G for which every partial psd matrix satisfying (3) has a psd completion are characterized in [6]; they are the graphs G which can be made chordal by adding a set of edges in such a way that no new clique of size 4 is created. Although (3) can be checked in polynomial time for rational x [5], the complexity of problem (PSD) for series-parallel graphs (or for the subclass of circuits) is not known. A strengthening of condition (3) (involving cuts in graphs) is formulated in [44].

Another approach to problem (PSD) is considered in [1,28], which is based on the study of the cone

$$\mathcal{P}_G := \left\{ X = (x_{ij})_{i,j \in V} : \begin{array}{c} X \succeq 0, \ x_{ij} = 0 \\ \forall i \neq j, \quad ij \notin E \end{array} \right\}$$

associated to graph $G = (V, E)$. Indeed, it is shown there that a partial matrix A with pattern G has a psd completion if and only if

$$\sum_{i \in V} a_{ii} x_{ii} + \sum_{\substack{i \neq j, \\ ij \in E}} a_{ij} x_{ij} \geq 0, \quad \forall X \in \mathcal{P}_G. \quad (4)$$

Obviously, it suffices to check (4) for all X *extremal* in $\mathcal{P}_G$ (i. e., X lying on an extremal ray of the cone $\mathcal{P}_G$).

Define the *order* of G as the maximum rank of an extremal matrix in $\mathcal{P}_G$. The graphs of order 1 are precisely the chordal graphs [1,58] and the graphs of order 2 have been characterized in [46]. One might reasonably expect that problem (PSD) is easier for graphs having a small order. This is indeed the case for graphs of order 1; the complexity of (PSD) remains however open for the graphs of order 2 (partial results are given in [48]).

Euclidean Distance Matrix Completion Problem

We consider here the *Euclidean distance matrix completion problem* (abbreviated as *distance matrix completion problem*) (EDM): Given a graph $G = (V = [1, n], E)$ and a real partial symmetric matrix $A = (a_{ij})$ with pattern G and with zero diagonal entries, determine whether A can be completed to a distance matrix; that is, whether there exist vectors $v_1, \ldots, v_n \in \mathbf{R}^k$ for some $k \geq 1$ such that

$$a_{ij} = \left\| v_i - v_j \right\|^2 \quad \text{for all } ij \in E. \quad (5)$$

(here, $\|v\| = \sqrt{\sum_{h=1}^{k} v_h^2}$ denotes the Euclidean norm of $v \in \mathbf{R}^k$.) The vectors $v_1, \ldots, v_n$ are then said to form a *realization* of A. A variant of problem (EDM) is the *graph realization problem* (EDMk), obtained by letting the dimension k of the space where one searches for a realization of A be part of the input data.

Distance matrices are a central notion in the area of distance geometry; their study was initiated by A. Cayley in the 18th century and it was continued in particular by K. Menger and I.J. Schoenberg in the 1930s. They are, in fact, closely related to psd matrices. The following basic connection was established in [63]. Given a symmetric $(n \times n)$-matrix $D = (d_{ij})_{i,j=1}^{n}$ with zero diagonal entries, consider the symmetric $((n-1) \times (n-1))$-matrix $X = (x_{ij})_{i,j=1}^{n-1}$ defined by

$$x_{ij} = \frac{1}{2}(d_{in} + d_{jn} - d_{ij}) \quad \text{for all } i, j = 1, \ldots, n-1. \quad (6)$$

Then, D is a distance matrix if and only if X is psd; moreover, D has a realization in the k-space if and only if X has rank $\leq k$. Other characterizations are known for distance matrices. As the literature on this topic is quite large, see the monographs [11,13,14], where further references can be found.

Problems (EDM) and (EDMk) have many important applications; for instance, to multidimensional scaling problems in statistics (cf. [49]) and to position-location problems, i. e., problem (EDMk) mostly in dimension $k \leq 3$. A much studied instance of the latter problem is the molecular conformation problem in chemistry; indeed, nuclear magnetic resonance spectroscopy permits to determine some pairwise interatomic distances, the question being then to reconstruct the global shape of the molecule from this partial information (cf. [13,41]).

In view of relation (6), problem (EDM) can be formulated as an instance of the semidefinite programming problem (P) and, therefore, it can be solved with an arbitrary precision in polynomial time. Exploiting this fact, some specific algorithms based on interior point methods are presented in [2] together with numerical tests. Moreover, problem (EDM) can be solved in polynomial time when restricted to partial rational matrices whose pattern is a chordal graph or, more generally, a graph with fixed minimum fill-in [48]; as in the psd case, this follows from the fact (mentioned below) that partial matrices that are completable to a distance matrix admit a good characterization when their pattern is a chordal graph.

While the exact complexity of problem (EDM) is not known, it has been shown in [62] that problem (EDMk) is NP-complete if $k = 1$ and NP-hard if $k \geq 2$ (even when restricted to partial matrices with entries in $\{1, 2\}$). Finding ϵ-optimal solutions to the graph realization problem is also NP-hard for small ϵ ([53]). The graph realization problem (EDMk) has been much studied, in particular in dimension $k \leq 3$, which is the case most relevant to applications. The problem can be formulated as a nonlinear global optimization problem: $f(v)$ such that $v = (v_1, \dots, v_n) \in \mathbf{R}^{kn}$, where the cost function $f(\cdot)$ can, for instance, be chosen as

$$f(v) = \sum_{ij \in E} (\|v_i - v_j\|^2 - a_{ij})^2.$$

Hence, $f(\cdot)$ is zero precisely when the v_i's provide a realization of the partial matrix A. This optimization problem is hard to solve (as it may have many local optimum solutions). Several algorithms have been proposed in the literature; see, in particular, [13,19,26,29,31,41,54,57]. They are based on general techniques for global optimization like tabu and pattern search [57], the continuation approach (which consists of transforming the original function $f(\cdot)$ into a smoother function having fewer local optimizers, [53,54]), or divide-and-conquer strategies aiming to break the problem into a sequence of smaller or easier subproblems [13,29,31]. In [29,31], the basic step consist of finding principal submatrices having a unique realization, treating each of them separately and then trying to combine the solutions. Thus arises the problem of identifying principal submatrices having a unique realization, which turns out to be NP-hard [62]. However, several necessary conditions for unicity of realization are known, related with connectivity and generic rigidity properties of the graph pattern [30,67]. Generic rigidity of graphs can be characterized and recognized in polynomial time only in dimension $k \leq 2$ ([42,51]) (cf. the survey [43] for more references).

Call a partial matrix A a *partial distance matrix* if every specified principal submatrix of A is a distance matrix. Being a partial distance matrix is obviously a necessary condition for A to be completable to a distance matrix. It is shown in [4] that every partial distance matrix with pattern G is completable to a distance matrix if and only if G is a chordal graph; moreover, if all specified principal submatrices of the partial matrix A have a realization in the k-space, then A admits a completion having a realization in the k-space.

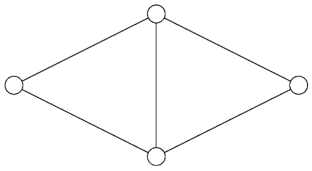
As noted in [33], if a partial matrix A with pattern G is completable to a distance matrix, then the associated vector $x := (\sqrt{a_{ij}})_{ij \in E}$ must satisfy the inequalities:

$$x_e - \sum_{f \in C \setminus \{e\}} x_f \leq 0 \quad \text{for all } e \in C,\ C \text{ circuit in } G. \qquad (7)$$

The graphs G for which every partial matrix (respectively, partial distance matrix) A with pattern G for which (7) holds is completable to a distance matrix, are the graphs containing no homeomorph of K_4 as an induced subgraph [45] (respectively, the graphs that can be made chordal by adding edges in such a way that no new clique of size 4 is created [33]). Note the analogy with the corresponding results for the psd completion problem; some connections between the two problems (EDM) and (PSD) are exposed in [38,45].

Completion to Completely Positive and Contraction Matrices

Call a matrix *doubly nonnegative* if it is psd and entrywise nonnegative. Every completely positive (cp, for short) matrix is obviously doubly nonnegative. The converse implication holds for matrices of order $n \leq 4$ (cf. [22]) and for certain patterns of the nonzero entries in A (cf. [40]). The cp property is obviously inherited by principal submatrices; call a partial matrix A a *partial cp matrix* if every fully specified principal submatrix of A is cp. It is shown in [15] that every partial cp matrix with graph pattern G is completable to a cp matrix if and only if G is a so-called block-clique graph. A *block-clique graph* being a chordal graph in which any two distinct maximal cliques overlap in at most one node or, equivalently, a chordal graph that does not contain an induced subgraph of the form:



Recall that an $(n \times m)$-matrix A is a contraction matrix if all eigenvalues of A^*A are less than or equal to 1 or, equivalently, if the matrix

$$\widetilde{A} = \begin{pmatrix} I_n & A \\ A^* & I_m \end{pmatrix} \tag{8}$$

is positive semidefinite. Call a partial matrix A a *partial contraction* if all specified submatrices of A are contractions. As every submatrix of a contraction is again a contraction, an obvious necessary condition for a partial matrix A to be completable to a contraction matrix is that A be a partial contraction. Thus arises the question of characterizing the graph patterns G for which every partial contraction with pattern G can be completed to a contraction matrix.

As we now deal with rectangular $n \times m$ partial matrices A, their pattern is the bipartite graph G with node set $U \cup V$, where U, V index the rows and columns of A and edges of G correspond to the specified entries of A. We may clearly assume to be dealing with partial matrices whose pattern is a connected graph (as the partial matrices associated with the connected components can be handled separately). Below is an example of a partial matrix A which is a partial contraction, but which is not completable to a contraction matrix:

$$A = \begin{pmatrix} ? & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & ? & \frac{1}{\sqrt{2}} \end{pmatrix}.$$

In fact, the graph pattern displayed in this example is in a sense present in every partial contraction which is not completable to a contraction. Namely, it is shown in [36] that the following assertions (i–iii) are equivalent for a connected bipartite graph G with node bipartition $U \cup V$:

i) Every partial contraction with pattern G can be completed to a contraction;
ii) G does not contain an induced matching of size 2 (i. e., if $e := uv$, $e' := u'v'$ are edges in G with $u \neq u' \in U$, $v \neq v' \in V$, then at least one of the pairs uv', $u'v$ is an edge in G; that is, G is nonseparable in the terminology of [21]);
iii) The graph $\widetilde{G}$ obtained from G by adding all edges uu' $(u \neq u' \in V)$ and vv' $(v \neq v' \in V)$ is chordal.

(Note that the implication iii) → i) is a consequence of the result on psd completions from [23] mentioned in the Section on the positive semidefinite completion problem above, as $\widetilde{G}$ is the graph pattern of the matrix $\widetilde{A}$ defined in (8).)

Rank Completions

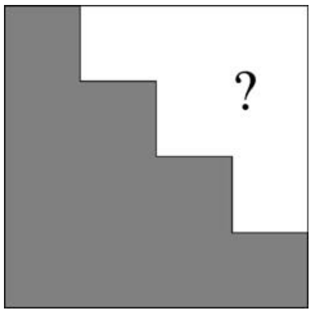
In this section, we consider the problem of determining the possible ranks for the completions of a given partial matrix. For a partial matrix A, let $\mathrm{mr}(A)$ and $\mathrm{MR}(A)$ denote, respectively, the minimum and maximum possible ranks for a completion of A. If B, C are completions of A of respective ranks $\mathrm{mr}(A)$, $\mathrm{MR}(A)$, then changing B into C by changing one entry of B into the corresponding entry of C at a time permits to construct completions realizing all ranks in the range $[\mathrm{mr}(A), \mathrm{MR}(A)]$. Hence, the question is to determine the two extreme values $\mathrm{mr}(A)$ and $\mathrm{MR}(A)$. As we see below, the value $\mathrm{MR}(A)$ can, in fact, be expressed in terms of ranks of fully specified submatrices of A and it can be computed in polynomial time; this constitutes a generalization of the celebrated Frobenius–König theorem (corresponding to the case when specified entries are equal to 0). On the other hand, determining $\mathrm{mr}(A)$ seems to be a much more difficult task.

We first deal with the problem of finding *maximum rank completions*. Let A be an $n \times m$ partial matrix with graph pattern G, i. e., G is the bipartite graph $(U \cup V, E)$ where U, V index respectively the rows and columns of A, and the edges of G correspond to the specified entries of A, and let $\overline{G}$ denote the complementary bipartite graph whose edges correspond to unspecified entries of A. Note that computing $\mathrm{MR}(A)$ amounts to computing the generic rank of A when viewing the unspecified entries of A as independent variables over the field containing the specified entries. For a subset $X \subseteq U \cup V$, let A_X denote the submatrix of A with respective row and column index sets $\{i \in [1, n]: u_i \notin X\}$ and $\{j \in [1, m]: v_j \notin X\}$. Call X a *cover* of $\overline{G}$ if every edge of $\overline{G}$ has at least one end node in X; that is, if A_X is a fully specified submatrix of A. Clearly, we have: $\mathrm{MR}(A) \leq \mathrm{rank}(A_X) + |X|$. In fact, the following equality holds:

$$\mathrm{MR}(A) = \min_{X \text{ cover of } \overline{G}} \mathrm{rank}(A_X) + |X| \qquad (9)$$

as shown in [12]. A determinantal version of the result was given in [25]. In the special case when all specified entries of A are equal to 0, then $\mathrm{MR}(A)$ coincides with the maximum cardinality of a matching in $\overline{G}$ and, therefore, the minimax relation (9) reduces to the Frobenius–König theorem (cf. [50] for details on the latter result). Moreover, one can determine $\mathrm{MR}(A)$ and construct a maximum rank completion of A in polynomial time. This was shown in [55] by a reduction to matroid intersection and, more recently, in [18] where a simple greedy procedure is presented that solves the problem by perturbing an arbitrary completion.

We now consider *minimum rank completions*. To start with, note that $\mathrm{mr}(A)$ may depend, in general, on the actual values of the specified entries of A (and not only on the ranks of the specified submatrices of A). Indeed, consider the partial matrix $A = \begin{pmatrix} ? & a & b \\ d & ? & c \\ e & f & ? \end{pmatrix}$ where a, $b, c, d, e, f \neq 0$. Then, $\mathrm{mr}(A) = 1$ if $ace = bdf$ and $\mathrm{mr}(A) = 2$ otherwise, while all specified submatrices have rank 1 in both cases. Thus arises the question of identifying the bipartite graphs G for which $\mathrm{mr}(A)$ depends only on the ranks of the specified submatrices of A for every partial matrix A with pattern G; such graphs are called *rank determined*. The graph pattern of the above instance A is the circuit C_6. Hence, C_6 is not rank determined. Call a bipartite graph G *bipartite chordal* if it does not contain a circuit of length ≥ 6 as an induced subgraph. Then, if a bipartite graph is rank determined, it is necessarily bipartite chordal [12]. It is conjectured there that, conversely, every bipartite chordal graph is rank determined. The conjecture was shown to be true in [66] for the nonseparable bipartite graphs (i. e., the bipartite graphs containing no induced matching of size 2; they are obviously bipartite chordal). Note that a partial matrix A has a nonseparable pattern if and only if it has (up to row/column permutation) the following 'triangular' form:



Then, $\mathrm{mr}(A)$ can be explicitly formulated in terms of the ranks of the specified submatrices of A; in the simplest case, the formula for $\mathrm{mr}(A)$ reads:

$$\mathrm{mr}\begin{pmatrix} B & ? \\ C & D \end{pmatrix} = \mathrm{rank}\begin{pmatrix} B \\ C \end{pmatrix} + \mathrm{rank}\begin{pmatrix} C & D \end{pmatrix} - \mathrm{rank}(C).$$

It is shown in [12] that the above conjecture holds when the pattern G is a path, or when G is obtained by 'gluing' a collection of circuits of length 4 along a common edge.

See also

- ► Interior Point Methods for Semidefinite Programming
- ► Semidefinite Programming and Determinant Maximization

References

1. Agler J, Helton JW, McCullough S, Rodman L (1988) Positive semidefinite matrices with a given sparsity pattern. Linear Alg & Its Appl 107:101–149
2. Alfakih AY, Khandani A, Wolkowicz H (1998) Solving Euclidean distance matrix completion problems via semidefinite programming. Comput Optim Appl 12:13–30
3. Alizadeh F (1995) Interior point methods in semidefinite programming with applications in combinatorial optimization. SIAM J Optim 5:13–51
4. Bakonyi M, Johnson CR (1995) The Euclidian distance matrix completion problem. SIAM J Matrix Anal Appl 16:646–654
5. Barahona F, Mahjoub AR (1986) On the cut polytope. Math Program 36:157–173
6. Barrett WW, Johnson CR, Loewy R (1996) The real positive definite completion problem: cycle completability. Memoirs Amer Math Soc, vol 584. Amer. Math. Soc., Providence, RI

7. Barrett WW, Johnson CR, Lundquist M (1989) Determinantal formulas for matrix completions associated with chordal graphs. Linear Alg & Its Appl 121:265–289
8. Barrett W, Johnson CR, Tarazaga P (1993) The real positive definite completion problem for a simple cycle. Linear Alg & Its Appl 192:3–31
9. Barvinok AI (1993) Feasibility testing for systems of real quadratic equations. Discrete Comput Geom 10:1–13
10. Barvinok AI (1995) Problems of distance geometry and convex properties of quadratic maps. Discrete Comput Geom 13:189–202
11. Blumenthal LM (1953) Theory and applications of distance geometry. Oxford Univ. Press, Oxford
12. Cohen N, Johnson CR, Rodman L, Woederman HJ (1989) Ranks of completions of partial matrices. In: Dym H, et al (eds) The Gohberg Anniv. Coll. vol I, Birkhäuser, Basel, pp 165–185
13. Crippen GM, Havel TF (1988) Distance geometry and molecular conformation. Res. Studies Press, Taunton, UK
14. Deza M, Laurent M (1997) Geometry of cuts and metrics. Algorithms and Combinatorics, vol 15. Springer, Berlin
15. Drew JH, Johnson CR (1998) The completely positive and doubly nonnegative completion problems. Linear Alg & Its Appl 44:85–92
16. Dym H, Gohberg I (1981) Extensions of band matrices with band inverses. Linear Alg & Its Appl 36:1–24
17. Ellis RL, Lay DC, Gohberg I (1988) On negative eigenvalues of selfadjoint extensions of band matrices. Linear Alg & Its Appl 24:15–25
18. Geelen J (1999) Maximum rank matrix completion. Linear Alg & Its Appl 288:211–217
19. Glunt W, Hayden TL, Raydan M (1998) Molecular conformations from distance matrices. J Comput Chem 14:175–190
20. Goemans MX (1997) Semidefinite programming in combinatorial optimization. Math Program 79:143–161
21. Golumbic MC (1980) Algorithmic theory and perfect graphs. Acad. Press, New York
22. Gray LJ, Wilson DG (1980) Nonnegative factorization of positive semidefinite nonnegative matrices. Linear Alg & Its Appl 31:119–127
23. Grone R, Johnson CR, Sá EM, Wolkowicz H (1984) Positive definite completions of partial hermitian matrices. Linear Alg & Its Appl 58:109–124
24. Grötschel M, Lovász L, Schrijver A (1988) Geometric algorithms and combinatorial optimization. Springer, Berlin
25. Hartfiel DJ, Loewy R (1984) A determinantal version of the Frobenius-König theorem. Linear Multilinear Algebra 16:155–165
26. Havel TF (1991) An evaluation of computational strategies for use in the determination of protein structure from distance constraints obtained by nuclear magnetic resonance. Program Biophys Biophys Chem 56:43–78
27. Helmberg C, Rendl F, Vanderbei RJ, Wolkowicz H (1996) An interior-point method for semidefinite programming. SIAM J Optim 6:342–361
28. Helton JW, Pierce S, Rodman L (1989) The ranks of extremal positive semidefinite matrices with given sparsity pattern. SIAM J Matrix Anal Appl 10:407–423
29. Hendrickson B (1990) The molecule problem: Determining conformation from pairwise distances. PhD Thesis. Techn. Report Dept. Computer Sci. Cornell Univ. 90–1159
30. Hendrickson B (1992) Conditions for unique graph realizations. SIAM J Comput 21:65–84
31. Hendrickson B (1995) The molecule problem: exploiting structure in global optimization. SIAM J Optim 5:835–857
32. Johnson CR (1990) Matrix completion problems: A survey. In: Johnson CR (ed) Matrix Theory and Appl. Proc Symp Appl Math. Amer. Math. Soc., Providence, RI, pp 171–198
33. Johnson CR, Jones C, Kroschel B (1995) The distance matrix completion problem: cycle completability. Linear Multilinear Algebra 39:195–207
34. Johnson CR, Kroschel B, Wolkowicz H (1998) An interior-point method for approximate positive semidefinite completions. Comput Optim Appl 9:175–190
35. Johnson CR, Rodman L (1984) Inertia possibilities for completions of partial Hermitian matrices. Linear multilinear algebra 16:179–195
36. Johnson CR, Rodman L (1986) Completion of matrices to contractions. J Funct Anal 69:260–267
37. Johnson CR, Rodman L (1988) Chordal inheritance principles and positive definite completions of partial matrices over function rings. In: Gohberg I (ed) Contributions to Operator Theory and its Applications. Birkhäuser, Basel, pp 107–127
38. Johnson CR, Tarazaga P (1995) Connections between the real positive semidefinite and distance matrix completion problems. Linear Alg & Its Appl 223(4):375–391
39. Khachiyan L, Porkolab L (1997) Computing integral points in convex semi-algebraic sets. 38th Annual Symp. Foundations Computer Sci., pp 162–171
40. Kogan N, Berman A (1993) Characterization of completely positive graphs. Discret Math 114:297–304
41. Kuntz ID, Thomason JF, Oshiro CM (1993) Distance geometry. Methods in Enzymologie 177:159–204
42. Laman G (1970) On graphs and rigidity of plane skeletal structures. J Eng Math 4:331–340
43. Laurent M (1997) Cuts, matrix completions and graph rigidity. Math Program 79:255–283
44. Laurent M (1997) The real positive semidefinite completion problem for series-parallel graphs. Linear Alg & Its Appl 252:347–366
45. Laurent M (1998) A connection between positive semidefinite and Euclidean distance matrix completion problems. Linear Alg & Its Appl 273:9–22
46. Laurent M (1998) On the order of a graph and its deficiency in chordality. CWI Report PNA-R9801
47. Laurent M (1998) A tour d'horizon on positive semidefinite and Euclidean distance matrix completion problems.

In: Pardalos PM, Wolkowicz H (eds) Topics in Semidefinite and Interior-Point Methods. Fields Inst Res Math Sci Commun. Amer. Math. Soc., Providence, RI, pp 51–76
48. Laurent M (2000) Polynomial instances of the positive semidefinite and Euclidean distance matrix completion problems. SIAM J Matrix Anal Appl 22(3)(874–894
49. de Leeuw J, Heiser W (1982) Theory of multidimensional scaling. In: Krishnaiah PR, Kanal LN (eds) Handbook Statist., vol 2, North-Holland, Amsterdam, pp 285–316
50. Lovász L, Plummer MD (1986) Matching theory. Akad. Kiadó, Budapest
51. Lovász L, Yemini Y (1982) On generic rigidity in the plane. SIAM J Alg Discrete Meth 3:91–98
52. Lundquist ME, Johnson CR (1991) Linearly constrained positive definite completions. Linear Alg & Its Appl 150:195–207
53. Moré JJ, Wu Z (1996) ϵ-optimal solutions to distance geometry problems via global continuation. In: Pardalos PM, Shalloway D (eds) DIMACS, vol 23, Amer. Math. Soc., Providence, RI, 151–168
54. Moré JJ, Wu Z (1997) Global continuation for distance geometry problems. SIAM J Optim 7:814–836
55. Murota K (1993) Mixed matrices: Irreducibility and decomposition. In: Brualdi RA et al (eds) Combinatorial and graph-theoretical problems in linear algebra. IMA. Springer, Berlin, pp 39–71
56. Nesterov YE, Nemirovsky AS (1994) Interior point polynomial algorithms in convex programming: Theory and algorithms. SIAM, Philadelphia
57. Pardalos PM, Lin X (1997) A tabu based pattern search method for the distance geometry problem. In: Giannessis F et al (eds) Math. Program. Kluwer, Dordrecht
58. Paulsen VI, Power SC, Smith RR (1989) Schur products and matrix completions. J Funct Anal 85:151–178
59. Porkolab L, Khachiyan L (1997) On the complexity of semidefinite programs. J Global Optim 10:351–365
60. Ramana MV (1997) An exact duality theory for semidefinite programming and its complexity implications. Math Program 77:129–162
61. Rose DJ (1970) Triangulated graphs and the elimination process. J Math Anal Appl 32:597–609
62. Saxe JB (1979) Embeddability of weighted graphs in k-space is strongly NP-hard. In: Proc. 17th Allerton Conf. Communications, Control and Computing, pp 480–489
63. Schoenberg IJ (1935) Remarks to M. Fréchet's article 'Sur la définition axiomatique d'une classe d'espaces vectoriels distanciés applicables vectoriellement sur l'espace de Hilbert', Ann of Math 36:724–732
64. WEB: http://orion.math.uwaterloo.ca:80/~hwolkowi/henry/software/readme.html
65. WEB: http://www.zib.de/helmberg/semidef.html
66. Woerdeman HJ (1987) The lower order of lower triangular operators and minimal rank extensions. Integral Eq Operator Theory 10:859–879
67. Yemini Y (1979) Some theoretical aspects of position-location problems. Proc. 20th Annual Symp. Foundations Computer Sci., pp 1–8

Matroids

PAOLA FESTA
Dip. Mat. e Inform., Universita Salerno,
Baronissi, Italy

MSC2000: 90C09, 90C10

Article Outline

Keywords

Combinatorial optimization; Greedy technique; Graph optimization

Matroids have been defined in 1935 as generalization of graphs and matrices. Starting from the 1950s they have had increasing interest and the theoretical results obtained have been used for solving several difficult problems in various fields such as civil, electrical, and mechanical engineering, computer science, and mathematics. A comprehensive treatment of matroids can not be contained in few pages or even in only one book. Thus, the scope of this article is to introduce the reader to this theory, providing the definitions of some different types of matroids and their main properties.

Historical Overview

In 1935, H. Whitney in [38] studied *linear dependence* and its important application in mathematics. A number of equivalent axiomatic systems for matroids is contained in his pioneering paper, that is considered the first scientific work about matroid theory.

In the 1950s and 1960s, starting from the Whitney's ideas, W. Tutte in [25,26,27,28,29,30,31,32,33] built a considerable body of theory about the structural properties of matroids, which became popular in the 1960s, when J. Edmonds in [5,6,7,8,9,10,11] introduced matroid theory in combinatorial optimization. From 1965 on, a growing number of researchers became interested in matroids. In 1976, D.J.A. Welsh ([34]) published the first book on matroid theory. In the 1970s, 1980s, and 1990s selected topics have been covered by a huge number of scientific publications, among them [1,2,3,12,13,15,17,18,20,21,23,24,35,36,37]. [16] provides an excellent historical survey, while [21] is a good book for students.

Definition of a Matroid

Matroids are combinatorial structures often treated in together with the *greedy technique*, which yields optimal solutions when applied for solving simple problems defined on matroids.

In order to provide the definition of a general matroid, some notation and further definitions are needed.

Definition 1 An ordered pair $S = (E, I)$, where $E = \{e_1, \dots, e_n\}$ and $I \subseteq 2^E$, is an *independent system* (SI) if and only if

$$\forall A, B \subseteq E: \; B \subset A \in I \Rightarrow B \in I. \tag{1}$$

E is also called *ground set*.

Note that the empty set is necessarily a member of I.

Definition 2 The members of I are called *independent sets*.

Definition 3 The members of $D = 2^E \setminus I$ are called *dependent sets*.

Definition 4 The members of the set

$$B = \{A \subseteq E\colon \; A \in I, \; \forall f \in E \setminus A : \; B \cup \{f\} \notin I\}$$

are called *maximal independent sets* or *bases*.

In other words, a *basis* is an independent set which is maximal with respect to set inclusion operation.

Definition 5 The members of the set

$$C = \{C \subseteq E\colon \; C \in D, \; \forall f \in C : \; C \setminus \{f\} \in I\}$$

are called *minimal dependent sets* or *circuits*. A 1-element circuit is a *loop*.

Definition 6 A *matroid* M is an independent system (E, I) such that if $A, B \in I$, $|A| < |B|$, then there is some element $x \in B \setminus A$ such that $A \cup \{x\} \in I$.

We say that M satisfies the *exchange property*.

Most combinatorial problems can be viewed as the problem of finding an element in one of the above defined sets corresponding to the optimal objective function value.

The word *matroid* is due to Whitney. He studied *matric matroids*, in which the elements of E are the rows of a given matrix and a set of rows is independent if they are linearly independent in the usual sense.

The following theorems express two equivalent axiomatic definitions of matroids in terms of bases and circuits.

Theorem 7 *A nonempty set B of subsets of E is the set of bases for a matroid $M = (E, I)$ if and only if for all B_1, $B_2 \in B$, $B_1 \neq B_2$, and $x \in B_1 \setminus B_2$, there exists an element $y \in B_2 \setminus B_1$ such that*

$$B_1 \cup \{y\} \setminus \{x\} \in B.$$

Theorem 8 *A set C of subsets of E is the set of circuits for a matroid $M = (E, I)$ if and only if the following two properties hold:*
1) *for all $X \neq Y \in C$, $X \nsubseteq Y$;*
2) *for all $X \neq Y \in C$ and $z \in X \cap Y$, there exists $Z \in C$ such that $Z \subseteq X \cup Y \setminus \{z\}$.*

Other alternative axiomatic characterizations of a matroid need some further definitions.

Let $M = (E, I)$ be a matroid.

Definition 9 For all $A \subseteq E$, let ρ: 2^E **N** be a function such that

$$\rho(A) = \max\{|X| : \; X \subseteq A, X \in I\}.$$

ρ is called *rank* of M.

Note that the rank of M is equal to the rank of E, which is given by the cardinality of the maximal independent subset of E. The rank is always well-defined, due to the following proposition.

Proposition 10 *If A is a subset of E and X and Y are maximal independent subsets of A, then $|X| = |Y|$.*

Proposition 10 claims that the maximal independent subsets contained in $A \subseteq E$ of a given matroid $M = (E, I)$ have the same cardinality. Choosing $A = E$, the following corollary holds.

Corollary 11 *The bases of any matroid have the same cardinality.*

Definition 12 A subset A of E is called a *closed* of M if

$$\rho(A \cup \{x\}) = \rho(A) + 1, \quad \forall x \in E \setminus A,$$

i. e. if it is not possible to add to A any element without increasing its rank.

Definition 13 The *closure operator* for M is a function $\sigma\colon 2^E \to 2^E$ such that for all $A \subseteq E$ $\sigma(A)$ is the closed of minimum cardinality that contains A, i. e.

$$\sigma(A) = A \cup \{x \in E \setminus A\colon\ \rho(A \cup \{x\}) = \rho(A)\}\,.$$

Definition 14 A subset A of E *covers* M if and only if it contains a basis of M, i. e.

$$\rho(A) = \rho(E).$$

With these further definitions at hand, the following theorems express three other equivalent axiomatic characterizations of a matroid in terms of its rank.

Theorem 15 *A function $\rho\colon 2^E \to \mathbf{N}$ is a rank function of a matroid $M = (E, I)$ if and only if for all $X \subseteq E$ and for all $y, z \in E$ the following three properties hold:*

1) $\rho(\emptyset) = 0$;
2) $\rho(X) \le \rho(X \cup \{y\}) \le \rho(X) + 1$;
3) $\rho(X) = \rho(X \cup \{y\}) = \rho(X \cup \{z\}) \Rightarrow \rho(X \cup \{y, z\}) = \rho(X)$.

Theorem 16 *A function $\rho\colon 2^E \to \mathbf{N}$ is a rank function of a matroid $M = (E, I)$ if and only if for all $X \neq Y \subseteq E$ the following three properties hold:*

1) $0 \le \rho(X) \le |X|$;
2) $X \subseteq Y \Rightarrow \rho(X) \le \rho(Y)$;
3) $\rho(X \cup Y) + \rho(X \cap Y) \le \rho(X) + \rho(Y)$.

Note that the second property of theorem 16 implies that ρ is a monotonic function, while the third property expresses its submodularity.

Theorem 17 *A function $\sigma\colon 2^E \to 2^E$ is a closure operator of a matroid $M = (E, I)$ if and only if for all $X \neq Y \subseteq E$ and for all $x, y \in E$ the following four properties hold:*

1) $X \subseteq \sigma(X)$;
2) $Y \subseteq X \Rightarrow \sigma(Y) \subseteq \sigma(X)$;
3) $\sigma(X) = \sigma(\sigma(X))$;
4) $y \notin \sigma(X),\ y \in \sigma(X \cup \{x\}) \Rightarrow x \in \sigma(X \cup \{y\})$.

Definition 18 A matroid $M = (E, I)$ is *weighted* if there is an associated weight function w that assigns a strictly positive weight $w(x)$ to each element $x \in E$.

The *weight function* w extends to subsets A of E by summation:

$$w(A) = \sum_{x \in A} w(x).$$

Minor of Matroids: Restriction and Contraction

A *minor* of a matroid $M = (E, I)$ is a 'submatroid' obtained from *deleting* or *contracting* from the ground set E one or more elements.

A *loop* is an element y of a matroid such that $\{y\}$ is not independent. Equivalently, $\{y\}$ does not lie in any independent set, nor in maximal independent sets.

Definition 19 Let $M = (E, I)$ be a matroid. If an element $\{x\}$ is not a loop, the matroid M/x, called a *contraction* of M, is defined as follows:

1) the ground set of M/x is $E \setminus \{x\}$;
2) a set A is independent in M/x if and only if $A \cup \{x\}$ is independent in M.

The concept of matroid contraction can be dualized. In fact, an element y is a *coloop* if it is contained in every basis of M.

Definition 20 Let $M = (E, I)$ be a matroid. If an element $\{x\}$ is not a coloop, the matroid $M \setminus x$, called a *restriction* of M, is defined as follows:

1) the ground set of $M\setminus x$ is $E \setminus \{x\}$;
2) a set A is independent in $M\setminus x$ if and only if it is independent in M.

The above definitions have been given in terms of restriction and contraction of only one element, but they can be easily extended to the restriction and contraction of a set X. The minors obtained will be denoted $M\backslash X$ and $M\backslash X$, respectively.

Representability of Matroids

One among the most common canonical examples of matroids is the *vectorial matroid*, whose ground set E is a finite set of vectors from a vector space, while the independent sets are the linearly independent subsets of vectors of E. A matroid $M = (E, I)$ is *representable* on a field F if there exists some vector space V over F, with some finite set E of vectors of V, so that M is isomorphic to the vectorial matroid of the set E. A *binary matroid* is a matroid representable over GF(2), while a *ternary matroid* is representable over GF(3).

In recent literature (as of 1999) the problem of classifying all the fields over which a given matroid is representable and the inverse problem of characterizing all the matroids that are representable on a given field have had growing interest. An important result for matroid representability is the following theorem.

Theorem 21 *A matroid $M = (E, I)$ is representable over any field if and only if it is representable over GF(2) and over some field of characteristic other than two.*

A matroid as in the previous theorem is called *regular*.

Connectivity of Matroids

Connectivity is an important concept in matroid theory.

Definition 22 A matroid $M = (E, I)$ admits a *k-separation* if there exists a partition (X, Y) of the ground set E such that

1) $|X| \geq k, |Y| \geq k$;
2) $\rho(X)+ \rho(Y) - \rho(E) \leq k - 1$.

Definition 23 The smallest k such that a matroid $M = (E, I)$ admits a k-separation is called the *connectivity* of M.

If $k \geq 2$, M is n-connected for any $n \leq k$; if $k = 1$, M is *disconnected*; if M admits any k-separations for all integers k, M has *infinite connectivity*.

An important result for matroid connectivity is the following theorem.

Theorem 24 *A matroid $M = (E, I)$ is disconnected if and only if there exists a partition (X, Y) of the ground set E such that every circuit C of M is either a subset of X or a subset of Y.*

Examples of Matroids

In this section some of the most popular types of matroids involved in combinatorial optimization will be described.

Uniform Matroid

Let E be a set of n elements and let I be the family of subsets A of E such that $|A| \leq k < n$. Then $M = (E, I)$ is called the *uniform matroid* of rank k and is denoted by $U_{k,n}$.

The sets of the bases and the circuits of $U_{k,n}$ are

$$B = \{X \subseteq E\colon\ |X| = k\}$$

and

$$C = \{X \subseteq E\colon\ |X| = k + 1\}\,,$$

respectively.

Moreover, for all $A \subseteq E$,

$$\rho(A) = \begin{cases} |A| & \text{if } |A| \leq K, \\ K & \text{otherwise,} \end{cases}$$

$$\sigma(A) = \begin{cases} A & \text{if } |A| \leq K, \\ E & \text{otherwise.} \end{cases}$$

Graphic Matroid

If F is the set of forests of a graph $G = (V, E)$, $M = (E, F)$ is called a *graphic matroid*. The circuits of M are the graph-theoretic circuits of G, while the rank of a subset E_1 of E is given by

$$\rho(E_1) = |V| - c(E_1),$$

where $c(E_1)$ is the number of connected components of $G_1 = (V, E_1)$.

Transversal Matroid

Let E be a finite set, $C = \{S_1, \dots, S_m\}$ a collection of subsets of E, and let $T = \{e_1, \dots, e_t\} \subseteq E$.

T is called a *transversal* of C if there exist distinct integers $j(1), \dots, j(t)$ such that $e_i \in S_{j(i)}$, $i = 1, \dots, t$. Let I be the set of all transversals of E, then $M = (E, I)$ is a *transversal matroid*.

Partition Matroid

Let E be a finite set, $\Pi = \{E_1, \ldots, E_p\}$ a *partition* of E, that is a collection of disjoint subsets of E covering E, and $d_1, \ldots, d_p$ p nonnegative integers. A subset A of E is *independent*, i. e. $A \in I$, if and only if $|A \cap E_j| \leq d_j$, $j = 1, \ldots, p$. The system $M = (E, I)$ is a matroid, called a *partition matroid*.

An example of a partition matroid can be obtained by considering any digraph $G = (V, E)$ and partitioning the edges of the set E according to which node is the head (or, equivalently, the tail) of each. Suppose that $d_j = 1$, $j = 1, \ldots, p$; then a set A of edges is independent if no two edges of A have the same head (or, equivalently, the same tail).

Dual Matroids

Let $M = (E, I)$ be a matroid, and let B be its set of bases.

The *dual matroid* $\overline{M}$ is the matroid on the ground set E, whose bases are the complements of the bases of M. Thus, a set A is independent in $\overline{M}$ if and only if A is disjoint from some basis of M. Note that $\overline{\overline{M}} = M$.

For a pair of matroids $(M, \overline{M})$ and their rank functions, the following propositions hold.

Proposition 25 *Let $M = (E, I)$ be a matroid, and let ρ be its rank function. Let $\overline{M} = (E, \overline{I})$ be the dual matroid of M; then*

$$\overline{\rho}(A) = |A| + \rho(E \setminus A) - \rho(E),$$

for each $A \subseteq E$.

Proposition 26 *Let $\overline{M}$ be the dual of the matroid $M = (E, I)$, let A be a subset of E and let $\overline{A} = E \setminus A$. If ρ and $\overline{\rho}$ are the rank functions of M and $\overline{M}$ respectively, then*
1) $\left|\overline{A}\right| - \overline{\rho}(\overline{A}) = \rho(E) - \rho(A)$;
2) $\overline{\rho}(\overline{E}) - \overline{\rho}(\overline{A}) = |A| - \rho(A)$.

Proposition 27 *Let $M = (E, I)$ be a matroid, then*
1) *x is a loop in M if and only if x is a coloop in $\overline{M}$ and vice versa;*
2) *If x is not a loop in M, then the dual of M/x is the matroid $\overline{M} \backslash x$;*
3) *If x is not a coloop in M, then the dual of $M \setminus x$ is the matroid $\frac{\overline{M}}{x}$.*

As example of the dual of a matroid, let us consider the vectorial matroid. Suppose that the vectors representing M are the columns of an $m \times n$ matrix A and that these vectors span F^m. Thus, A has rank m and is the matrix of a linear transformation T from F^n onto F^m. Let K be the kernel of T, and B the matrix of a linear embedding of U into F^n. Note that B is a $n \times (n - m)$ matrix (whose columns are the basis for U) and has rank $n - m$. Moreover, the columns of the $(n - m) \times n$ matrix B^{T} are indexed by the same set as the columns of A and $B^{\mathsf{T}} A = 0$. B^{T} is the dual matroid $\overline{M}$ of the vectorial matroid M.

Greedy Algorithms on Weighted Matroids

Many combinatorial problems for which the greedy technique gives an optimal solution can be formulated in terms of finding a maximum-weight independent subset in a weighted matroid. In more detail, there is given a weighted matroid $M = (E, I)$ and the objective is to find an independent set $A \in I$ such that $w(A)$ is maximized (also called an *optimal subset* of M). Since the weight $w(x)$ of any element $x \in E$ is positive, a maximum-weight independent subset is always a maximal independent subset.

In the *minimum spanning tree problem*, for example, there are given a connected undirected graph $G = (V, E)$ and a length function w such that $w(e)$ is the positive length of the edge e. The objective is to find an acyclic subset T of E that connects all of the vertices of G and whose total length

$$w(T) = \sum_{e \in T} w(e)$$

is minimized. This is a classical combinatorial problem and can be formulated as a problem of finding an optimal subset of a matroid. In fact, consider the graphic weighted matroid M_G with weight function w' such that $w'(e) = w_0 - w(e)$, where w_0 is larger than the maximum length of any edge. It can be easily seen that for each $e \in E$, $w'(e) \geq 0$ and that an optimal subset of M_G is a spanning tree of minimum total length in the original graph G. In more detail, each maximal independent subset A corresponds to a spanning tree and since

$$w'(A) = (|V| - 1) \cdot w_0 - w(A)$$

for any maximal independent subset A, the independent subset that maximizes $w'(A)$ must minimize $w(A)$.

J.B. Kruskal in [14] and R.C. Prim in [22] proposed two greedy strategies for solving efficiently the minimum spanning tree, but in the following is reported the pseudocode of a greedy algorithm that works for any weighted matroid. The algorithm GREEDY takes as input a matroid $M = (E, I)$ and a weight function w and returns an optimal subset A.

```
set $A = \emptyset$
sort $E[M] = \{x_1, \ldots, x_k\}$ into nonincreasing order
by weight $w$
FOR $i = 1$ to $t$
    IF $A \cup \{x_i\} \in I[M]$
        set $A = A \cup \{x_i\}$
return(A)
```

Greedy(*M*,*w*)

Like any other greedy algorithm, GREEDY always makes the choice that looks best at the moment. In fact, it considers in turn each element x_i belonging to $E[M]$, whose element are sorted into nonincreasing order by weight w and immediately adds x to the building set A if $A \cup \{x_i\}$ is still independent. Note that the returned set A is always independent, because it is initialized to the empty set, which is independent by definition of a matroid, and then at each iteration an element x_i is added to A while preserving the A's independence. A is also an optimal subset of the matroid M and therefore, a minimum spanning tree for the original graph G. To prove its optimality, it is enough to show that weighted matroids exhibit the two ingredients whose existence guarantee that a greedy strategy will solve optimally the given problem: the *greedy-choice property* and the *optimal substructure property*. The proof that matroids exhibit both these properties can be found in [4]. Generally speaking, the proof of the exhibition of the greedy-choice property consists of showing that a globally optimal solution can be obtained by making a locally optimal (greedy) choice. The proof examines a global optimal solution. It shows that the solution can be modified so that a greedy choice is made at the first step and that this choice reduces the original problem into an equivalent problem having smaller size. By induction, it is proved that a greedy choice can be made at each step. To show that making a greedy choice reduces the original problem into a similar but smaller problem reduces the proof of correctness to demonstrating that an optimal solution must exhibit optimal substructure. The optimal substructure property is exhibited by a given problem, if an optimal solution to the problem contains within it optimal solutions to subproblems. The validity of this property guarantee the applicability of greedy strategies as well as dynamic programming algorithms.

See also

► Oriented Matroids

References

1. Aigner M (1979) Combinatorial theory. Springer, Berlin
2. Bachem A, Kern W (1992) Linear programming duality: An introduction to oriented matroids. Springer, Berlin
3. Björner A, Las Vergnas M, Sturmfels B, White N, Ziegler GM (1993) Oriented matroids. Encycl Math Appl, vol 46. Cambridge Univ. Press, Cambridge
4. Cormen TH, Leiserson CE, Rivest RL (1990) Introduction to algorithms. MIT, Cambridge, MA
5. Edmonds J (1965) Lehman's switching game and a theorem of Tutte and Nash-Williams. J Res Nat Bureau Standards (B) 69:73–77
6. Edmonds J (1965) Maximum matching and a polyhedron with {0, 1} vertices. J Res Nat Bureau Standards (B) 69:125–130
7. Edmonds J (1965) Minimum partition of a matroid into independent subsets. J Res Nat Bureau Standards (B) 69:67–72
8. Edmonds J (1965) Paths, trees, and flowers. Canad J Math 17:449–467
9. Edmonds J (1967) Optimum branchings. J Res Nat Bureau Standards (B) 71:233–240
10. Edmonds J (1967) Systems of distinct representatives and linear algebra. J Res Nat Bureau Standards (B) 71:241–245
11. Edmonds J (1970) Submodular functions, matroids, and certain polyhedra. In: Guy R, Hanani H, Sauer N, Schönheim J (eds) Combinatorial Structures and Their Applications. Gordon and Breach, New York
12. Fujishige S (1991) Submodular functions and optimization. Ann Discret Math, vol 47. North-Holland, Amsterdam
13. Crapo HH, Rota GC (1970) On the foundations of combinatorial theory: Combinatorial geometries. preliminary MIT, Cambridge, MA
14. Kruskal JB (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. Proc Amer Math Soc 7:48–50
15. Kung JPS (1986) Numerically regular hereditary classes of combinatorial geometries. Geometriae Dedicata 21:85–105
16. Kung JPS (1986) A source book in matroid theory. Birkhäuser, Basel
17. Lawler EL (1976) Combinatorial optimization: Networks and matroids. Holt, Rinehart and Winston, New York

18. Lovász L, Plummer MD (1986) Matching theory. Akad. Kiadó, Budapest
19. Maffioli F (1990) Elementi di programmazione matematica. Masson, Paris
20. Murota K, Iri M, Nakamura M (1987) Combinatorial canonical form of layered mixed matrices and its application to block-triangularization of systems of linear/nonlinear equations. SIAM J Alg Discrete Meth 8:123–149
21. Oxley JG (1992) Matroid theory. Oxford Univ. Press, Oxford
22. Prim RC (1957) Shortest connection networks and some generalizations. Bell System Techn J 36:1389–1401
23. Recski A (1989) Matroid theory and its application in electrical networks and statics. Springer, Berlin
24. Schrijver A (1986) Theory of linear and integer programming. Wiley, New York
25. Tutte WT (1958) A homotopy theorem for matroids I–II. Trans Amer Math Soc 88:144–160; 161–174
26. Tutte WT (1959) Matroids and graphs. Trans Amer Math Soc 90:527–552
27. Tutte WT (1960) An algorithm for determining wheter a given binary matroid is graphic. Proc Amer Math Soc 11:905–917
28. Tutte WT (1961) On the problem of decomposing a graph into n connected factors. J London Math Soc 36:221–230
29. Tutte WT (1964) From matrices to graphs. Canad J Math 16:108–127
30. Tutte WT (1965) Lectures on matroids. J Res Nat Bureau Standards (B) 69:1–47
31. Tutte WT (1966) Connectivity in graphs. Univ. Toronto Press, Toronto
32. Tutte WT (1966) Connectivity in matroids. Canad J Math 18:1301–1324
33. Tutte WT (1971) Introduction to the theory of matroids. Amer. Elsevier, New York
34. Welsh DJA (1976) Matroid theory. Acad. Press, New York
35. White N (1986) Theory of matroids. Cambridge Univ. Press, Cambridge
36. White N (1987) Combinatorial geometries. Cambridge Univ. Press, Cambridge
37. White N (1991) Matroid applications. Cambridge Univ. Press, Cambridge
38. Whitney H (1935) On the abstract properties of linear dependence. Amer J Math 57:509–533

Maximum Constraint Satisfaction: Relaxations and Upper Bounds

Michel Minoux[1], P. Mavrocordatos[2]
[1] Université Paris 6, Paris, France
[2] Algotheque and Université Paris 6, Paris, France

MSC2000: 90C10

Article Outline

Keywords

Constraint satisfaction; Relaxation

Maximum constraint satisfaction problems (MAX-CSPs) generalize *maximum satisfiability* (MAX-SAT) to include cases where the variables are no longer restricted to binary (or Boolean) values.

MAX-CSP is *NP*-complete even in the special case of *binary CSPs*. Therefore designing procedures to compute upper bounds to the exact (unknown) optimum value (maximum number of satisfied constraints) is a relevant issue. Such bounds may be useful, in particular, to provide estimates of the quality of solutions obtained from various heuristic approaches.

This article describes a systematic way of computing upper bounds for large scale MAX-CSP instances such as those arising from the so-called *radio link frequency assignment problem* (RLFAP). After discussing the general relaxation principle and the basic procedure from which the bounds are derived, we present results of extensive computational experiments on series of 90 instances of RLFAP including both real test problems and randomly generated 'realistic' test problems (for sizes ranging from 396 variables and about 1700 constraints to 831 variables and about 4800 constraints).

These results clearly indicate that the proposed approach is practically useful to produce fairly accurate upper bounds for such large MAX-CSP problems.

Introduction

Constraint satisfaction problems (CSPs) may be viewed as a generalization of *satisfiability* (SAT) to include cases where, instead of taking binary values only (0–1 or true-false) the variables may take on a finite number (> 2) of given possible values.

For an infeasible CSP, a relevant question, both theoretically and practically, is to determine an assignment of values to variables such that the number of satisfied constraints is the *largest possible*. This is the so-called *maximum constraint satisfaction problem* (MAX-CSP), which generalizes in a natural way *maximum satisfiability* (MAX-SAT).

Since MAX-2SAT is *NP*-complete (see e. g. [12, pp. 259–260]) even the subclass of MAX-CSP corresponding to *binary CSPs* (those problems with constraints involving pairs of variables only) is *NP*-complete. Therefore, for very large instances such as those arising from practical applications (e. g. the RLFAP discussed below) one can only hope for approximate solutions using some of the currently available heuristic approaches such as: simulated annealing, tabu search, genetic algorithms, or local search of various kinds.

However, for many applications, getting an approximate solution without any information about the quality of this solution (e. g. measured by the difference between the cost of this solution and the optimal cost) may be of little value.

We address in this paper the problem of computing *upper bounds* to the optimum cost of MAX-CSP problems from which estimates on the quality of heuristic solutions can be derived.

The article is organized as follows. Basic definitions about CSPs and MAX-CSPs are recalled in the second section. Modeling the so-called radio link frequency assignment problem (RLFAP) in terms of CSP and MAX-CSP is addressed in the third section. Then we present a general class of relaxations for MAX-CSP problems and its specialization to the computation of MAX-CSP bounds for RLFAP. Finally results of extensive computational experiments carried out on series of both real test problems and realistic randomly generated test problems are presented. To our knowledge, this is the first time extensive computational results of this kind are reported for such large scale MAX-CSP problems.

CSP and MAX-CSP

A *constraint satisfaction problem* (CSP) is defined by specifying:

- a set of n variables $x_1, \dots, x_n$;
- for each variable x_i, $i \in I = \{1, \dots, n\}$ the domain of i, i. e. the (finite) set D_i of possible values for x_i;
- a set of K constraints φ_k, $k = 1, \dots, K$. For each $k \in [1, K]$, constraint φ_k is defined by its *support set* (i. e. the subset $S_k = \text{supp}(\varphi_k)$ of indices of the variables involved in the constraint) and an *oracle* which, given any combination $\overline{x}_{[S_k]}$ of values for variables in S_k, answers TRUE if $\varphi_k(\overline{x}_{[S_k]}) = \text{TRUE}$, i. e. if the combination is allowed, FALSE otherwise. (For any $S \subset \{1, \dots, n\}$ and $x \in D_1 \times \cdots \times D_n$, $x_{[S]}$ denotes the vector x restricted to components in S.)

Given a CSP specified as above, we define a *free assignment* as any n-tuple $x \in D = D_1 \times \cdots \times D_n$. A *feasible assignment* (or *solution*) is a free assignment such that $\varphi_k(x_{[S_k]}) = \text{TRUE}$ for all $k = 1, \dots, K$.

For simplicity, we restrict here to the case where each variable takes scalar values only (i. e. real or integer values), but we note that more general CSPs may be defined with variables taking, for instance, vector values.

The *arity of a constraint* φ_k is the cardinality of its support set: $|S_k| = |\text{supp}(\varphi_k)|$. A *binary CSP* is a constraint satisfaction problem in which $|\text{supp}(\varphi_k)| \leq 2$ for all $k = 1, \dots, K$.

The constraint hypergraph associated with a given CSP is the hypergraph having vertex set $I = \{1, \dots, n\}$ and edge set $\{S_1, \dots, S_K\}$. In case of a binary CSP this is a graph.

The two examples below are interesting special cases of the general definition and show *NP* completeness of arbitrary CSPs.

Example 1 (Satisfiability) SAT is easily recognized as a special case of CSP where $\forall i$: $D_i = \{\text{TRUE, FALSE}\}$ and where there is a constraint φ_k corresponding to each clause C_k with $\varphi_k(x) = \text{TRUE} \Leftrightarrow$ clause C_k is satisfied under truth assignment x.

Example 2 (Hypergraph q-coloring; see [2, Chap. 19]) Let $q > 1$ be a given integer and $H = [V, E]$ an hypergraph with vertex set V and edge set E. The problem is to assign one out of q colors to each vertex of H so that each edge of H has vertices of different colors. Clearly this may be formulated as a CSP problem where there is one variable x_i for each $v_i \in V$, with domain $D_i = \{1,$

$\dots, q\}$, and one constraint φ_k for each edge $e_k = \{i_1, \dots, i_p\} \in E$ such that $\varphi(\overline{x}_{i_1}, \dots, \overline{x}_{i_p}) = \text{TRUE} \Leftrightarrow$ no two values in $\{\overline{x}_{i_1}, \dots, \overline{x}_{i_p}\}$ are equal. Note that when H is a *graph* (i. e. $|e_k| = 2$ for all $e_k \in E$), the resulting CSP is a binary CSP.

For an infeasible CSP, one basic question is to determine a 'best possible' or 'least infeasible' assignment. If the criterion for quality (or degree of 'feasibility') of an assignment x is taken to be the number $\sigma(x)$ of constraints satisfied under that assignment, we are led to the so-called *MAX-CSP* problem:

- *Given*: a CSP defined by its variables $x_1, \dots, x_n$, domains $D_1, \dots, D_n$, and constraints $\varphi_1, \dots, \varphi_K$.
- *Find*: $x \in D_1 \times \dots \times D_n$ such that

$$\sigma(x) = \left|\left\{k \in [1, K] \colon \ \varphi_k(x_{[S_k]}) = \text{TRUE}\right\}\right|$$

is maximized.

Example 3 (MAX-SAT, MAX-2SAT) Clearly, MAX-SAT is a special case of MAX-CSP when the given CSP is a satisfiability problem. The associated decision problem is *NP*-complete even for the special case of MAX-2SAT ([13]), showing that MAX-CSP is *NP*-complete even for binary CSPs.

Heuristics for approximately solving the MAX-SAT problem have been proposed by [17,19,23,27]. A branch and bound algorithm for MAX-SAT based on probabilistic bounds is described in [3] with computational results up to 100 binary variables and 1000 clauses. The branch and cut algorithm described in [20] presents computational results for general Max-3SAT problems up to 100 binary variables and 575 clauses. For a recent survey on SAT and MAX-SAT, see [9].

For more general MAX-CSP problems, many heuristic approaches have been investigated such as *tabu search* ([4,7]), *simulated annealing* [5], *genetic algorithms* [18]. Exact Algorithms for random MAX-CSP problems were proposed in [11]. However in the computational experiments reported, the sizes of the problems for which exact optimal solutions were found are rather small (144 variables with domains of cardinality 4 and 646 constraints for the largest problems solved in [11]).

MAX-CSP and the Radio Link Frequency Assignment Problem

Operating large radio link telecommunication networks gives rise to the so-called *radio link frequency assignment problem* (RLFAP), which is to choose, for each transmission link, a specific operating frequency (among a given list of allowed values) while satisfying a list of noninterference constraints, (most constraints usually involving pairs of links). A CSP formulation of RLFAP is as follows: With n denoting the number of links, for each link $i = 1, \dots, n$, there is an associated variable x_i representing the frequency to be assigned to link i. The domain D_i of x_i is the (finite) set of allowed frequencies for link i (frequencies are expressed in Hz, KHz, MHz or any other specified unit).

Any assignment $x \in S = D_1 \times \dots \times D_n$ is not allowed because a number of constraints, called *noninterference constraints* have to be satisfied.

We will only consider here the case of *binary noninterference constraints* (i. e. involving only pairs of links), which is relevant to many applications of interest (see e. g. [15,16]). For a given pair of links i and j, two (exclusive) types of constraints are possible:

- equality constraints of the form

$$\text{(E)} \quad \left|x_i - x_j\right| = w_{ij};$$

- inequality constraints of the form

$$\text{(I)} \quad \left|x_i - x_j\right| \geq w_{ij}.$$

The real number w_{ij} which represents the requested slack or minimum requested slack between the two assigned frequencies will be called the *weight of the constraint*.

An instance of RLFAP is therefore specified by n (number of links), a list of domains $D_1, \dots, D_n$ and a list of constraints i. e. a list of quadruples of the form (i, j, w_{ij}, T_{ij}) where: i, j are the indices of the two links involved, w_{ij} is the weight of the constraint, and T_{ij} its type ((E) or (I)). The *constraint graph* associated with an instance of RLFAP is defined as the undirected graph G with node set $\{1, \dots, n\}$ and with an edge (i, j) for each constraint (i, j, w_{ij}, T_{ij}). We denote K the total number of constraints in an instance of RLFAP. Benchmarks of the RLFAP involving real instances up to 916 variables and 5744 constraints have been made publicly available in the context of the European Project CALMA (see [8,15]). In those practical instances, the number of equality constraints (of type (E)) is never more than $n/2$, and assignments satisfying all of them can easily be found. We will denote $S' \subset S = D_1 \times \cdots \times D_n$ the set of all such assignments. All assignments $x \in S \setminus S'$ must be disregarded because they are physically meaningless, therefore, from now on, we will only consider assignments in S' as possible solution to RLFAP.

An assignment in S' which satisfies all constraints of type (I) will be called *feasible*. The feasibility version of RLFAP may therefore be stated as the following CSP:

- *Given*: an instance of RLFAP.
- *Question*: does there exist a feasible frequency assignment?
- *Answer*: yes or no and, if yes, output a feasible assignment x.

Efficient solution methods for RLFAP are of major interest to numerous practical applications in the context of civilian mobile communication networks as well as of military networks. Since the available spectrum is severely limited and the communication needs (traffic requirements) are continuously increasing, a high proportion of the instances of the RLFAP encountered in applications turn out to be *infeasible*.

When faced with an instance which is either infeasible or which is presumably infeasible (e. g. because running a heuristic solution method just failed to produce a feasible solution) a key question for the practitioner becomes to determine a 'best possible' or 'least infeasible' assignment.

This leads to the 'optimization version' of the RLFAP in the form of the following *MAX-CSP*:

- *Given*: an instance of RLFAP with n variables (links) and K constraints.
- *Question*: determine $x^* \in S'$ such that $\sigma(x^*)$ (number of satisfied constraints) is maximized:

$$\sigma(x^*) = \max_{x \in S'} \{\sigma(x)\}.$$

In view of the *NP*-completeness of MAX-CSP for binary CSPs, guaranteed optimal solutions to the above for large scale instances (such as those of the CELAR benchmarks) cannot be reasonably expected from currently available techniques in combinatorial optimization. A less ambitious, though practically relevant objective, addressed in the following section, is to try and obtain good *upper bounds* to an optimal solution value.

We note here that in the case where an upper bound $\widehat{\sigma}$ is found such that $\widehat{\sigma} < K$, then we can deduce that the given RLFAP has *no feasible solution*. Thus, an interesting by-product of computing bounds will be to produce *proofs of infeasibility* of a given instance of RLFAP. Clearly, such an information may be of considerable importance to the practitioner.

A General Class of Relaxations for Computing MAX-CSP Bounds

MAX-CSP may be reformulated as the discrete optimization problem

$$\begin{cases} \max & z = \displaystyle\sum_{k=1}^{K} y_k \\ \text{s.t.} & g_k(x) \geq y_k, \ \forall k = 1, \dots, K, \\ & y_k = 0 \text{ or } 1, \ \forall k, \\ & x = (x_1, \dots, x_n)^\top \in S'. \end{cases} \qquad (1)$$

In the above, for all $k = 1, \dots, K$, $g_k(x) \geq 1$ if $\varphi_k(x_{[S_k]})$ = TRUE, and $g_k(x) < 1$ if $\varphi_k(x_{[S_k]})$ = FALSE. Note that in the case of RLFAP, this specializes to: $g_k(x) = |x_i - x_j|/w_k$, where x_i and x_j are the two variables involved in constraint k, and w_k the weight of constraint k.

A *relaxation* of an optimization problem such as (1) is obtained by replacing its solution set by a larger solution set. Clearly if the relaxed problem can be solved exactly (i. e. to guaranteed optimality) then its optimal objective function value is an upper bound (in case of maximization) to the optimum objective function value of the original problem.

There exists a number of standard ways of relaxing an optimization problem such as (1), e. g. using Lagrangian relaxation (e. g. [10]) or considering the so-called *continuous relaxation* of some of the variables (e. g. relaxing the constraints on the y_k variables in (1) to $0 \leq y_k \leq 1$). However, in our treatment of RLFAP, those standard relaxations have not been considered because they do not give rise to easily solvable relaxed problems. We therefore investigated a different approach according to the following general principle.

The relaxations we consider are based on the identification of those parts of the constraint graph or hypergraph which are responsible for the infeasibility of the whole problem. Preliminary computational results obtained in [25] have shown that, at least for MAX-CSP problems deriving from RLFAP, it is most often possible to identify in a given instance an infeasible induced subproblem of sufficiently reduced size to make the corresponding MAX-CSP bound computable in reasonable time.

This suggests to consider relaxations of (1) formed by subproblems induced by properly chosen subsets of constraints. Thus, if $\mathcal{K}' \subset \mathcal{K} = \{1, \dots, K\}$ is the subset of constraints chosen, the induced relaxation considered is:

$$\begin{cases} \max & z = \sum_{k=1}^{K} y_k \\ \text{s.t.} & g_k(x) \geq y_k, \ \forall k \in \mathcal{K}', \\ & y_k = 0 \text{ or } 1, \ \forall k = 1, \dots, K, \\ & x \in S'. \end{cases} \tag{2}$$

Note that, in an optimal solution to (2)

$$k \in \mathcal{K} \backslash \mathcal{K}' \Rightarrow y_k = 1.$$

Therefore $\overline{z}$, the optimum objective function value of (2), may be rewritten as:

$$\overline{z} = K - |\mathcal{K}'| + \overline{z}',$$

where $\overline{z}'$ is the optimum value of the problem:

$$R[\mathcal{K}'] \quad \begin{cases} \max & z' = \sum_{k \in \mathcal{K}'} y_k \\ \text{s.t.} & g_k(x) \geq y_k, \ \forall k \in \mathcal{K}', \\ & y_k = 0 \text{ or } 1, \ \forall k \in \mathcal{K}', \\ & x \in S'. \end{cases}$$

Clearly, the constraint graph or hypergraph G' corresponding to a relaxation $R[\mathcal{K}']$ is deduced from the constraint graph or hypergraph G by deleting all edges associated with the constraints in $\mathcal{K} \setminus \mathcal{K}'$. Also observe that if G' has several distinct connected components, then the solution of $R[\mathcal{K}']$ *decomposes into independent subproblems*, one for each connected component.

If the constraint graph or hypergraph G' is of sufficiently small size, then it is possible to solve $R[\mathcal{K}']$ exactly, and the optimum solution value obtained clearly leads to an upper bound to the optimum value of the original problem. When G' is too large to get the exact optimal solution value of $R[\mathcal{K}']$ then we will content ourselves with getting an *upper bound* to this exact optimal value (see the procedure SOLVE.RELAX below). Clearly, any such upper bound still provides a valid upper bound to the original problem. Of course, in the above approach, the quality of the bound derived from $R[\mathcal{K}']$ essentially depends on how to select the subset $\mathcal{K}'$. We now describe the selection procedure which has been used in our computational experiments.

Building Relaxations for RLFAP Using Maximum Cliques

We now specialize the general relaxation scheme described above to derive bounds for RLFAP. The presentation below improves and extends our preliminary work in [25].

The basic idea of our selection procedure for choosing $\mathcal{K}' \subset \mathcal{K}$ is that, for RLFAP, infeasibility is more likely to occur on subsets of links which are all mutually constrained, i. e. on subsets of links which induce a *clique* (complete subgraph) in the constraint graph. Since for RLFAP the constraint graphs arising from applications are always very sparse (less than 1% density for the CELAR instances), it is known that finding a clique of maximum cardinality can be efficiently done even using simple approaches such as implicit enumeration.

In [6] an efficient implicit enumeration based algorithm with good computational results for large sparse graphs up to 3000 vertices is described; however, it assumes very small maximum clique sizes (in the computational results presented in [6], *maximum clique* sizes do not exceed 11, and the running times seem to increase extremely fast with this parameter). Unfortunately, in view of the fact that, for our large RLFAP instances, the maximum clique sizes turned out to be commonly in the range [12, 25], the above algorithm could not be used.

We therefore worked out a different implementation of the implicit enumeration technique which allowed us to find guaranteed maximum cliques for all the test problems treated within acceptable computing times (see results at the end of the paper). Using this maximum clique algorithm, the procedure for building a relaxation to MAX-CSP for RFLAP is as follows.

The heuristic solution method used in our experiments to implement step b1) is a variant of *local search* consisting in iteratively improving an initial starting solution; at each iteration an exact tree search is carried out to find an optimal solution to a subproblem involving only a few variables. In our computational experiments we observed that the impact of the quality of the heuristic solutions produced at step b1) on the quality of the relaxation obtained at the end of BUILD.RELAX was practically negligible (the main reason for this is that $\overline{\sigma}$ is only used as a stopping criterion in the process of successive extraction of maximum cliques). The computational results shown below confirm that bounds of good average quality indeed result from the above construction.

```
a     Set: $\overline{G} = [\overline{X}, \overline{U}] \leftarrow G$ (the initial constraint graph), $i \leftarrow 0$
b     Current step:
   1  Apply a heuristic algorithm to get a good approximate solution to MAX-CSP on $\overline{G}$.
      Let $\overline{\sigma}$ denote the number of constraints satisfied in this solution.
      IF $\overline{\sigma} = |\overline{U}|$ go to c) (end of the construction),
      ELSE set: $i \leftarrow i + 1$.
   2  Look for a maximum clique on $\overline{G}$. Let $C_i$ be the clique obtained, with node set
      $N(C_i)$ and edge set $E(C_i)$.
   3  Let $\overline{G'}$ denote the subgraph of $\overline{G}$ induced by $\overline{X} \setminus N(C_i)$ (obtained from $\overline{G}$ by deleting
      all edges having at least one endpoint in $N(C_i)$).
      Set $\overline{G} \leftarrow \overline{G'}$ and return to b).
c     IF $i = 0$, the problem is feasible and step b1) produces an assignment satisfying all
      the constraints. Terminate.
      ELSE the relaxation $R[\mathcal{K}']$ obtained corresponds to the set $\mathcal{K}'$ of all constraints in
      $\cup_{j=1}^{i} E(C_j)$.
```

Procedure BUILD.RELAX

Solving the Relaxed Problem $R[\mathcal{K}']$

In order to solve the relaxed problem $R[\mathcal{K}']$ we use a basic procedure called FIND.SOLUTION($R[\mathcal{K}']$, θ) which, for any integer value $\theta \in [1, |\mathcal{K}'|]$, answers YES or NO depending on whether there exists a solution to $R[\mathcal{K}']$ with objective function value $z \geq \theta$ or not. In case of a YES answer, the procedure also exhibits the corresponding solution. We assume that this procedure is *exact* i. e. always finds the right answer. Clearly, any value of θ leading to a NO answer produces an *upper bound* to the optimal solution value of $R[\mathcal{K}']$.

The procedure SOLVE.RELAX($R[\mathcal{K}']$) determines a decreasing sequence of upper bounds to the optimal value of $R[\mathcal{K}']$ until either termination is obtained (at step c)) or the maximum computation time has been reached.

In the former case, the exact optimum solution value to $R[\mathcal{K}']$ is obtained; in the latter case, only an upper bound to this optimal value is produced.

```
a  Initialization: Set $\theta \leftarrow | \mathcal{K}' |$.
b  Current step:
   Apply FIND.SOLUTION($R[\mathcal{K}'], \theta$)
   IF the answer is NO,
   THEN set $\theta \leftarrow \theta - 1$ and return to b).
   ELSE perform step c).
c  A YES answer has been obtained at step b): $\theta$ is the optimal solution value to $R[\mathcal{K}']$. Terminate.
```

Procedure SOLVE.RELAX($R[\mathcal{K}']$)

When G', the constraint graph of $R[\mathcal{K}']$ has several distinct connected components corresponding to subsets of constraints, $\mathcal{K}'_1, \ldots, \mathcal{K}'_p$, then solving $R[\mathcal{K}']$ decomposes into the solution of several smaller subproblems $R[\mathcal{K}_1']$, …, $R[\mathcal{K}_p']$. In the procedure SOLVE.RELAX, this decomposability may be exploited in various possible ways. In our implementation, this is done by organizing the computation into *phases* numbered $t = 0, 1, \ldots$. The current upper bound value UB is initialized by: UB $\leftarrow |\mathcal{K}'|$. The current phase t consists in running the procedure FIND.SOLUTION on each of the subproblems $R[\mathcal{K}'_j]$, $j = 1, \ldots, p$, with the parameter $\theta = |\mathcal{K}'_j| - t$. Each time a NO answer is obtained, UB is updated by UB $\leftarrow$ UB $- 1$. Clearly with the above process, when a YES answer has been obtained for some subproblem R$[\mathcal{K}_j']$ during phase t, this subproblem should not be considered any more at later phases $t' > t$. The computation stops either at the end of a phase during which a YES answer has been obtained for all subproblems; or when a user-specified time limit has been reached.

The basic procedure FIND.SOLUTION has been implemented as a classical depth first tree search process of the implicit enumeration type, (achieved by means of a recursive C function). Since getting the exact answer (YES or NO) is essential to the derivation of our bounds, the procedure FIND.SOLUTION is run until full completion of the tree search (i. e. when all the nodes of the tree have been explored implicitly or explicitly).

Computational Results

In order to validate the above described approach, systematic computational experiments have been carried out on two series of test problems.

The first set was composed of 15 infeasible real problems which arose from actual network engineering studies carried out on three distinct large radio link networks (one in the 2GHz frequency range, one in the 2, 5GHz frequency range and one in the 4GHz frequency range).

The second series concerned a set of 5 × 15 = 75 ‘realistic’ test problems generated by applying some random perturbation to the above 15 real problems. More precisely, each problem of the second series is generated from one problem of the first series by changing the weight w_{ij} of each inequality constraint of the form: $|x_i - x_j| \geq w_{ij}$ to: $\widetilde{w}_{ij} = w_{ij} \times (\alpha + \beta\Phi)$ where Φ is a pseudorandom number drawn from a uniform distribution on [0, 1] and α, β are chosen parameters (of course the pseudorandom drawing is assumed to be independent from one constraint to the next).

Maximum Constraint Satisfaction: Relaxations and Upper Bounds, Table 1

Prob. #	n	K	NF	Relaxation # var.	# const.
1	680	2389	8	44	257
2	680	3367	16	38	339
3	680	4103	24	84	671
4	680	2725	8	74	490
5	680	2576	8	46	311
6	680	2470	8	44	284
7	831	3451	16	16	113
8	831	4802	24	33	248
9	396	1792	12	70	375
10	396	1792	12	70	375
11	396	1792	12	70	375
12	396	1792	12	70	375
13	396	1792	12	70	375
14	396	1792	12	70	375
15	396	1792	12	70	375

Maximum Constraint Satisfaction: Relaxations and Upper Bounds, Table 2

Prob. #	HS	Best upper bound obtained within 15s	5’	1 h
1	2376	2387	2385	2383
2	3358	3367	3366	3365
3	4090	4102	4098	4098
4	2700	2720	2713	2708
5	2559	2571	2569	2564
6	2457	2467	2464	2459
7	3440	3450	3450	3450
8	4781	4800	4800	4799
9	1762	1786	1780	1777
10	1759	1786	1780	1776
11	1761	1786	1780	1778
12	1764	1786	1780	1776
13	1761	1786	1780	1775
14	1757	1786	1780	1775
15	1764	1786	1783	1777

Table 1 presents the characteristics of the 15 real test problems treated, numbered 1 to 15 and provides for each problem: number of variables (n), number of constraints (K), number of distinct frequencies used (NF) and the main characteristics of the relaxed subproblem obtained from the procedure BUILD.RELAX: number of variables #var, and number of constraints #const.

Table 3 presents in a similar way the characteristics of the 5 × 15 = 75 test problems deduced from the previous ones by random perturbation. The 5 instances corresponding to each basic problem i are numbered $i_1, \ldots,$ i_5. For each instance the values of the parameters α and β used to generate the instance are displayed together with the characteristics (number of variables, number of constraints) of the relaxed subproblem produced by BUILD.RELAX.

The computation times taken to construct the relaxed subproblems (using BUILD.RELAX) on the problems of Tables 1 and 3, are all between 5 minutes to 35 minutes with an average of about 12 minutes.

Table 2 shows the results obtained on the 15 real test problems of Table 1 and Table 4 shows the results for the 5 × 15 problems of Table 3. The computer used

Maximum Constraint Satisfaction: Relaxations and Upper Bounds, Table 3

Prob.	α	β	Relaxation	
#			#var.	#const.
1_1	0, 5	1	42	257
1_2	0, 5	1	42	261
1_3	0, 8	0, 4	42	261
1_4	0, 8	0, 4	42	261
1_5	0, 8	0, 4	42	261
2_1	0, 5	1	38	339
2_2	0, 5	1	38	339
2_3	0, 8	0, 4	38	339
2_4	0, 8	0, 4	38	339
2_5	0, 8	0, 4	38	339
3_1	0, 5	1	54	671
3_2	0, 5	1	70	460
3_3	0, 8	0, 4	84	480
3_4	0, 8	0, 4	84	671
3_5	0, 8	0, 4	54	671
4_1	0, 5	1	74	490
4_2	0, 5	1	74	490
4_3	0, 8	0, 4	74	490
4_4	0, 8	0, 4	74	490
4_5	0, 8	0, 4	74	490
5_1	0, 5	1	46	311
5_2	0, 5	1	46	311
5_3	0, 8	0, 4	46	311
5_4	0, 8	0, 4	46	311
5_5	0, 8	0, 4	46	311
6_1	0, 5	1	44	284
6_2	0, 5	1	44	284
6_3	0, 8	0, 4	44	284
6_4	0, 8	0, 4	44	284
6_5	0, 8	0, 4	44	284
7_1	0, 8	0, 4	16	113
7_2	0, 8	0, 4	16	113
7_3	0, 8	0, 4	16	113
7_4	0, 8	0, 4	16	113
7_5	0, 8	0, 4	16	113
8_1	0, 5	1	33	248
8_2	0, 5	1	33	248
8_3	0, 5	1	33	248
8_4	0, 5	1	33	248
8_5	0, 5	1	33	248
9_1	0, 2	1, 6	12	375
9_2	0, 2	1, 6	12	66
9_3	0, 2	1, 6	48	66
9_4	0, 2	1, 6	24	264
9_5	0, 2	1, 6	36	132
10_1	0, 2	1, 6	36	375
10_2	0, 2	1, 6	12	198
10_3	0, 2	1, 6	48	66
10_4	0, 2	1, 6	24	264
10_5	0, 2	1, 6	36	132
11_1	0, 2	1, 6	36	375
11_2	0, 2	1, 6	12	198
11_3	0, 2	1, 6	48	66
11_4	0, 2	1, 6	24	264
11_5	0, 2	1, 6	36	132
12_1	0, 2	1, 6	24	375
12_2	0, 2	1, 6	12	132
12_3	0, 2	1, 6	48	66
12_4	0, 2	1, 6	24	264
12_5	0, 2	1, 6	36	162
13_1	0, 2	1, 6	36	375
13_2	0, 2	1, 6	12	198
13_3	0, 2	1, 6	48	66
13_4	0, 2	1, 6	24	264
13_5	0, 2	1, 6	36	132
14_1	0, 2	1, 6	36	375
14_2	0, 2	1, 6	12	198
14_3	0, 2	1, 6	48	66
14_4	0, 2	1, 6	24	264
14_5	0, 2	1, 6	36	132
15_1	0, 2	1, 6	24	375
15_2	0, 2	1, 6	12	132
15_3	0, 2	1, 6	48	66
15_4	0, 2	1, 6	24	264
15_5	0, 2	1, 6	36	132

was a PC Pentium 166 workstation with 32Mb RAM. For each problem we provide: HS, the best heuristic solution value obtained (number of satisfied constraints); the best upper bounds obtained after 15 seconds, 5 minutes and 1 hour. The results in Table 2 confirm that our approach is practical to consistently produce good bounds for real RLFAP instances within acceptable solution times.

Maximum Constraint Satisfaction: Relaxations and Upper Bounds, Table 4

Prob. #	HS	Best upper bound obtained within			Prob. #	HS	Best upper bound obtained within		
		15 s	5'	1 h			15 s	5'	1 h
1_1	2376	2386	2383	2378	9_1	1779	1791	1791	1791
1_2	2376	2386	2383	2378	9_2	1777	1791	1790	1789
1_3	2376	2386	2383	2378	9_3	1774	1788	1788	1785
1_4	2376	2386	2383	2378	9_4	1777	1790	1789	1789
1_5	2376	2386	2383	2378	9_5	1779	1789	1787	1787
2_1	3358	3366	3365	3365	10_1	1780	1789	1788	1787
2_2	3358	3367	3365	3365	10_2	1780	1791	1790	1788
2_3	3358	3367	3366	3365	10_3	1776	1788	1787	1785
2_4	3358	3367	3365	3365	10_4	1778	1790	1789	1789
2_5	3358	3367	3366	3365	10_5	1777	1789	1788	1788
3_1	4081	4103	4101	4101	11_1	1783	1789	1789	1789
3_2	4081	4102	4101	4101	11_2	1780	1791	1789	1788
3_3	4086	4102	4098	4098	11_3	1777	1788	1788	1786
3_4	4086	4102	4098	4098	11_4	1780	1790	1789	1789
3_5	4088	4102	4101	4101	11_5	1779	1789	1788	1787
4_1	2700	2720	2713	2708	12_1	1779	1790	1790	1789
4_2	2700	2720	2713	2708	12_2	1780	1791	1790	1789
4_3	2700	2720	2713	2708	12_3	1777	1788	1788	1786
4_4	2700	2720	2713	2708	12_4	1780	1790	1789	1787
4_5	2700	2720	2713	2708	12_5	1778	1789	1788	1787
5_1	2559	2571	2569	2564	13_1	1782	1789	1789	1788
5_2	2559	2572	2569	2564	13_2	1777	1790	1789	1789
5_3	2559	2572	2569	2564	13_3	1776	1788	1787	1786
5_4	2559	2573	2569	2564	13_4	1779	1790	1789	1789
5_5	2559	2573	2569	2564	13_5	1777	1789	1788	1788
6_1	2457	2467	2464	2459	14_1	1782	1789	1789	1788
6_2	2457	2467	2464	2459	14_2	1775	1791	1789	1789
6_3	2457	2467	2464	2459	14_3	1775	1788	1787	1786
6_4	2457	2467	2464	2459	14_4	1779	1791	1789	1789
6_5	2457	2467	2464	2459	14_5	1776	1789	1788	1788
7_1	3438	3450	3450	3450	15_1	1780	1790	1790	1789
7_2	3437	3450	3450	3450	15_2	1779	1791	1789	1788
7_3	3421	3430	3430	3430	15_3	1777	1788	1788	1788
7_4	3414	3424	3424	3424	15_4	1781	1790	1789	1788
7_5	3436	3450	3450	3450	15_5	1780	1789	1788	1788
8_1	4780	4800	4800	4799					
8_2	4783	4800	4800	4799					
8_3	4778	4800	4800	4799					
8_4	4781	4800	4800	4799					
8_5	4781	4800	4800	4799					

From Tables 2 and 4, it is seen that for all the instances treated, the difference between the heuristic solution values HS and the best upper bounds obtained are always quite small. More precisely for all the examples treated, the ratio $R = (\text{UB} - \text{HS})/\text{UB}$ is most often well below 1% (Problem 14 in Table 2 is the only one for which $R > 1\%$). We note that since HS is only a *lower bound*, R is a pessimistic estimate of the relative difference between the best upper bound obtained and the optimal, unknown, solution value.

Also, from Table 4, it is seen that the results obtained appear to be fairly stable, in spite of the importance of the perturbations applied to generate the corresponding 75 instances. In addition to practical applicability, and efficiency, this clearly shows good stability and robustness in the behavior of our algorithms. To our knowledge, this is the first time a systematic way of deriving upper bounds to such large scale MAX-CSP problems has been implemented and fully tested.

To conclude, let us mention that, in view of the results obtained, the techniques described here have been included in an industrial software tool for radio network engineering developed by the French MOD (DGA/CELAR).

See also

► Frequency Assignment Problem
► Graph Coloring

References

1. Babel L, Tinhofer G (1990) A branch and bound algorithm for the maximum clique problem. ZOR: Methods and Models of Oper Res 34:207–217
2. Berge C (1973) Graphes et hypergraphes, 2nd edn. Dunod, Paris
3. Boros E, Prékopa A (1988) Probabilistic bounds and algorithms for the maximum satisfiability problem. RUTCOR Res. Report Rutgers Univ. New Jersey (USA) RRR#17-88
4. Bouju A, Boyce JF, Dimitropoulos CHD, von Scheidt G, Taylor JG (1995) Tabu search for the radio link frequency assignment problem. Proc. Conf. on Applied Decision Technologies: Modern Heuristic methods, Brumel Univ., Uxbridge (UK), Uxbridge, UK pp 233–250 work carried out in the CALMA PROJECT.
5. Bourret P (1995) Simulated annealing. Deliverable 2.3 of the CALMA Project. Report 3/3507.00/DERI-ONERA-CERT, CALMA
6. Carraghan R, Pardalos PM (1990) An exact algorithm for the maximum clique problem. Oper Res Lett 9:375–382
7. Castelino DJ, Hurley S, Stephens NM (1996) A tabu search algorithm for frequency assignment. Ann Oper Res 63:301–319
8. CELAR (1994) Radio link frequency assignment problem benchmark, CELAR, ftp.cs.unh.edu/pub/csp/archive/code/benchmarks
9. Du D-Z, Gu J, Pardalos PM (eds) (1997) Satisfiability problem: Theory and applications. DIMACS, vol 35. Amer Math Soc
10. Fisher ML (1981) The Lagrangian relaxation method for solving integer programming problems. Managem Sci 27:11–18
11. Freuder EC, Wallace RJ (1992) Partial constraint satisfaction. Artif Intell 58:21–70
12. Garey MR, Johnson DS (1979) Computers and intractability. A guide to the theory of NP-completeness. Freeman, New York
13. Garey MR, Johnson DS, Stokmeyer L (1976) Some simplified NP-complete graph problems. Theoret Comput Sci 1:237–267
14. Gondran M, Minoux M (1995) Graphes et algorithmes, 3rd edn. Eyrolles, Paris
15. Hajema W, Minoux M, West C (June 25, 1992) CALMA project specification. Statement of the Radio Link Frequency Assignment Problem. Appendix 3 to EUCLID RTP, 6-4 Implementing Arrangement
16. Hale WK (1980) Frequency assignment theory and applications. Proc IEEE 68(12):1497–1514
17. Hansen P, Jaumard B (1987) Algorithms for the maximum satisfiability problem. RUTCOR Res. Report Rutgers Univ. New Jersey (USA) RRR#43-87
18. Hurley S, Thiel SU, Smith DH (1996) A comparison of local search algorithms for radio link frequency assignment problems. Proc. ACM Symposium on Applied Computing, pp 251–257
19. Johnson DS (1974) Approximation algorithms for combinatorial problems. J Comput Syst Sci 9:256–278
20. Joy S, Mitchell J, Borcher SB (1997) A branch and cut algorithm for MAX-SAT and weighted MAX-SAT. In: Satisfiability Problem: Theory and Appl. In: DIMACS, vol 35. Amer Math Soc, pp 519–536
21. Kumar V (1992) Algorithms for constraint satisfaction problems: A survey. AI Magazine Spring:32–44
22. Lanfear TA (1989) Graph theory and radio frequency assignment. NATO EMC Analysis Project, vol 5. NATO, Brussels
23. Lieberherr KJ (1982) Algorithmic extremal problems in combinatorial optimization. J Algorithms 3:225–244
24. Lieberherr KJ, Specker E (1981) Complexity of partial satisfaction. J ACM 28(2):411–421
25. Mavrocordatos P, Minoux M (1995) Allocation de ressources dans les réseaux (frequency allocation in networks). Final Techn. Report CELAR contract #0114193. Résolution de problèmes d'optimisation combinatoire pour application à l'allocation optimisée de fréquences dans les grands réseaux.
26. Pardalos PM, Rodgers GP (1992) A branch and bound algorithm for the maximum clique problem. Comput Oper Res 19(5):363–375

27. Poljak S, Turzik D (1982) A polynomial algorithm for constructing a large bipartite subgraph, with an application to a satisfiability problem. Canad J Math XXXIV(3):519–524
28. Resende MGC, Pitsoulis LS, Pardalos PM (1997) Approximate solution of weighted MAX-SAT problems using GRASP. Satisfiability Problem: Theory and Appl. In: DIMACS, vol 35. Amer Math Soc, pp 393–405
29. Roberts FS (1991) T-colorings of graphs; recent results and open problems. Discret Math 93:229–245
30. Smith DH, Hurley S (1997) Bounds for the frequency assignment problem. Discret Math 167/168:571–582
31. Smith DH, Hurley S, Thiel SU (1998) Improving heuristics for the frequency assignment problem. Europ J Oper Res 107:76–86
32. de Werra D, Gay Y (1994) Chromatic scheduling and frequency assignment. Discrete Appl Math 49:165–174

Maximum Cut Problem, MAX-CUT

CLAYTON W. COMMANDER
Air Force Research Laboratory, Munitions Directorate, and Dept. of Industrial and Systems Engineering, University of Florida, Gainesville, USA

Article Outline

Introduction

The MAXIMUM CUT problem (MAX-CUT) is one of the simplest graph partitioning problems to conceptualize, and yet it is one of the most difficult combinatorial optimization problems to solve. The objective of MAX-CUT is to partition the set of vertices of a graph into two subsets, such that the sum of the weights of the edges having one endpoint in each of the subsets is maximum. This problem is known to be $\mathcal{NP}$-complete [18,27]; however, it is interesting to note that the inverse problem, i. e., that of looking for the minimum cut in a graph is solvable in polynomial time using network flow techniques [1]. MAX-CUT is an important combinatorial problem and has applications in many fields including VLSI circuit design [9,32] and statistical physics [5]. For other applications, see [16,21]. For a detailed survey of MAX-CUT, the reader can refer to [33].

Organization

In this paper, we introduce the MAXIMUM CUT problem and review several heuristic methods which have been applied. In Subsect. "C-GRASP Heuristic" we describe the implementation of a new heuristic based optimizing a quadratic over a hypercube. The heuristic is designed under the C-GRASP (Continuous Greedy Randomized Adaptive Search Procedure) framework. Proposed by Hirsch, Pardalos, and Resende [23], C-GRASP is a new stochastic metaheuristic for continuous global optimization problems. Numerical results are presented and compared with other heuristics from the literature.

Idiosyncrasies

We conclude this section by introducing the symbols and notations we will employ throughout this paper. Denote a graph $G = (V, E)$ as a pair consisting of a set of vertices V, and a set of edges E. Let the map $w\colon E \mapsto \mathbb{R}$ be a weight function defined on the set of edges. We will denote an edge-weighted graph as a pair (G,w). Thus we can easily generalize an un-weighted graph $G = (V, E)$ as an edge-weighted graph (G,w), by defining the weight function as

$$w_{ij} := \begin{cases} 1, & \text{if } (i, j) \in E\,, \\ 0, & \text{if } (i, j) \notin E\,. \end{cases} \qquad (1)$$

We use the symbol "$b := a$" to mean "the expression a defines the (new) symbol b". Of course, this could be conveniently extended so that a statement like "$(1-\epsilon)/2 := 7$" means "define the symbol ϵ so that $(1-\epsilon)/2 = 7$ holds". We will employ the typical symbol S^c to denote the complement of the set S; further let $A \setminus B$ denote the set-difference, $A \cap B^c$. Agree to let the expression $x \leftarrow y$ mean that the value of the variable y is assigned to the variable x. Finally, to denote the cardinality of a set S, we use $|S|$. We will use **bold** for words which we define, *italics* for emphasis, and SMALL CAPS for problem names. Any other locally used terms and symbols will be defined in the sections in which they appear.

Formulation

Consider an undirected edge-weighted graph (G,w), where $G = (V, E)$ is the graph, and w is the weight function. A **cut** is defined as a partition of the vertex set into two disjoint subsets S and $\bar{S} := V \setminus S$. The **weight** of the cut $(S, \bar{S})$ is given by the function $W\colon S \times \bar{S} \mapsto \mathbb{R}$ and is defined as

$$W(S, \bar{S}) := \sum_{i \in S, j \in \bar{S}} w_{ij} \,. \tag{2}$$

For an edge-weighted graph (G,w), a **maximum cut** is a cut of maximum weight and is defined as

$$MC(G, w) := \max_{\forall S \subseteq V} W(S, V \setminus S) \,. \tag{3}$$

We can formulate MAX-CUT as the following integer quadratic programming problem:

$$\max \ \frac{1}{2} \sum_{1 \le i < j \le n} w_{ij}(1 - y_i y_j) \tag{4}$$

subject to:

$$y_i \in \{-1, 1\}, \qquad \forall i \in V \,. \tag{5}$$

To see this, notice that each subset $V \supseteq S := \{i \in V : y_i = 1\}$ induces a cut $(S, \bar{S})$ with corresponding weight equal to

$$W(S, \bar{S}) = \frac{1}{2} \sum_{1 \le i < j \le n} w_{ij}(1 - y_i y_j) \,. \tag{6}$$

An alternative formulation of MAX-CUT based on the optimization of a quadratic over the unit hypercube was given by Deza and Laurent in [12].

Theorem 1 *Given a graph $G = (V, E)$ with $|V| = n$, the optimal objective function value of the* MAXIMUM CUT *problem is given by*

$$\max_{x \in [0,1]^n} x^{\mathrm{T}} W(e - x) \,, \tag{7}$$

where $W = [w_{ij}]_{i,j=1}^{n}$ is the matrix of edge weights, and $e := [1, 1, \ldots, 1]^{\mathrm{T}}$ is the unit vector.

Proof 1 Let

$$f(x) := x^{\mathrm{T}} W(e - x) \tag{8}$$

denote the objective function from Eq. (7). To begin with, notice that the matrix W has a zero diagonal, i. e., $w_{ii} = 0$, $\forall i \in 1, 2, \ldots, n$. This implies that $f(x)$ is linear with respect to each variable, and thus there always exists an optimal solution, x^* of (7) such that $x^* \in \{0, 1\}^n$. Therefore, we have shown that

$$\max_{x \in [0,1]^n} x^{\mathrm{T}} W(e - x) = \max_{x \in \{0,1\}^n} x^{\mathrm{T}} W(e - x). \tag{9}$$

The next step is to show that there is a bijection between binary vectors of length n and cuts in G. Consider *any* binary vector $\hat{x} \in \{0, 1\}^n$. Now suppose we partition the vertex set V into two disjoint subsets $V_1 := \{i | \hat{x}_i = 0\}$ and $V_2 := \{i | \hat{x}_i = 1\}$. Then, evaluating the objective function we have

$$f(\hat{x}) = \sum_{(i,j) \in V_1 \times V_2} w_{ij}, \tag{10}$$

which is equal to $W(V_1, V_2)$, the value of the cut defined by the partition of $V = V_1 \bigcup V_2$ (see Eq. (2) above).

Alternatively, consider *any* partition of V into two disjoint subsets $V_1, V_2 \subseteq V$. That is

$$V = V_1 \bigcup V_2 \quad \text{and} \quad V_1 \bigcap V_2 = \emptyset \,.$$

Now, we can construct the vector $\hat{x}$ as follows:

$$\hat{x}_i = \begin{cases} 1, & \text{if } i \in V_1 \\ 0, & \text{if } i \in V_2 \,. \end{cases} \tag{11}$$

Once again, evaluating the objective function on $\hat{x}$, we have

$$f(\hat{x}) = \sum_{(i,j) \in V_1 \times V_2} w_{ij} \,. \tag{12}$$

Hence $f(\hat{x}) = W(V_1, V_2)$ and we have the result.[1] Alas, we have shown the bijection between binary n-vectors

[1]Notice that the result holds even if (without the loss of generality) $V_1 = V$ and $V_2 = \emptyset$. In this case, a cut induced by (V_1, V_2) will be a maximum cut if $w_{ij} \le 0, \quad \forall i, j \in V$.

and cuts in G. In summary, we have

$$\begin{aligned}&\max_{x\in[0,1]^n} x^{\mathrm{T}} W(e-x)\\ &= \max_{x\in\{0,1\}^n} x^{\mathrm{T}} W(e-x)\\ &= \max_{V=V_1\bigcup V_2, V_1\bigcap V_2=\emptyset} \sum_{(i,j)\in V_1\times V_2} w_{ij}\,.\end{aligned}$$

□

There are several classes of graphs for which MAX-CUT is solvable in polynomial time [25]. These include planar graphs [11], weakly bipartite graphs with non-negative edge weights [20], and graphs without K_5 minors [4]. The general problem however is known to be $\mathcal{APX}$-complete [31]. This implies that unless $\mathcal{P} = \mathcal{NP}$, MAX-CUT does not admit a polynomial time approximation scheme [30].

Methods

The MAXIMUM CUT problem is one of the most well-studied discrete optimization problems [27]. Since the problem is $\mathcal{NP}$-hard in general, there has been an incredible amount of research done in which heuristic techniques have been applied. Before we present the new heuristic approach, we review some of the prior work that has been done.

Review of Solution Approaches

There have been many semidefinite and continuous relaxations based on this formulation. This was first shown by Lovász in [28]. In 1995, Goemans and Williamson [19] used a semidefinite relaxation to achieve an approximation ratio of .87856. This implication of this work is significant for two reasons. The first is of course, the drastic improvement of the best known approximation ratio for MAX-CUT of 0.5 which had not been improved in over 20 years [36]. Secondly, and perhaps more significantly is that until 1995, research on approximation algorithms for nonlinear programming problems did not receive much attention. Motivated by the work of Goemans and Williamson, semidefinite programming techniques were applied to an assortment of combinatorial optimization problems successfully yielding the best known approximation algorithms for GRAPH COLORING [7,26], BETWEENNESS [10], MAXIMUM SATISFIABILITY [13,19], and MAXIMUM STABLE SET [2], to name a few [29].

As noted in [16], the use of interior point methods for solving the semidefinite programming relaxation have proven to be very efficient. This is because methods such as the one proposed by Benson, Ye, and Zhang in [6] exploit the combinatorial structure of the relaxed problem. Other algorithms based on the nonlinear semidefinite relaxation include the work of Helmberg and Rendl [22] and Homer and Peinado [24].

The work of Burer et al. in [8] describes the implementation of a rank-2 relaxation heuristic dubbed `circut`. This software package was shown to compute better solutions than the randomized heuristic of Goemans and Williamson, in general [16]. In a recent paper dating from 2002, Festa, Pardalos, Resende, and Ribeiro [16] implement and test six randomized heuristics for MAX-CUT. These include variants of Greedy Randomized Adaptive Search Procedures (GRASP), Variable Neighborhood Search, and path-relinking algorithms [35]. Their efforts resulted in improving the best known solutions for several graphs and quickly producing solutions that compare favorably with the method of Goemans and Williamson [19] and `circut` [8]. For several sparse instances, the randomized heuristics presented in [16] outperformed `circut`.

In [25], Butenko et al. derive a "worst-out" heuristic having an approximation ratio of at least 1/3 which they refer to as the *edge contraction method*. The also present a computational analysis of several greedy construction heuristics for MAX-CUT based on variations of the 0.5-approximation algorithm of Sahni and Gonzalez [36]. With this, we now move on and describe the implementation of a new heuristic for MAX-CUT based on the new metaheuristic Continuous GRASP [23].

C-GRASP Heuristic

The Continuous Greedy Randomized Adaptive Search Procedure (C-GRASP) is a new metaheuristic for continuous global optimization [23]. The method is an extension of the widely known discrete optimization algorithm Greedy Randomized Adaptive Search Procedure (GRASP) [15]. Preliminary results are quite promising, indicating that C-GRASP is able to quickly converge to the global optimum on standard benchmark test func-

```
procedure GRASP(MaxIter, RandomSeed)
1  $f^* \leftarrow 0$
2  $X^* \leftarrow \emptyset$
3  for $i = 1$ to MaxIter do
4      $X \leftarrow$ ConstructionSolution($G, g, X, \alpha$)
5      $X \leftarrow$ LocalSearch($X, N(X)$)
6      if $f(X) \geq f(X^*)$ then
7          $X^* \leftarrow X$
8          $f^* \leftarrow f(X)$
9      end
10 end
11 return $X^*$
end procedure GRASP
```

Maximum Cut Problem, MAX-CUT, Figure 1
GRASP for maximization

tions. The traditional GRASP is a two-phase procedure which generates solutions through the controlled use of random sampling, greedy selection, and local search. For a given problem Π, let F be the set of feasible solutions for Π. Each solution $X \in F$ is composed of k discrete components $a_1, \ldots, a_k$. GRASP constructs a sequence $\{X\}_i$ of solutions for Π, such that each $X_i \in F$. The algorithm returns the best solution found after all iterations. The GRASP procedure can be described as in the pseudo-code provided in Fig. 1. The *construction phase* receives as parameters an instance of the problem G, a ranking function $g\colon A(X) \mapsto \mathbb{R}$ (where $A(X)$ is the domain of feasible components $a_1, \ldots, a_k$ for a partial solution X), and a parameter $0 < \alpha < 1$. The construction phase begins with an empty partial solution X. Assuming that $|A(X)| = k$, the algorithm creates a list of the best ranked αk components in $A(X)$, and returns a uniformly chosen element x from this list. The current partial solution is augmented to include x, and the procedure is repeated until the solution is feasible, i. e., until $X \in F$.

The *intensification phase* consists of the implementation of a hill-climbing procedure. Given a solution $X \in F$, let $N(X)$ be the set of solutions that can found from X by changing one of the components $a \in X$. Then, $N(X)$ is called the neighborhood of X. The improvement algorithm consists of finding, at each step, the element X^* such that

$$X^* := \arg\max_{X' \in N(X)} f(X') ,$$

where $f\colon F \mapsto \mathbb{R}$ is the objective function of the problem. At the end of each step we make the assignment $X^* \leftarrow X$ if $f(X) > f(X^*)$. The algorithm will eventually achieve a local optimum, in which case the solution X^* is such that $f(X^*) \geq f(X')$ for all $X' \in N(X^*)$. X^* is returned as the best solution from the iteration and the best solution from all iterations is returned as the overall GRASP solution. GRASP has been applied to many discrete problems with excellent results. For an annotated bibliography of GRASP applications, the reader is referred to the work of Festa and Resende in [17].

Like GRASP, the C-GRASP framework is a multi-start procedure consisting of a construction phase and a local search [14]. Specifically, C-GRASP is designed to solve continuous problems subject to box constraints. The feasible domain is given as the n-dimensional rectangle $S := \{x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n\colon l \leq x \leq u\}$, where $l, u \in \mathbb{R}^n$ are such that $l_i \leq u_i$, for $i = 1, 2, \ldots, n$. Pseudo-code for the basic C-GRASP is provided in Fig. 2. Notice that the algorithm takes as input the dimension n, upper and lower bounds l and u, the objective function f, and parameters MaxIters, MaxNumIterNoImprov, NumTimesToRun, MaxDirToTry, and a number $\alpha \in (0, 1)$.

To begin with, the optimal objective function value f^* is initialized to $-\infty$. The procedure then enters the main body of the algorithm in the **for** loop from lines 2–21. The value NumTimesToRun is the total number of C-GRASP iterations that will be performed. To begin with, more initialization takes place as the current solution x is initialized as a random point inside the hyperrectangle, which is generated according to a function UnifRand($[l, u)$) which is uniform onto $[l,u)$[2]. Furthermore, the parameter which controls the discretization of the search space, h, is set to 1. Next, the construction phase and local search phases are entered. In line 9, the new solution is compared to the current best solution. If the objective function value corresponding to the current solution dominates the incumbent, then the current solution replaces the incumbent and NumIterNoImprov is set to 0. This parameter

[2]This is the "typical" definition of a *Uniform* distribution. That is, $P : X \mapsto \mathbb{R}$ is uniform onto $[A,B)$, if, for any subinterval $I \subset [A, B)$, the measure of $P^{-1}(I)$ equals the length of I.

```
procedure C-GRASP($n, l, u, f(\cdot)$, MaxIters, MaxNumIterNoImprov, NumTimesToRun,
          MaxDirToTry, $\alpha$)
1   $f^* \leftarrow -\infty$
2   for $j = 1$ to NumTimesToRun do
3      $x \leftarrow$ UnifRand($[l, u)$)
4      $h \leftarrow 1$
5      NumIterNoImprov $\leftarrow 0$
6      for Iter = 1 to MaxIters do
7         $x \leftarrow$ ConstructGreedyRandomized($x, f(\cdot), n, h, l, u, \alpha$)
8         $x \leftarrow$ LocalSearch($x, f(\cdot), n, h, l, u$, MaxDirToTry)
9         if $f(x) \geq f^*$ then
10           $x^* \leftarrow x$
11           $f^* \leftarrow f(x)$
12           NumIterNoImprov $\leftarrow 0$
13        else
14           NumIterNoImprov $\leftarrow$ NumIterNoImprov + 1
15        end if
16        if NumIterNoImprov $\geq$ MaxNumIterNoImprov then
17           $h \leftarrow h/2$
18           NumIterNoImprov $\leftarrow 0$
19        end if
20     end for
21  end for
22  return $x^*$
end procedure C-GRASP
```

Maximum Cut Problem, MAX-CUT, Figure 2
C-GRASP pseudo-code adapted from [23]

controls when the discretization measure h is reduced. That is, after a total of MaxNumIterNoImprov iterations occur in which no solution better than the current best solution is found, h is set to $h/2$ and the loop returns to line 6. By adjusting the value of h, the algorithm is able to locate general areas of the search space which contain high quality solutions, and then narrow down the search in those particular regions. The best solution after a total of NumTimesToRun iterations is returned as the best solution.

The construction phase of the C-GRASP takes as input the randomly generated solution $x \in S$ (see Fig. 2, line 3). Beginning with all coordinates unfixed, the method then performs a line search on each unfixed coordinate direction of x holding the other $n-1$ directions constant. The objective function values resulting from the line search solution for each coordinate direction are stored in a vector, say V. An element $v_i \in V$ is then selected uniformly at random from the maximum $(1-\alpha)100\%$ elements of V, and the v_i coordinate direction is fixed. This process repeats until all n coordinates of x have been fixed. The resulting solution is returned as the C-GRASP solution from the current iteration. For a slightly more detailed explanation of this procedure, the reader is referred to [23].

As for the local search phase, this procedure *simulates* the role of calculating the gradient of the objective function $f(\cdot)$. As mentioned earlier, gradients are not used in C-GRASP because oftentimes, they are difficult to compute and result in slow computation times. Therefore, the gradient is approximated as follows. Given the construction phase solution x, the local search generates a set of directions and determines in which direction (if any) the objective function improves.

The directions are calculated according to a bijective function T which maps the interval of integers $[1, 3^n) \cap \mathbb{Z}$ onto their balanced ternary representation.

Maximum Cut Problem, MAX-CUT, Table 1
Parameters used for C-GRASP

$\alpha = 0.4$	MaxDirToTry = 20
NumTimesToRun = 20	MaxIters = 1000
MaxNumIterNoImrpov = 1	

Recall that n is the dimension of the problem under consideration. That is, $T\colon [1, 3^n) \cap \mathbb{Z} \mapsto \{-1, 0, 1\}^{\mathbb{N}}$. Clearly, as $n \to \infty$, the number of search directions grows exponentially. Therefore, only MaxDirToTry directions are generated[3] and tested on the current solution. For each direction d, the point $\hat{x} := x + hd$ is constructed and $f(\hat{x})$ is computed. Recall that h is the parameter which controls the density of the search space discretization. If the constructed point $\hat{x} \in S$ has a more favorable objective value than the current point x, then $\hat{x}$ replaces x, and the process continues. The phase terminates when a locally optimal point $x^* \in S$ is found. The point x^* is said to be locally optimal if $f(x^*) \geq f(x^* + hd) \forall d \in \{1, 2, \ldots,$ MaxDirToTry$\}$. Again, for a slightly more in depth description of this procedure, the reader should see the paper by Hirsch et al. [23].

Computational Results The proposed procedure was implemented in the C++ programming language and complied using Microsoft® Visual C++ 6.0. It was tested on a PC equipped with a 1700 MHz Intel® Pentium® M processor and 1 GB of RAM operating under the Microsoft® Windows® XP environment. The C-GRASP parameters used are provided in Table 1. First, we tested the C-GRASP on 10 instances produced by the Balasundarm–Butenko problem generator in [3]. Though these problems are relatively small, they have proven themselves to be quite formidable against the Multilevel Coordinate Search (MCS) black-box optimization algorithm. We also tested the C-GRASP on 12 instances from the TSPLIB [34] collection of test problems for the TRAVELING SALESMAN PROBLEM. These problems are also used as benchmark problems for testing MAX-CUT heuristics [19].

For further comparison, all instances were tested using the rank-2 relaxation heuristic circut [8], as well as with a simple 2-exchange local search heuristic which

[3]uniformly at random

```
procedure LocalSearch($G$, MaxIter)
1  $f^* \leftarrow -\infty$
2  $x^* \leftarrow \emptyset$
3  for $j = 1$ to MaxIter do
4      $x \leftarrow$ KruskalMST($x, G$)
5      $x \leftarrow$ LocalImprove($x, G$)
6      if $f(x) \geq f^*$ then
7          $x^* \leftarrow x$
8          $f^* \leftarrow f(x)$
9      end if
10 end for
11 return $x^*$
end procedure LocalSearch
```

Maximum Cut Problem, MAX-CUT, Figure 3
The 2-exchange local search routine

is outlined in the pseudo-code provided in Fig. 3. The method receives as input a parameter MaxIter indicating the maximum number of iterations to be performed and $G = (V, E)$ the instance of the problem whereupon a maximum spanning tree is found using Kruskal's algorithm [1]. The spanning tree, due to its natural bipartite structure provides a feasible solution to which a swap-based local improvement method is applied in line 5. The local improvement works as follows. For all pairs of vertices (u,v) such that $u \in S$ and $v \in \bar{S}$, a swap is performed. That is, we place $u \in \bar{S}$ and $v \in S$. If the objection function is improved, the swap is kept; otherwise, we undo the swap and examine the next (u,v) pair. The local search was tested on the same PC as the C-GRASP. The circut heuristic was compiled using Compaq® Visual Fortran on a PC equipped with a 3.60 GHz Intel® Xeon® processor and 3.0 GB of RAM operating under the Windows® XP environement.

Table 2 provides computational results of the algorithms on the 10 Balasundarum–Butenko instances from [3]. The first three columns provide the instance name, the number of vertices and the optimal solution. The solutions from the heuristics are provided next. The solutions from the Multilevel Coordinate Search algorithm were provided in [3]. For all of these instances, the time required by the C-GRASP, circut, and the local search to find their best solutions was fractions of a second. Computing times were not listed for the MCS algorithm in [3]. Notice that the 2-exchange

Maximum Cut Problem, MAX-CUT, Table 2
Comparative results from the Balasundaram–Butenko instances from [3]

Name	$\|V\|$	Opt	C-GRASP	MCS	circut	LS
G5-1	5	126	126	125	125	126
G5-2	5	40	40	39	40	40
G8-1	8	1987	1802	1802	1987	1987
G8-2	8	1688	1631	1671	1688	1688
G10-1	10	1585	1585	1513	1585	1585
G10-2	10	1377	1373	1373	1377	1377
G15-1	15	399	389	389	399	399
G15-2	15	594	594	593	594	594
G20-1	20	273	267	273	273	273
G20-2	20	285	285	282	285	285

local search computed optimal solutions for each of these instances, followed closely by circut which found optimal cuts for all but one problem. As for the continuous heuristics, the C-GRASP found optimal solutions for 5 of the 10 instances while the MCS procedure produced optimal cuts for only 1 instance. For the 5 instances where C-GRASP produced suboptimal solutions, the average deviation from the optimum was 3.54%.

Table 3 shows results of the C-GRASP, local search, and circut heuristics when applied to 12 instances from the TSPLIB collection of test problems for the TRAVELING SALESMAN problem [34]. The first two columns provide the instance name and the size of the vertex set $|V|$. Next the solutions are provided along with the associated computing time required by the respective heuristic. Notice that for all 12 instances, the three heuristics all found the same solutions. Notice that in terms of computation time, the simplest heuristic, the 2-exchange local search seems to be the best performing of the three methods tested. The rank-2 relaxation algorithm circut is also very fast requiring only 2.99 s on average to compute the solution. On the other hand, the C-GRASP method did not scale as well as the others. We see that there is a drastic increase in the solution time as the number of vertices increases beyond 48.

This is not particularly surprising. The philosophical reasoning behind the slow computation time of the C-GRASP relative to the discrete heuristics being that the C-GRASP is a *black-box* method and does not take into account any information about the problem other than the objective function. To the contrary, the local search and circut specifically exploit the combinatorial structure of the underlying problem. This allows them to quickly calculate high quality solutions.

Conclusions

In this paper, we implemented a new metaheuristic for the MAXIMUM CUT problem. In particular, we proposed the use of a continuous greedy randomized adaptive search procedure (C-GRASP) [23], for a continuous formulation of the problem. To our knowledge,

Maximum Cut Problem, MAX-CUT, Table 3
Comparative results from TRAVELING SALESMAN PROBLEM instances [34]

Name	$\|V\|$	C-GRASP	Time (s)	LS	Time (s)	circut	Time (s)
burma14	14	283	0.120	283	0.00	283	.046
gr17	17	24986	0.19	24986	0.00	24986	.047
bays29	29	53990	0.701	53990	0.01	53990	1.109
dantzig42	42	42638	1.832	42638	0.01	42638	1.75
gr48	48	320277	4.216	320277	0.00	320277	3.672
hk48	48	771712	2.804	771712	0.00	771712	2.516
gr96	96	105328	52.425	105328	0.01	105328	14.250
kroA100	100	5897368	66.445	5897368	0.01	5897368	2.359
kroB100	100	5763020	94.175	5763020	0.01	5763020	2.531
kroC100	100	5890745	66.545	5890745	0.01	5890745	2.500
kroD100	100	5463250	94.155	5463250	0.03	5463250	2.547
kroE100	100	5986587	69.64	5986587	0.03	5986587	2.500

this is the first application of C-GRASP to continuous formulations of discrete optimization problems. Numerical results indicate that the procedure is able to compute optimal solutions for problems of relatively small size. However, the method becomes inefficient on problems approaching 100 nodes. The main reason for this is the fact that C-GRASP is a black-box method, in that it does not take advantage of any information about the problem structure. Recall that the only input to the method is some mechanism to compute the objective function. A natural extension of the work presented here is to *enhance* the C-GRASP framework to take advantage of the structure of the problem at hand. Using a priori information about the problem being considered, one could modify the algorithm to include these properties which would presumably reduce the required computation time.

See also

- ▶ Combinatorial Test Problems and Problem Generators
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Derivative-Free Methods for Non-smooth Optimization
- ▶ Greedy Randomized Adaptive Search Procedures
- ▶ Heuristic Search
- ▶ Integer Programming
- ▶ NP-complete Problems and Proof Methodology
- ▶ Quadratic Integer Programming: Complexity and Equivalent Forms
- ▶ Random Search Methods
- ▶ Semidefinite Programming: Optimality Conditions and Stability
- ▶ Semidefinite Programming and the Sensor Network Localization Problem, SNLP
- ▶ Solving Large Scale and Sparse Semidefinite Programs
- ▶ Variable Neighborhood Search Methods

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network Flows: Theory, Algorithms, and Applications. Prentice-Hall
2. Alon N, Kahale N (1995) Approximating the independence number via the θ-function. Technical report. Tel Aviv University, Tel Aviv, Israel
3. Balasundaram B, Butenko S (2005) Constructing test functions for global optimization using continuous formulations of graph problems. J Optim Method Softw 20(4–5): 439–452
4. Barahona F (1983) The max-cut problem in graphs is not contractible to k_5. Oper Res Lett 2:107–111
5. Barahona F, Grötschel M, Jünger M, Reinelt G (1998) An application of combinatorial optimization to statisical physics and circuit layout design. Oper Res 36:493–513
6. Benso S, Ye Y, Zhang X (2000) Solving large-scale sparse semidefinite programs for combinatorial optimization. SIAM J Optim 10:443–461
7. Blum A (1994) New approximation algorithms for graph coloring. J ACM 41(3):470–516
8. Burer S, Monteiro RDC, Zhang Y (2001) Rank-two relaxation heuristics for MAX-CUT and other binary quadratic programs. SIAM J Optim 12:503–521
9. Chang KC, Du D-Z (1987) Efficient algorithms for layer assignment problems. IEEE Trans Computer-Aided Des 6: 67–78
10. Chor B, Sudan M (1995) A geometric approach to betweenness. In: Proc. of the 3rd Annual European Symposium Algorithms. pp 227–237
11. de la Vega WF (1996) MAX-CUT has a randomized approximation scheme in dense graphs. Random Struct Algoritm 8(3):187–198
12. Deza M, Laurent M (1997) Geometry of cuts and metrics. Springer-Verlag
13. Feige U, Goemans MX (1995) Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In: Proc. of the 3rd Israel Symposium on Theory of Computing and Systems. pp 182–189
14. Feo TA, Resende MGC (1989) A probabilistic heuristic for a computationaly difficult set covering problem. Oper Res Lett 8:67–71
15. Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. J Glob Optim 6:109–133
16. Festa P, Pardalos PM, Resende MGC, Ribeiro CC (2002) Randomized heuristics for the MAX-CUT problem. Optim Method Softw 7:1033–1058
17. Festa P, Resende MGC (2002) GRASP: an annotated bibliography. In: Ribeiro C, Hansen P (eds) Essays and surveys in metaheuristics. Kluwer, pp 325–367
18. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman and Company
19. Goemans M, Williamson DP (1995) Improved approximation algorithms for MAX-CUT and satisfiability problems using semidefinite programming. J ACM 42:1115–1145
20. Grotschel M, Pulleyblank WR (1981) Weakly bipartite graphs and the max-cut problem. Oper Res Lett 1:23–27
21. Hager WW, Krylyuk Y (1999) Graph partitioning and continuous quadratic programming. SIAM J Discret Math 12:500–523

22. Helmberg C, Rendl F (2000) A spectral bundle method for semidefinite programming. SIAM J Optim 10:673–696
23. Hirsch MJ, Pardalos PM, Resende MGC (2006) Global optimization by continuous GRASP. Optim Lett (in press)
24. Homer S, Peinado M (1997) Two distributed memory parallel approximation algorithms for Max-Cut. J Parallel Distrib Comput 1:48–61
25. Kahruman-Anderoglu S, Kolotoglu E, Butenko S, Hicks IV (2007) On greedy construction heuristics for the MAX-CUT problem. Int J Comput Sci Eng (in press)
26. Karger D, Motwani R, Sudan M (1994) Approximate graph coloring by semidefinite programming. In: 35th Annual Symposium on Foundations of Computer Science. pp 2–13
27. Karp RM (1972) Reducability among combinatorial problems. In: Miller R, Thatcher J (eds) Complexity of computer computations. Plenum Press, pp 85–103
28. Lovász L (1979) On the shannon capacity of a graph. IEEE Trans Inf Theory IT-25(1):1–7
29. Lu H-I (1996) Efficient approximation algorithms for some semidefinite programs. Ph.d. dissertation, Brown University
30. Papadimitriou CH (1994) Computational complexity. Addison Wesley Longman
31. Papadimitriou CH, Yannakakis M (1991) Optimization, approximation, and complexity classes. J Comput Syst Sci 43(3):425–440
32. Pinter RY (1984) Optimal layer assignment for interconnect. J VLSI Comput Syst 1:123–137
33. Poljak S, Tuza Z (1995) The max-cut problem: a survey. In: Cook W, Lovász L, Seymour P (eds) Special year in combinatorial optimization. DIMACS Series in discrete mathematics and computer science. American Mathematical Society
34. Reinelt G (1991) TSPLIB - a traveling salesman problem library. ORSA J Comput 3:376–384
35. Resende MGC, Ribeiro CC (2003) Greedy randomized adaptive search procedures. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer, pp 219–249
36. Sahni S, Gonzalez T (1976) *P*-complete approximation problems. J ACM 23(3):555–565

Maximum Entropy and Game Theory

PANOS PARPAS, BERÇ RUSTEM
Department of Computing,
Imperial College, London, UK

Article Outline

Abstract

In decision making under uncertainty an important step is uncertainty quantification. Game theory has been traditionally used since it injects robustness into the decision process. Another popular framework is that of maximum entropy. The purpose of this article is to briefly explain the two solution concepts and point out situations in which they are identical.

Background

Consider the following optimization problem:

$$\inf_{x\in X} F(x) \triangleq \int_{\Omega} f(x,\omega)\, \mathrm{d}P(\omega)\,, \tag{1}$$

where x is the decision vector, ω is a vector representing the random parameters that are distributed according to the probability measure P. We are not concerned here with the exact properties of $f(\cdot,\cdot)$ for the problem to be valid, the interested reader is referred to [2] where the properties of this type of problem are made more precise. Instead we give two well known and studied examples of this formulation. The first is the so called two-stage recourse problem and the second is the chance constrained formulation. The former can be formulated as in Eq. (1) with the following definition for the objective function:

$$f(x_1,\omega) \triangleq f_d(x_1) + f_u(x_1,\omega)\,, \tag{2}$$

and where

$$f_u(x_1,\omega) \triangleq \inf_{x_2(\omega)} \{f_2(x_1,x_2(\omega),\omega) \mid x_2(\omega)\in X_2(\omega,x_1)\}\,. \tag{3}$$

In Eq. (2) the objective function is split into two parts, the deterministic (f_d) and the uncertain part (f_u) of the problem . The decision to be taken is x_1. The full consequences of following a particular strategy are not known exactly since the true cost will depend on the solution of Eq. (3). The objective is therefore to find the decision x_1 that is best on average.

A different decision model is given by chance constrained programming problems. These can be formulated as follows:

$$\inf_x \{f(x) \mid \Pr(g(x,\omega) \geq 0) \geq \alpha\} \ ,$$

where we optimize an objective function and impose constraints that need to be satisfied with a probability of above a certain threshold α.

These two models have been widely studied and have found many applications where traditional optimization is used. It is also evident that their usefulness revolves around our ability to provide a reasonable description of the uncertainties.

In order to provide a description of the uncertainties a technique based on moment matching can be used. Under this framework we assume that the decision maker can not provide an exact description of the distribution but only knows some of its moments. The problem is to recover a meaningful probability measure given this knowledge: suppose a vector of functions $m(\omega) = [m_1(\omega) \dots m_n(\omega)]$ and a vector of scalars $\mu = [\mu_1 \dots \mu_n]$ are given, the problem is to find a P such that:

$$\begin{aligned} \int_\Omega m_i(\omega)\,\mathrm{d}P(\omega) = \mu_i, \quad i = 1,\dots,n \\ \int_\Omega \mathrm{d}P(\omega) = 1 \ , \\ P(\omega) \geq 0 \ , \quad \text{a.e} \ , \end{aligned} \tag{4}$$

where Ω is a compact subset of $\mathbb{R}^m$. By $\mathcal{P}$ we will denote the set of of all finite signed measures that are defined on the σ-field $\mathcal{F}$ of Ω. The vector $m(\omega)$ represents the (generalized) moments of the distribution. The problem in Eq. (4) is the so called generalized Hausdorff moment problem. The aim is to recover a compactly supported distribution from a finite number of its general, not necessarily power, moments. This is a variation of the classical moment problem formulated by Stieltjes. In [4] one can find a comprehensive summary of the main results when $\Omega = [0,1]$. Prekopa [13] provides an excellent summary of results that are especially relevant in stochastic optimization problems.

Methods

Solving optimization problems where the uncertainty is only known through its moments requires some kind of regularization in order to fix the probability measure with which the optimization is to be done. Two popular frameworks are game theoretic and maximum entropy approaches. Under the game theory framework one selects the distribution with the worst case realization of the uncertainties. When a maximum entropy solution is sought, one optimizes with respect to the distribution with the maximum uncertainty.

Game Theory Approach

When P is unknown or not known exactly then the decision maker assumes that if strategy x is followed then the consequences of following this strategy will be decided by some law of Nature. Motivated by the application oriented requirement for robust decision making, we assume that Nature is antagonistic. If we decide to follow strategy x then Nature will follow strategy P^*. The latter is the solution of the following optimization problem:

$$\Phi(x) = \sup_{P \in \mathcal{P}} \int_\Omega f(x,\omega)\,\mathrm{d}P(\omega) \ , \tag{5}$$

where $\Phi(x)$ represents the value (outcome) of the game if strategy x is followed. Obviously $\Phi(x) \geq F(x)$ for given $x \in X$ and for all $P \in \mathcal{P}$. Therefore, after we minimize Eq. (5) for x we will be guaranteed to attain a value which is as good as $\Phi(x)$ whatever strategy nature decides to follow. The robustness property of the minimax strategy originates from the latter property. We thus reformulate Eq. (1) as follows:

$$\inf_{x \in X} \sup_{P \in \mathcal{P}} \int_\Omega f(x,\omega)\,\mathrm{d}P(\omega) \ . \tag{6}$$

This approach has its origins in game theory and has been used extensively in many areas of optimization. See for example [9] for an excellent introduction to game theory, applications of minimax especially in economics and finance can be found in [14]. Numerical algorithms to solve Eq. (6) have been proposed in [5,6]. The general idea of these algorithms is to solve the inner maximization problem using results for general Chebyshev inequalities [4,17]. However these methods require several global optimization steps to be performed at each iteration in order to identify the support of the measure that maximizes Eq. (5). Moreover it is usually assumed that the max-function is convex. Algorithms based on stochastic quasi-gradient methods were proposed in [4,17]. Recently Shapiro et al. [15] proposed the use of a reference distribution in order to reformulate the minimax problem into a standard stochastic programming problem. Bertsimas et al. [1] and Lassere [11] proposed a semidefinite formulation of the inner maximization problem in Eq. (6).

Maximum Entropy Formulation

The formulation of the moment problem using the maximum entropy principle was initiated by Jaynes [8]. The derivations in this Section are more or less standard (see e. g. [12]).We will assume that a multivariate continuous density is postulated, discrete distributions share similar properties. Under these assumptions the maxent formulation is given by:

$$\begin{aligned} \inf_{p\in\mathcal{P}_c}\ Q(p,h) &= \int_\Omega p(\omega)\ln\frac{p(\omega)}{h(\omega)}\,\mathrm{d}\omega \\ \text{s.t} \int_\Omega m_i(\omega)p(\omega)\,\mathrm{d}\omega &= \mu_i \quad i=0\ldots n\,, \end{aligned} \tag{7}$$

where $\mathcal{P}_c$ denotes the restriction on $\mathcal{P}$ to all absolutely continuous measures w.r.t $\mathrm{d}\omega$, we will write $\nu \ll \mu$ to mean that ν is absolutely continuous w.r.t μ. The function in the objective function is the so called Kullback Leibler divergence (see e. g. [3]) and serves as a kind of distance metric between $h(\omega)$ and $p(\omega)$; the former is a distribution that is assumed to be known. The objective is to find a p.d.f with the prescribed moments that is as close to h as possible. If such a function is not known then we take $h(\omega) \equiv 1$ (i. e. the uniform distribution) and the problem becomes the classical maximum entropy formulation. The convex functional defined by $Q(p,h)$ is always strictly positive and is zero if and only if $p = h$ a.e. It is also worth mentioning that in general $Q(p,h) \neq Q(h,p)$. These properties of $Q(p,h)$ are well known. We refer the interested reader to [3] for more properties of the entropy function. We assume that $m_0(\omega) = \mu_0 = 1$. Note that by considering general moments as opposed to power moments allows us to impose fractile constraints, this property is important in many applications.

While the problem in Eq. (7) is a convex optimization problem it cannot be handled using standard numerical algorithms. For this reason one considers the dual of Eq. (7). The Lagrangian associated with Eq. (7) is given by:

$$\begin{aligned} L(p,\lambda) = &\int_\Omega p(\omega)\ln\frac{p(\omega)}{h(\omega)}\,\mathrm{d}\omega \\ &+ \sum_{i=0}^{n}\lambda_i\left(\int_\Omega m_i(\omega)p(\omega)\,\mathrm{d}\omega - \mu_i\right)\,. \end{aligned}$$

The dual problem of Eq. (7) is given by:

$$\sup_\lambda D(\lambda) = \inf_{p\in\mathcal{P}_c} L(p,\lambda)\,. \tag{8}$$

It is well known that the inner minimization on Eq. (8) can be done explicitly using functional derivatives [12,16]:

$$\begin{aligned} &L(p+\delta p,\lambda) \\ &= \int_\Omega (p(\omega)+\delta p(\omega))\ln\left\{\frac{p(\omega)}{h(\omega)}\left(1+\frac{\delta p(\omega)}{p(\omega)}\right)\right\}\,\mathrm{d}\omega \\ &\quad + \sum_{i=0}^{n}\lambda_i\left(\int_\Omega m_i(\omega)(p(\omega)+\delta p(\omega))\,\mathrm{d}\omega - \mu_i\right) \\ &= L(p,\lambda) + \int_\Omega\left\{1+\ln\left(\frac{p(\omega)}{h(\omega)}\right)\right\}\delta p(\omega)\,\mathrm{d}\omega \\ &\quad + \sum_{i=0}^{n}\int_\Omega \lambda_i m_i(\omega)\delta p(\omega)\,\mathrm{d}\omega\,, \end{aligned}$$

where to get the last equality we assumed that δp is small, used the approximation $\ln(1+\epsilon) \approx \epsilon$ (which is valid for small ϵ) and ignored second order terms. The stationary points of the Lagrangian must therefore satisfy:

$$p(\omega) = h(\omega)\exp\left(-1-\sum_{i=0}^{n}\lambda_i m_i(\omega)\right)\,. \tag{9}$$

Using the normalization condition we have:

$$\mathcal{Z} = \exp(1+\lambda_0) = \int_\Omega h(\omega)\exp\left(-\sum_{i=1}^{n}\lambda_i m_i(\omega)\right)\,\mathrm{d}\omega.$$

Using the equation above we can write Eq. (9) as follows:

$$p(\omega) = \frac{h(\omega)}{\mathcal{Z}}\exp\left(-\sum_{i=1}^{n}\lambda_i m_i(\omega)\right)\,. \tag{10}$$

Finally, using Eq. (10) and the normalization constraint in Eq. (8) we find the following explicit form for the dual problem:

$$\sup_{\lambda} D(\lambda) = -\ln \mathcal{Z} - \sum_{i=1}^{n} \lambda_i \mu_i \,. \tag{11}$$

The dual formulation given above is more useful than the primal problem since the dual problem is amenable to conventional optimization algorithms.

Relationships Between Game Theory and Maximum Entropy

The minimax approach has proven to be a prudent method for problems where the nature of the uncertainty is not known exactly. We will approach the problem somewhat differently by dispensing the usual assumptions of convexity but allowing the decision maker to adopt mixed strategies. Such an approach (in the context of Stochastic Programming) has been described in Kolbin [10] but has not received much attention. The advantage of allowing mixed strategies is that the problem exhibits a saddle point. Topsøe [18] showed that if the decision problem has a specific structure (will be outlined below) then the solution of the maximum entropy problem and that of zero-sum games are dual to each other. Recently Grünwald et al. [7] has further developed this approach so that it can be applied to more general games. This generalization however has been done at the expense of defining more general entropy functionals; these do not, in general, render themselves to numerical algorithms. We believe this relationship to be very interesting and can under certain conditions be used as an additional motivation for adopting the maximum entropy principle.

Let Ω be a compact subset of $\mathbb{R}^n$ and let $\mathcal{F}$ be the σ-field generated by Ω. We will use ω to denote a random vector whose distribution is known to belong to a family $\mathcal{P}$. The following meaningless formulation of a stochastic programming problem:

$$\inf_{x \in X} f(x, \omega)$$
$$\text{s.t } g_i(x, \omega) \le 0 \,, \quad i = 1, \dots, k$$

can be placed into a pertinent form by formulating it as a two-person zero sum game $G = (x, \omega, q)$. The first player is the Decision Maker (DM) that selects vectors $x \in X \subset \mathbb{R}^m$. The second player is Nature that selects an event $\omega \in \mathcal{F}$ with probability $P(\omega)$, it is further assumed that the exact probability measure of Nature is only known to belong to a certain family $\mathcal{P}$. The function q represents the outcome of the game given the strategies the two players decide to follow. Kolbin [10] suggested the following form:

$$q(x, \omega) = f(x, \omega) + \sum_{i=1}^{k} \beta_i (g_i(x, \omega)) \,, \tag{12}$$

where $\beta_i(a)$ is a continuous non-decreasing penalty function that is 0 when $a \le 0$. An example of such a function is the max-penalty function given by: $c_i \max\{g_i(x, \omega), 0\}$ (c_i is a penalty parameter).

The DM would like to minimize the outcome of the game given by Eq. (12) whereas Nature being antagonistic would like to maximize this quantity:

$$\inf_{x \in X} \sup_{P \in \mathcal{P}} H(x, P) = \int_{\Omega} q(x, \omega) \, \mathrm{d}P(\omega) \,. \tag{13}$$

For the game above to exhibit a saddle point convexity assumptions need to be imposed on q. Many problems of interest do not have this property and it is necessary to resort to mixed strategies for the DM. Using our assumptions that q is continuous, and the compactness of Ω and X, it can be shown [10] that if we allow the DM to follow mixed strategies then the game in Eq. (13) will have a saddle point, i.e:

$$\begin{aligned} &\inf_{K \in \mathcal{K}} \sup_{P \in \mathcal{P}} \int_{\Omega \times X} q(x, \omega) \, \mathrm{d}P(\omega) \, \mathrm{d}K(x) \\ &\quad = \sup_{P \in \mathcal{P}} \inf_{K \in \mathcal{K}} \int_{\Omega \times X} q(x, \omega) \, \mathrm{d}P(\omega) \, \mathrm{d}K(x) \\ &\quad = H(K^*, P^*) \,, \end{aligned} \tag{14}$$

where the set $\mathcal{K}$ represents the family of randomized strategies of the DM.

Assume that the DM selects a probability measure $K \in \mathcal{K}$ and Nature selects $P \in \mathcal{P}$ and both have their support in Ω (or X). Moreover assume that the objective function of the game has the following functional form:

$$H(K, P) = \int_{\Omega} -p(\omega) \ln k(\omega) \, \mathrm{d}\omega \,, \tag{15}$$

where $p(\omega)$ and $k(\omega)$ are the Radon–Nikodym derivatives w.r.t $\mathrm{d}\omega$ of $P \in \mathcal{P}$ and $K \in \mathcal{K}$ respectively.

Topsøe [18] observed that under these conditions the maximum entropy solution and the minimax solution coincide. To see why this is the case, suppose that nature selects a probability measure from the following family:

$$\mathcal{P}_c = \left\{ \frac{\mathrm{d}P}{\mathrm{d}\omega} \mid P \in \mathcal{P}, \int_\Omega m_i(\omega)\, \mathrm{d}P(\omega) = \mu_i, \right. \\ \left. i = 1, \ldots, n \ \ P \ll \ \mathrm{d}\omega \right\} ,$$

where $P \ll \ \mathrm{d}\omega$ is used to denote that P is absolutely continuous w.r.t to $\mathrm{d}\omega$. $\mathcal{K}_c$, the family of admissible strategies for the DM is defined in an analogous manner. If Nature adopts a maximum entropy distribution, then its strategy can be found by solving:

$$\sup_{p \in \mathcal{P}_c} M(p) = \int_\Omega -p(\omega) \ln p(\omega)\, \mathrm{d}\omega \,.$$

The optimal solution will be given by:

$$p^*(\omega) = \exp\{-1 - \lambda_0^* - \sum_{i=1}^n \lambda_i^* m_i(\omega)\} \,.$$

The optimal strategy of the DM can then be obtained by solving:

$$\inf_{k \in \mathcal{K}_c} \int_\Omega -p^*(\omega) \ln k(\omega)\, \mathrm{d}\omega \,.$$

Using the information inequality [3] we have:

$$\int_\Omega -p^*(\omega) \ln k(\omega)\, \mathrm{d}\omega \ \geq \int_\Omega -p^*(\omega) \ln p^*(\omega)\, \mathrm{d}\omega \,, \tag{16}$$

the above inequality is satisfied as an inequality if and only if $k(\omega) = p^*(\omega)$ a.e. Consequently the optimal strategy for the DM is the same as Nature's strategy.

Conversely, assuming that the game has the functional form given in Eq. (15), then the minimax solution of the game in Eq. (14) is the same as the maximum entropy solution. Indeed, by using the information inequality we have:

$$\sup_{p \in \mathcal{P}_c} \int_\Omega -p(\omega) \ln k(\omega)\, \mathrm{d}\omega \\ \geq \int_\Omega -p^*(\omega) \ln k(\omega)\, \mathrm{d}\omega \\ \geq \int_\Omega -p^*(\omega) \ln p^*(\omega)\, \mathrm{d}\omega = M(p^*) \,,$$

where p^* is the distribution of maximum entropy. From the above relationship it follows that:

$$M(p^*) \leq \inf_{k \in \mathcal{K}_c} \sup_{p \in \mathcal{P}_c} \int_\Omega -p(\omega) \ln k(\omega)\, \mathrm{d}\omega \,, \tag{17}$$

if we choose $k = p^*$ as the minimizer of the left hand side of Eq. (14), then:

$$\sup_{p \in \mathcal{P}_c} \int_\Omega -p(\omega) \ln p^*(\omega)\, \mathrm{d}\omega = -\lambda_0^* - \sum_{i=1}^n \lambda_i^* \mu_i \\ = M(p^*) \,,$$

it follows from above that:

$$\inf_{k \in \mathcal{K}} \sup_{p \in \mathcal{P}_c} \int_\Omega -p(\omega) \ln k(\omega)\, \mathrm{d}\omega \leq M(p^*) \,, \tag{18}$$

and therefore $k = p^*$ is indeed the minimizer of the left hand side of Eq. (14). From the well known property of minimax problems:

$$M(p^*) = \inf_{k \in \mathcal{K}} \sup_{p \in \mathcal{P}_c} \int_\Omega -p(\omega) \ln k(\omega)\, \mathrm{d}\omega \\ \geq \sup_{p \in \mathcal{P}_c} \inf_{k \in \mathcal{K}} \int_\Omega -p(\omega) \ln k(\omega)\, \mathrm{d}\omega \,,$$

and from:

$$\sup_{p \in \mathcal{P}_c} \inf_{k \in \mathcal{K}} \int_\Omega -p(\omega) \ln k(\omega)\, \mathrm{d}\omega \\ \geq \inf_{k \in \mathcal{K}} \int_\Omega -p^*(\omega) \ln k(\omega)\, \mathrm{d}\omega = M(p^*) \,,$$

we conclude that the game has a saddle point at $p = k = p^*$.

For games in the form of Eq. (15), the relationship between game theory and maximum entropy is most useful both theoretically and practically. For games not in the form described above can still be approached via maximum entropy methods but the definition of the entropy functional is given by a more general functional form. Grünwald et al. [7] defined the generalized entropy function as:

$$M(P) = \inf_{k \in \mathcal{K}} \int_{\Omega \times X} q(x, \omega)\, \mathrm{d}P(\omega)\, \mathrm{d}K(x) \,.$$

The maximum entropy problem becomes:

$$\max_{p \in \mathcal{P}_c} M(P) \,. \tag{19}$$

They showed that using the generalized definition of entropy one could find the same results for both game theory and maximum entropy problems. Even though the formulation in Eq. (19) is very general, unfortunately there is no general way to solve it. However, the relationship between the two principles is worth further investigation.

References

1. Bertsimas D, Sethuraman J (2000) Moment problems and semidefinite optimization. In: Handbook of semidefinite programming. Internat Ser Oper Res Manag Sci, vol 27. Kluwer, Boston, pp 469–509
2. Birge JR, Louveaux F (1997) Introduction to stochastic programming. Springer Series in Operations Research. Springer, New York
3. Cover TM, Thomas JA (1991) Elements of information theory. Wiley Series in Telecommunications. A Wiley-Interscience Publication. Wiley, New York
4. Dette H, Studden WJ (1997) The theory of canonical moments with applications in statistics, probability, and analysis. Wiley Series in Probability and Statistics: Applied Probability and Statistics. A Wiley-Interscience Publication. Wiley, New York
5. Ermoliev Y, Gaivoronski A, Nedeva C (1985) Stochastic optimization problems with incomplete information on distribution functions. SIAM J Control Optim 23(5):697–716
6. Gaivoronski A (1991) A numerical method for solving stochastic programming problems with moment constraints on a distribution function. Stochastic programming, Part II (Ann Arbor, MI, 1989). Ann Oper Res 31(1–4):347–369
7. Grünwald PD, Dawid AP (2004) Game theory, maximum entropy, minimum discrepancy and robust Bayesian decision theory. Ann Statist 32(4):1367–1433
8. Jaynes ET (1957) Information theory and statistical mechanics I. Phys Rev 106:620–630
9. Karlin S (1992) Mathematical methods and theory in games, programming, and economics. vol I: Matrix games, programming, and mathematical economics, vol II: The theory of infinite games, Reprint of the 1959 original. Dover, New York
10. Kolbin VV (1970) Stochastic programming. In: Theory of Probability. Mathematical Statistics. Theoretical Cybernetics. 1968 (Russian). Akad Nauk SSSR Vsesojuz Inst Naučn i Tehn Informacii, Moscow, pp 5–68
11. Lasserre JB (2002) Bounds on measures satisfying moment conditions. Ann Appl Probab 12(3):1114–1137
12. Mead LR, Papanicolaou N (1984) Maximum entropy in the problem of moments. J Math Phys 25(8):2404–2417
13. Prékopa A (1995) Stochastic programming. Mathematics and its Applications, vol 324. Kluwer, Dordrecht
14. Rustem B, Howe M (2002) Algorithms for worst-case design and applications to risk management. Princeton University Press, Princeton, NJ
15. Shapiro A, Ahmed S (2004) On a class of minimax stochastic programs. SIAM J Optim (electronic) 14(4):1237–1249
16. Smith DR (1998) Variational methods in optimization. Reprint of the 1974 original. Dover, Mineola, NY
17. Smith JE (1995) Generalized Chebychev inequalities: theory and applications in decision analysis. Oper Res 43(5):807–825
18. Topsøe F (1979) Information-theoretical optimization techniques. Kybernetika 15:8–27

Maximum Entropy Principle: Image Reconstruction

Entropy Optimization for Image Reconstruction

SHU-CHERNG FANG[1], JACOB H.-S. TSAO[2]
[1] North Carolina State University, Raleigh, USA
[2] San Jose State University, San Jose, USA

MSC2000: 94A17, 94A08

Article Outline

Keywords

Entropy optimization; Image reconstruction; Maximum entropy principle; Principle of maximum entropy

Images can be used to characterize the underlying distribution of certain physical properties, such as density, shape, and brightness, of an object under investigation. In many applications where an image is required, only a finite number of observations and/or indirect measurements can be made. Image reconstruction is a procedure for processing the measurement data to construct an image of the object. This section introduces the basic concept of *image reconstruction from projection data*. Two types of entropy optimization models, namely, the finite-dimensional model and vector-space model, and three classes of entropy optimization methodologies, namely, the discretization methods, Banach-space methods (e. g., MENT) and Hilbert-space methods (e. g., *finite element method*) are included. For more details about image reconstruction, the reader is referred to [2,7,13] and the references therein.

A very important scientific application of image reconstruction is in *computerized tomography* (CT) for medical diagnosis. Physicians need to know, for example, the location, shape, and size of a suspected tumor inside a patient's brain in order to plan a suitable course of treatment. With computerized tomography, images of cross-sections of a human body can be constructed from data obtained by measuring the attenuation of X-rays along a large number of straight lines (or strips) through each cross-section. For ease of introduction, we illustrate the basic ideas about image reconstruction with the example of two-dimensional X-ray CT, with the understanding that the discussion can be generalized to higher-dimensional settings.

In this example, the distribution to be determined is that of the X-ray linear attenuation coefficient of human body tissues. The total attenuation of the X-ray beam between a source and a detector is approximately the integral of the linear attenuation coefficient along the line between the source and the detector. The unknown distribution of the X-ray linear attenuation coefficient is represented by a density function f of two variables, which assumes zero-value outside a squared-shape region. The squared region is usually referred to as the support of the image.

Two basic types of entropy optimization models, namely, *finite-dimensional model* and *vector-space model*, are commonly used to decide the density function f. The finite-dimensional models approximate the density values over the support of the image at a finite number of grid points, while the density is approximated by a real-value function for the entire scanning region in the vector-space models. The latter models were motivated to reconstruct the image with only a small number of available projections.

In the finite-dimensional models, the support of the density f is represented by n (given by the users) regularly spaced grid points, and the values of the density function f at these points are denoted by $\mathbf{f} \equiv (f_1, \ldots, f_n)$. Assume that m projections are made and the measurement data $\mathbf{d} \equiv (d_1, \ldots, d_m)$ are obtained.

The relationship between the unknown density values $\mathbf{f}$ and the observed measurement $\mathbf{d}$ can be approximated by a linear relation

$$\mathbf{d} \approx \mathbf{A}\mathbf{f}, \tag{1}$$

where $\mathbf{A} = [a_{ij}]$ is a projection matrix.

Note that the approximation sign in (1) reflects possible errors in modeling and measurement. Also note that, in the classical square pixel model, the image is discretized by partitioning its support into a finite number of equi-sized square regions (called pixels or cells) whose centers are those n sample points. By assuming that the density function f is constant in each of the equi-sized pixels, i. e., $f = f_j$ throughout pixel j, the value of a_{ij} in the projection matrix is simply the length of the intersection of the line corresponding to the ith projection with the pixel surrounding the jth sample point.

Once the projection matrix $\mathbf{A}$ is defined and the measurement $\mathbf{d}$ is known, the problem is to find an $\mathbf{f}$ satisfying (1). To cope with the errors mentioned above, G.T. Herman [6] suggested that (1) be replaced by an 'interval constraint' and a nonnegativity constraint be added:

$$\mathbf{d} - \boldsymbol{\epsilon} \leq \mathbf{A}\mathbf{f} \leq \mathbf{d} + \boldsymbol{\epsilon}, \tag{2}$$

$$\mathbf{f} \geq \mathbf{O}, \tag{3}$$

where $\epsilon = (\epsilon_1, \ldots, \epsilon_m)$ is an m-vector of user-chosen tolerance levels. Note that (2) can be replaced by an equivalent system of inequalities

$$\mathbf{A}'\mathbf{f} \leq \mathbf{d}', \tag{4}$$

with twice as many one-sided inequalities [2,6].

For such an image reconstruction model, we can adopt either the *'feasibility approach'* to find a solution to (2) and (3) directly, or the *'optimization approach'* to find a solution that is not only feasible in the above sense but also optimal with respect to a certain criterion. In the literature, at least three different types of optimization problems have been proposed, namely, the entropy maximization problem, the quadratic minimization problem, and the maximum likelihood problem.

The entropy optimization problem seeks to optimize an entropic objective function subject to (2) and (3) as follows.

Model 1:

$$\begin{cases} \max & -\sum_{j=1}^{n} f_j \ln f_j \\ \text{s.t.} & \mathbf{d}-\boldsymbol{\epsilon} \le \mathbf{A}\mathbf{f} \le \mathbf{d}+\boldsymbol{\epsilon}, \\ & \mathbf{f} \ge \mathbf{O}. \end{cases} \tag{5}$$

Some researchers proposed models in which the f_j's are normalized in such a way that $\sum_{j=1}^{n} f_j = 1$, and the projection matrix and the measurement data differ from those of Model 1. See, e. g., [4]. In this way, a solution that is consistent with the measurement data but remains maximally noncommittal can be found. Note that an optimal solution to such models can also be interpreted as the most probable solution that is consistent with the measurement data [3].

Other variations of Model 1 exist. Despite possible modeling and measurement errors, one common practice is to replace (1) and inequalities (2), and (5) by a system of equations: $\mathbf{Af} = \mathbf{d}$.

A different version of the finite-dimensional entropy optimization model begins with the definition of an error vector $\mathbf{e} = (e_1, \ldots, e_m)^\mathsf{T}$, where

$$e_i \equiv d_i - \sum_{j=1}^{n} a_{ij} f_j, \quad i = 1, \ldots, m.$$

Assume that errors $e_1, \ldots, e_m$ exist due to imprecise measurement and are independent noise terms with zero mean and known variance σ_i^2. S.F. Burch et al. [1] observed that the strong law of large numbers implies that

$$Q(\mathbf{f}) \equiv \frac{1}{m} \sum_{i=1}^{m} \frac{\left(\sum_{j=1}^{n} a_{ij} f_j - d_i\right)^2}{\sigma_i^2} \to 1,$$
$$\text{as} \quad m \to \infty.$$

Thus, if m is sufficiently large, the following entropy optimization problem with quadratic constraints can be useful:

Model 2:

$$\begin{cases} \max & -\sum_{j=1}^{n} f_j \ln f_j \\ \text{s.t.} & \frac{1}{m}(\mathbf{Af}-\mathbf{d})^\top \mathbf{S}^2 (\mathbf{Af}-\mathbf{d}) = 1, \\ & f_j \ge 0, \quad j = 1, \ldots, n, \end{cases}$$

where $\mathbf{S}$ is a diagonal matrix with $1/\sigma_i$ being its ith diagonal element.

Concerns such as the smoothing effect, nonuniformity, peakness, and exactness [14] of a constructed image can also be addressed in this model with proper modification of the objective functions and constraints. So far, we have used the square pixel model to illustrate the idea of entropy optimization for image reconstruction. Other models exist [2].

For an introduction to the concept of *Shannon's entropy* and related entropy optimization principles, i. e., *principle of maximum entropy* and *principle of minimum cross-entropy*, see ► Entropy optimization: Shannon measure of entropy and its properties. A large amount of literature has been devoted to developing iterative methods for solving finite-dimensional *entropy optimization* problems with linear and/or quadratic constraints. For details and a unification of such methods, see [3].

The method currently employed in most CT systems is the 'filtered back-projection' method, which is based on a finite-dimensional model. (See [5,10] for details.) Compared to the iterative methods for solving entropy optimization problems, this method provides speed, which enables reconstruction of the image while X-ray transmission data are being collected. Hence the time between scanning and obtaining reconstructed images is reduced. However, there are situations where iterative methods produce comparable or better reconstructed images than the filtered back-projection method, e. g., in image reconstruction with few projections or in high-contrast image reconstruction. The ever increasing computer speed and its companion reduction in cost may increase the desirability of employing iterative methods in CT systems.

In many situations, e. g., conducting diagnostic experiments on plasma in magnetic confinement devices or laser target impositions with measurements on fusion reactor cores, only few projections are available, e. g., less than 10. When the finite-dimensional entropy optimization model is applied, it tends to produce 'streaking' artifacts. This motivated the use of the vector-space model.

Take the two-dimensional X-ray CT problem as an example. By assuming that the unknown density function $f(x, y)$ is continuous over a compact support D

such that

$$f(x,y) \geq 0 \text{ and } \int\int_D f(x,y)\,dx\,dy = 1, \qquad (6)$$

G. Minerbo [9] defined the entropy of $f(x, y)$ as

$$\zeta(f) = -\int\int_D f(x,y)\ln[f(x,y)A]\,dx\,dy,$$

where A is the area of D. Denote the set of continuous, nonnegative functions with compact support in D by $C_+(D)$.

The scanning area is partitioned into parallel strips, each of which is penetrated by an X-ray beam. Let θ_j, $j = 1, \dots, J$, be the J distinct projection angles with respect to the X-axis of the scanning area. Also let $M(j)$ be the number of parallel beams associated with the jth projection or view, and $S_{j1} < \cdots < S_{jM(j)}$ be a set of abscissas for the jth view. The projection data are assumed to be in the form of the following 'strip integrals':

$$P_{jm}(f) \equiv \int_{S_{jm}}^{S_{j(m+1)}} \int_{-\infty}^{\infty} f(s\cos\theta_j - t\sin\theta_j, s\sin\theta_j + t\cos\theta_j)\,dt\,ds,$$

where $m = 1, \dots, M(j)$ and $j = 1, \dots, J$. It is assumed that, for $j = 1, \dots, J$,

$$\int_{-\infty}^{\infty} f(s\cos\theta_j - t\sin\theta_j, s\sin\theta_j + t\cos\theta_j)\,dt = 0, \quad \text{for} \quad s < S_{j1} \quad \text{or} \quad s > S_{jM(j)}.$$

Let G_{jm} denote the observed values of $P_{jm}(f)$, for $m = 1, \dots, M(j)$, and $j = 1, \dots, J$. Note that (6) implies $G_{jm} \geq 0$ and $\sum_{m=1}^{M(j)} G_{jm} = 1$.

Then the vector-space model results in the following *optimization problem*:

Model 3:

$$\begin{cases} \sup\limits_{C_+(D)} & \zeta(f) \\ \text{s.t.} & P_{jm}(f) = G_{jm}, \\ & \quad m = 1, \dots, M(j); \\ & \quad j = 1, \dots, J. \end{cases} \qquad (7)$$

A finite-dimensional unconstrained dual problem can be derived by using the technique of Lagrange multipliers. An algorithm known as MENT [9] was proposed. It was shown that the solutions produced by MENT converge to a density function f^* which satisfies the constraint (7) with $\zeta(f^*) = \sup_{C_+(D)} \zeta(f)$. However, the limiting density function f^* is not continuous. Actually, as pointed out in [8], f^* is piecewise constant and $f^* \notin C_+(D)$. When few projections are available and the object being scanned has a simple structure (or close to circular symmetry in density), some preliminary computational results indicated the potential of this approach.

Recognizing the fact that the supremum of Model 3 is not attained by any function $f \in C_+(D)$, M. Klaus and R.T. Smith [8] defined an alternative formulation in a richer class of functions than $C_+(D)$. More precisely, they replaced $C_+(D)$ by $L^2_+(D)$, the set of all nonnegative square integrable functions on D, as the setting. Note that all piecewise-constant functions over D are contained in $L^2_+(D)$. Also recognizing that measurements may not be consistent and even be flawed, they considered an optimization problem where the objective function is the original entropy functional $\zeta(f)$ minus a penalty term corresponding to the residual error in meeting the measurement constraints, and the constraint is that the maximizer lies in a weakly compact set that is determined by known physical information about the density function of the object to be scanned. A corresponding formulation becomes

Model 4:

$$\sup_{f\in\Omega} G(f) \equiv \zeta(f) - \gamma \sum_{j,m} [G_{jm} - P_{jm}(f)]^2,$$

where $\gamma > 0$ is an adjustable penalty parameter and Ω is a convex and weakly (sequentially) compact set of nonnegative functions in $L^2_+(D)$, with a compact support in D and containing physical information known *a priori* about the object to be scanned, e. g., upper and lower bounds on the density function. (A set Ω of nonnegative functions in $L^2_+(D)$ is weakly (sequentially) compact if and only if every sequence in Ω has a weakly convergent subsequence whose weak limit lies in Ω; a sequence $\{f_n(x, y)\}$ converges weakly to $f(x, y)$ if and only if the sequence $\{\langle f_n(x, y), g(x, y)\rangle\}$ converges to $\langle f(x, y), g(x, y)\rangle$ for every $g(x, y) \in L^2_+(D)$, where $\langle h_1, h_2\rangle \equiv \int\int h_1(x, y)\, h_2(x, y)\, dxdy$ denotes the inner product of h_1 and h_2 in the space of $L^2_+(D)$.)

With the aid of the theory of Hilbert space, it can be shown [8] that G has a unique maximizer in Ω, for any given data G_{jm}, $m = 1, \dots, M(j)$, $j = 1, \dots, J$.

Based on this alternative formulation, the density function $f(x, y)$ can be approximated by using the *finite element method* [11]. For simplicity, assume that $D = [-1, 1] \times [-1, 1]$. First, we superimpose a fixed rectangular mesh on D, with uniform mesh size $h = 1/n$ in both the x and y directions. We also use the product of piecewise linear functions in x and y as the finite element space S^h. In this way, a basis for S^h has the form

$$\psi_k(x, y) = \psi_i(x)\psi_l(y), \text{for} \quad k = 1, \dots, (2n+1)^2,$$

where

$$l = \left\{ \frac{(k-1) - (k-1) \pmod{2n+1}}{2n+1} \right\} - n,$$
$$i = k - (l+n)(2n+1) - n - 1,$$

and

$$\psi_j(t) = \begin{cases} 0 & \text{if} \quad t \le (j-1)h \\ & \quad \text{or} \quad t \ge (j+1)h, \\ \frac{t-(j-1)h}{h}, & \text{if} \quad (j-1)h \le t \le jh, \\ \frac{(j+1)h-t}{h}, & \text{if} \quad jh \le t \le (j+1)h. \end{cases}$$

It is reasonable to expect that, in practice, one should know a priori the minimum and maximum densities of the object being examined. Hence we focus on a simple constraint set

$$\Omega = \left\{ f \in L^2_+(D) \colon \begin{array}{c} 0 < a \le f \le b < \infty \text{ a.e.,} \\ f = 0 \text{ a.e., in } \mathbb{R}^2 \setminus D \end{array} \right\}.$$

The density function $f(x, y)$ is then approximated in S^h by

$$f(x, y) = \sum_{k=1}^{N} c_k \psi_k(x, y),$$

where $N = (2n+1)^2$ and c_k's are chosen as the optimal solution of the following finite-dimensional optimization problem:

$$\begin{cases} \sup\limits_{c \in \mathbb{R}^N} & G\left(\sum\limits_{k=1}^{N} c_k \psi_k(x, y) \right) \\ \text{s.t.} & 0 < a \le \sum\limits_{k=1}^{N} c_k \psi_k(x, y) \le b. \end{cases}$$

This problem can be further reduced to

$$\begin{cases} \sup_{c \in \mathbb{R}^N} & \zeta\left(\sum\limits_{k=1}^{N} c_k \psi_k(x, y) \right) \\ & -\gamma \sum\limits_{j,m} \left[G_{jm} - \sum\limits_{k=1}^{N} c_k P_{jm}(\psi_k(x, y)) \right]^2 \\ \text{s.t.} & 0 < a \le c_k \le b, \quad k = 1, \dots, N. \end{cases}$$

Preliminary computational results reported in [11, 12] indicate some improvements of this alternative approach over the MENT algorithm when the object under investigation does not have circular symmetry in density and has a high density area near the edge of the scanning region.

See also

- ▶ Entropy Optimization: Interior Point Methods
- ▶ Entropy Optimization: Parameter Estimation
- ▶ Entropy Optimization: Shannon Measure of Entropy and its Properties
- ▶ Jaynes' Maximum Entropy Principle
- ▶ Optimization in Medical Imaging

References

1. Burch SF, Gull SF, Skilling JK (1983) Image restoration by a powerful maximum entropy method. Computer Vision, Graphics, and Image Processing 23:113–128
2. Censor Y, Herman GT (1987) On some optimization techniques in image reconstruction. Applied Numer Math 3:365–391
3. Fang S-C, Rajasekera JR, Tsao H-SJ (1997) Entropy optimization and mathematical programming. Kluwer, Dordrecht
4. Frieden BR (1972) Restoring with maximum likelihood and maximum entropy. J Optical Soc Amer 62:511–518
5. Hendee WR (1983) The physical principles of computed tomography. Little, Brown and Company, Boston, MA
6. Herman GT (1975) A relaxation method for reconstructing objects from noisy X-rays. Math Program 8:1–19
7. Herman GT (ed) (1979) Image reconstruction from projections: implementation and applications. Springer, Berlin
8. Klaus M, Smith RT (1988) A Hilbert space approach to maximum entropy reconstruction. Math Meth Appl Sci 10:397–406
9. Minerbo G (1979) MENT: A maximum entropy algorithm for reconstructing a source from projection data. Computer Graphics and Image Processing 10:48–68
10. Natterer F (1986) Mathematics of computerized tomography. Wiley, New York

11. Smith RT, Zoltani CK (1987) An application of the finite element method to maximum entropy tomographic image reconstruction. J Sci Comput 2(3):283–295
12. Smith RT, Zoltani CK, Klem GJ, Coleman MW (1991) Reconstruction of tomographic images from sparse data sets by a new finite element maximum entropy approach. Applied Optics 30(5):573–582
13. Stark H (ed) (1987) Image recovery: theory and application. Acad. Press, New York
14. Wang Y, Lu W (1992) Multi-criterion maximum entropy image reconstruction from projections. IEEE Trans Medical Imaging 11:70–75

Maximum Flow Problem

RAVINDRA K. AHUJA[1], THOMAS L. MAGNANTI[2], JAMES B. ORLIN[3]

[1] Department Industrial and Systems Engineering, University Florida, Gainesville, USA

[2] Sloan School of Management and Department Electrical Engineering and Computer Sci., Massachusetts Institute Technol., Cambridge, USA

[3] Sloan School of Management, Massachusetts Institute Technol., Cambridge, USA

MSC2000: 90C35

Article Outline

Keywords

Network; Maximum flow problem; Minimum cut problem; Augmenting path algorithm; Preflow-push algorithm; Max-flow min-cut theorem

The *maximum flow problem* seeks the maximum possible flow in a capacitated network from a specified source node s to a specified sink node t without exceeding the capacity of any arc. A closely related problem is the *minimum cut problem*, which is to find a set of arcs with the smallest total capacity whose removal separates node s and node t. The maximum flow and minimum cut problems arise in a variety of application settings as diverse as manufacturing, communication systems, distribution planning, matrix rounding, and scheduling. These problems also arise as subproblems in the solution of more difficult network optimization problems. In this article, we study the maximum flow and minimum cut problems, briefly introducing the underlying theory and algorithms, and presenting some applications. See [2] for a wealth of additional material that amplifies on this discussion.

Let $G = (N, A)$ be a *directed network* defined by a set N of n *nodes* and a set A of m *directed arcs*. We refer to nodes i and j as *endpoints* of arc (i, j). A *directed path* $i_1 - i_2 - \cdots - i_k$ is a set of arcs $(i_1, i_2), \dots, (i_{k-1}, i_k)$. Each arc (i, j) has an associated *capacity* u_{ij} denoting the maximum amount of flow on this arc. We assume that each arc capacity u_{ij} is an integer, and let $U = \max \{u_{ij}:(i, j) \in A\}$. The network has two distinguished nodes, a *source node* s and a *sink node* t. To help in representing a network, we use the arc adjacency list $A(i)$ of node i, which is the set of arcs emanating from it, that is, $A(i) = \{(i, j) \in A: j \in N\}$.

The maximum flow problem is to find the maximum flow from the source node s to the sink node t that satisfies the arc capacities and mass balance constraints at all nodes. We can state the problem formally as follows.

$$\max v \tag{1}$$

subject to

$$\sum_{\{j:\ (i,j)\in A\}} x_{ij} - \sum_{\{j:\ (j,i)\in A\}} x_{ji} = \begin{cases} v & \text{for } i = s, \\ 0 & \text{for } i \notin \{s, t\}, \\ -v & \text{for } i = t, \end{cases} \tag{2}$$

$$0 \le x_{ij} \le u_{ij} \text{ for all } (i, j) \in A. \tag{3}$$

We refer to a vector $x = \{x_{ij}\}$ satisfying (2) and (3) as a *flow* and the corresponding value of the scalar variable v as the *value* of the flow. We refer to the constraints (2) as the *mass balance constraints*, and refer to the constraints (3) as the *flow bound constraints*.

In examining the maximum flow problem, we impose two assumptions:

i) all arc capacities are integer; and
ii) whenever the network contains arc (i, j), then it also contains arc (j, i).

The second assumption is nonrestrictive since we allow arcs with zero capacity.

Sometimes the flow vector x might be required to satisfy lower bound constraints imposed upon the arc flows; that is, if $l_{ij} \geq 0$ specifies the lower bound on the flow on arc $(i, j) \in A$, we impose the condition $x_{ij} \geq l_{ij}$. We refer to this problem as the *maximum flow problem with nonnegative lower bounds*. It is possible to transform a maximum flow problem with nonnegative lower bounds into a maximum flow problem with zero lower bounds.

The minimum cut problem is a close relative of the maximum flow problem. A *cut* $[S, \overline{S}]$ partitions the node set N into two subsets S and $\overline{S} = N - S$ It consists of all arcs with one endpoint in S and the other in $\overline{S}$. We refer to the arcs directed from S to $\overline{S}$, denoted by $(S, \overline{S})$, as *forward arcs* in the cut and the arcs directed from $\overline{S}$ to S, denoted by $(S, \overline{S})$, as *backward arcs* in the cut. The cut $[S, \overline{S}]$ is called an $s - t$-cut if $s \in S$ and $t \in \overline{S}$. We define the *capacity of the cut* $[S, \overline{S}]$, denoted as $u[S, \overline{S}]$, as $\sum_{(i,j)\in(S,\overline{S})} u_{ij}$. A *minimum cut* in G is an $s - t$-cut of minimum capacity. We will show that any algorithm that determines a maximum flow in the network also determines a minimum cut in the network.

The remainder of this article is organized as follows. To help in understanding the importance of the maximum flow problem, we begin by describing several applications. In the next section we present some preliminary results concerning flows and cuts. We next discuss two important classes of algorithms for solving the maximum flow problem: augmenting path algorithms, and preflow-push algorithms. As described in the next section, augmenting path algorithms augment flow along directed paths from the source node to the sink node. The proof of the validity of the augmenting path algorithm yields the well-known max-flow min-cut theorem, which implies that the value of a maximum flow in a network equals the capacity of a minimum cut in the network. In the next section, we study preflow-push algorithms that 'flood' the network so that some nodes have excesses and then incrementally 'relieve' the flow from nodes with excesses by sending flow from excess nodes forward toward the sink node or backward toward the source node. In the final section, we study implications of the max-flow min-cut theorem and prove some max-min results in combinatorics.

We would like to design maximum flow algorithms that are guaranteed to be efficient in the sense that their worst-case running times, that is, the total number of multiplications, divisions, additions, subtractions, and comparisons in the worst-case grow slowly in some measure of the problem's size. We say that a maximum flow algorithm is an $O(n^3)$ algorithm, or has a worst-case complexity of $O(n^3)$, if it is possible to solve any maximum flow problem using a number of computations that is asymptotically bounded by some constant times the term n^3. We say that an algorithm is a *polynomial time algorithm* if it's worst-case running time is bounded by a polynomial function of the input size parameters. For a maximum flow problem, the input size parameters are n, m, and $\log U$ (the number of bits needed to specify the largest arc capacity). We refer to a maximum flow algorithm as a *pseudopolynomial time algorithm* if its worst-case running time is bounded by a polynomial function of n, m, and U. For example, an algorithm with worst-case complexity of $O(nm \log U)$ is a polynomial time algorithm, but an algorithm with worst-case complexity of $O(nmU)$ is a pseudopolynomial time algorithm.

Applications

The maximum flow problem arises in a variety of situations and in several forms. Sometimes, it arises directly in combinatorial applications that on the surface might not appear to be maximum flow problems at all; at other times, it occurs as a subproblem in the solution of more difficult network optimization problems. In this section, we describe three applications of the maximum flow problem.

Capacity of Physical Networks

An oil company needs to ship oil from a refinery to a storage facility using the pipelines of its underlying distribution network. In this problem context, the refinery corresponds to a particular node s in the distribution network and the storage facility corresponds to another node t. The capacity of each arc is the maximum amount of oil per unit time that can flow along it. The value of a maximum $s-t$ flow determines the maximum flow rate from the source node s to the sink node t. Similar applications arise in other settings, for example, determining the transmission capacity between two nodes of a telecommunications network.

Feasible Flow Problem

The feasible flow problem consists of finding a feasible flow satisfying the following constraints:

$$\sum_{\{j:\ (i,j)\in A\}} x_{ij} - \sum_{\{j:\ (j,i)\in A\}} x_{ji} = b(i) \text{for all } i \in N, \quad (4)$$

$$0 \le x_{ij} \le u_{ij} \text{ for all } (i,j) \in A. \quad (5)$$

We assume that $\sum_{i\in N} b(i) = 0$. The following distribution scenario illustrates how the feasible flow problem arises in practice. Suppose that merchandise available at several seaports is desired by other ports. We know the stock of merchandise available at the 'supply' ports, the amount required at the other ports, and the maximum quantity of merchandise that can be shipped on a particular sea route. We wish to know whether we can satisfy all of the demands by using the available supplies.

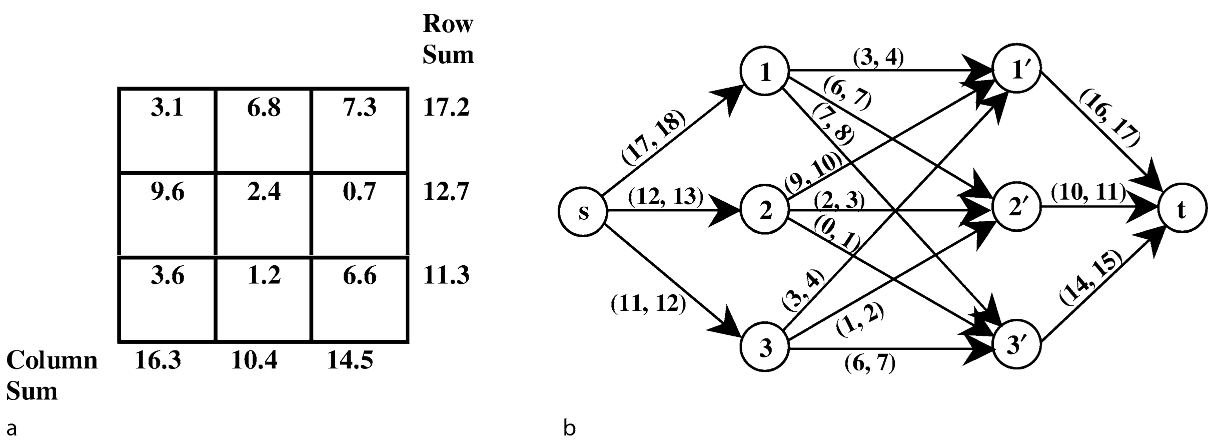
We can solve the feasible flow problem by solving a maximum flow problem defined on an augmented network as follows. We introduce two new nodes, a source node s and a sink node t. For each node i with supply (that is, with $b(i) > 0$), we add an arc (s, i) with capacity $b(i)$, and for each node i with demand (that is, with $b(i) < 0$), we add an arc (i, t) with capacity $-b(i)$. We refer to the new network as the *transformed network*. We then solve a maximum flow problem from node s to node t in the transformed network. It is easy to show that the model (4)–(5) has a feasible solution if and only if the maximum flow saturates all the arcs emanating from the source node, that is, $x_{sj} = u_{sj}$ for all arcs $(s, j) \in A(s)$. Moreover, if each $b(i)$ and u_{ij} is integer, then model (4)–(5) always has an integer feasible solution whenever it has a feasible solution (see Theorem 3).

Sometimes in a feasible flow problem arcs have nonnegative lower bounds, that is, the flow bound constraints are $l_{ij} \le x_{ij} \le u_{ij}$ instead of $0 \le x_{ij} \le u_{ij}$, for some constants $l_{ij} \ge 0$ for each $(i, j) \in A$. By substituting $y_{ij} = x_{ij} - l_{ij}$ for x_{ij}, we can transform this problem to the formulation (4)–(5). Then (5) reduces to $0 \le y_{ij} \le (u_{ij} - l_{ij})$ and (4) reduces to the same set of equations, but with a different right-hand side vector b'.

Matrix Rounding Problem

This application is concerned with consistent rounding of the elements, the row sums, and the column sums of a matrix. We are given a $p \times q$ matrix of real numbers $D = \{d_{ij}\}$, with row sums α_i and column sums β_j. We can round any real number d to the next smaller integer $\lfloor d \rfloor$ or to the next larger integer $\lceil d \rceil$, and the decision to round up or round down is entirely up to us. The matrix-rounding problem requires that we round the matrix elements, and the row and column sums of the matrix so that the sum of the rounded elements in each row equals the rounded row sum, and the sum of the rounded elements in each column equals the rounded column sum. We refer to such a rounding as a *consistent rounding*. The matrix-rounding problem arises is several application contexts, for example, the rounding of census data to disguise data on individuals.

Using a numerical example, we will show how to transform a matrix rounding problem into a maximum flow problem. Figure 1a) shows an instance of the matrix rounding problem and Fig. 1b) gives the maximum flow network G for this problem. The network G contains a node i corresponding to each row i of the matrix D, a node j corresponding to each column j of D, a source node s, and a sink node t. The network contains an arc (i, j) corresponding to the ijth element in the matrix, an arc (s, i) for each row i (this arc represents the sum of row i), an arc (j, t) for each column j (this arc represents the sum of column j). For any arc (i, j), we define its upper bound $u_{ij} = \lceil d_{ij} \rceil$ and lower bound $l_{ij} = \lfloor d_{ij} \rfloor$. Notice that the flow $x_{ij} = d_{ij}$ is a real-valued feasible flow x in the network. Since there is a one-to-one correspondence between the consistent roundings of the matrix and feasible integer flows in the corresponding network, we can find a consistent rounding



Maximum Flow Problem, Figure 1
Network for the matrix rounding problem

by solving a feasible flow problem on the corresponding network. The feasible flow algorithm will produce an integer feasible flow (because of Theorem 3), which corresponds to a consistent rounding.

Preliminaries

In this section, we discuss some elementary properties of flows and cuts. We will use these properties to prove the celebrated max-flow min-cut theorem and to establish the correctness of the augmenting path algorithm described in the next section.

Residual Network

The concept of residual network plays a central role in the development of maximum flow algorithms. Given a flow x, the residual capacity r_{ij} of any arc $(i, j) \in A$ is the maximum additional flow that can be sent from node i to node j using the arcs (i, j) and (j, i). (Recall the assumption from the first Section that whenever the network contains arc (i, j), it also contains the arc (j, i).) The residual capacity r_{ij} has two components:

i) $u_{ij} - x_{ij}$, the unused capacity of arc (i, j);
ii) the current flow x_{ji} on arc (j, i), which we can cancel to increase the flow from node i to node j.

Consequently, $r_{ij} = u_{ij} - x_{ij} + x_{ji}$. We refer to the network $G(x)$ consisting of the arcs with positive residual capacities as the *residual network* (with respect to the flow x). Figure 2 gives an example of a residual network.

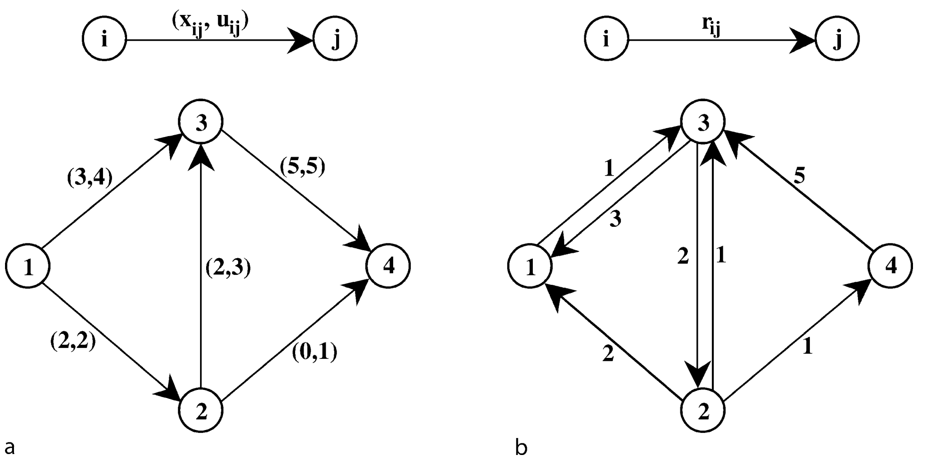
Flow across an $s-t$-Cut

Let x be a flow in the network. Adding the mass balance constraint (2) for the nodes in S, we obtain the equation

$$\begin{aligned} v &= \sum_{i\in S}\left[\sum_{\{j:\ (i,j)\in A\}} x_{ij} - \sum_{\{j:\ (j,i)\in A\}} x_{ji}\right] \\ &= \sum_{(i,j)\in(S,\overline{S})} x_{ij} - \sum_{(i,j)\in(\overline{S},S)} x_{ij}. \end{aligned} \tag{6}$$

The second equality uses the fact that whenever both the nodes p and q belong to the node set S and $(p, q) \in A$, the variable x_{pq} in the first term within the bracket (for node $i = p$) cancels the variable $-x_{pq}$ in the second term within the bracket (for node $j = q$). The first expression in the right-hand side of (6) denotes the amount of flow from the nodes in S to nodes in $\overline{S}$, and the second expression denotes the amount of flow returning from the nodes in $\overline{S}$ to the nodes in S. Therefore, the right-hand side denotes the total (net) flow across the cut, and (6) implies that the flow across any $s-t$-cut $[S, \overline{S}]$ equals v. Substituting $x_{ij} \le u_{ij}$ in the first expression of (6) and $x_{ij} \ge 0$ in the second expression yields: $v \le \sum_{(i,j)\in(S,\overline{S})} u_{ij} = u[S, \overline{S}]$ implying that the value of any flow can never exceed the capacity of any cut in the network. We record this result formally for future reference.

Lemma *The value of any flow can never exceed the capacity of any cut in the network. Consequently, if the value of some flow x equals the capacity of some cut*



Maximum Flow Problem, Figure 2
Illustrating the construction of a residual network; a) the original network, with arc capacities and a flow *x*; b) the residual network

$[S, \overline{S}]$, *then x is a maximum flow and the cut* $[S, \overline{S}]$ *is a minimum cut.*

The *max-flow min-cut theorem*, to be proved in the next section, states that the value of some flow always equals the capacity of some cut.

Generic Augmenting Path Algorithm

In this section, we describe one of the simplest and most intuitive algorithms for solving the maximum flow problem, an algorithm known as the *augmenting path algorithm.*

Let x be a feasible flow in the network G, and let $G(x)$ denote the residual network corresponding to the flow x. We refer to a directed path from the source to the sink in the residual network $G(x)$ as an augmenting path. We define the residual capacity $\delta\ (P)$ of an augmenting path P as the maximum amount of flow that can be sent along it, that is, $\delta\ (P) = \min\ set r_{ij}(i, j) \in P$. Since the residual capacity of each arc in the residual network is strictly positive, the residual capacity of an augmenting path is strictly positive. Therefore, we can always send a positive flow of δ units along it. Consequently, whenever the network contains an augmenting path, we can send additional flow from the source to the sink. (Sending an additional δ units of flow along an augmenting path decreases the residual capacity of each arc (i, j) in the path by δ units.) The generic augmenting path algorithm is essentially based upon this simple observation. The algorithm identifies augmenting paths in $G(x)$ and augments flow on these paths until the network contains no such path. The algorithm below describes the generic augmenting path algorithm.

We can identify an augmenting path P in $G(x)$ by using a graph search algorithm. A graph search algorithm starts at node s and progressively finds all nodes that are reachable from the source node using directed paths. Most search algorithms run in time proportional to the number of arcs in the network, that is, $O(m)$ time, and either identify an augmenting path or conclude that $G(x)$ contains no augmenting path; the latter happens when the sink node is not reachable from the source node.

```
BEGIN
    $x := 0$;
    WHILE $G(x)$ contains a directed path from
    node $s$ to node $t$ DO
    BEGIN
        identify an augmenting path $P$ from $s$ to $t$;
        set $\delta := \min\{r_{ij} : (i, j) \in P\}$;
        augment $\delta$ units of flow along $P$;
        update $G(x)$;
    END;
END;
```

Generic augmenting path algorithm

For each arc $(i, j) \in P$, augmenting δ units of flow along P decreases r_{ij} by δ units and increases r_{ji} by δ units. The final residual capacities r_{ij} when the algorithm terminates specifies a maximum (arc) flow in the following manner. Since $r_{ij} = u_{ij} - x_{ij} + x_{ji}$, the arc flows satisfy the equality $x_{ij} - x_{ji} = u_{ij} - r_{ij}$. If $u_{ij} > r_{ij}$, we can set $x_{ij} = u_{ij} - r_{ij}$ and $x_{ji} = 0$; otherwise, we set $x_{ij} = 0$ and $x_{ji} = r_{ij} - u_{ij}$.

We use the maximum flow problem given in Fig. 3) to illustrate the algorithm. Fig 3a) shows the residual network corresponding to the starting flow $x = 0$, which is identical to the original network. The residual network contains three augmenting paths: $1 - 3 - 4$, $1 - 2 - 4$, and $1 - 2 - 3 - 4$. Suppose the algorithm selects the path $1 - 3 - 4$ for augmentation. The residual capacity of this path is $\delta = \min\{r_{13}, r_{34}\} = \min\{4, 5\} = 4$. This augmentation reduces the residual capacity of arc (1, 3) to zero (thus we delete it from the residual network) and increases the residual capacity of arc (3, 1) to 4 (so we add this arc to the residual network). The augmentation also decreases the residual capacity of arc (3, 4) from 5 to 1, and increases the residual capacity of arc (4, 3) from 0 to 4. Figure 3b) shows the residual network at this stage. In the second iteration, the algorithm selects the path $1 - 2 - 3 - 4$ and augments 1 unit of flow; Fig. 3c) shows the residual network after the augmentation. In the third iteration, the algorithm augments one unit of flow along the path $1 - 2 - 4$. Figure 3d) shows the corresponding residual network. Now the residual network contains no augmenting path and so the algorithm terminates.

Does the augmenting path algorithm always find a maximum flow? The algorithm terminates when the search algorithm fails to identify a directed path in $G(x)$ from node s to node t, indicating that no such path exists (we prove later that the algorithm would terminate finitely). At this stage, let S denote the set of nodes in N that are reachable in $G(x)$ from the source node using directed paths, and $\overline{S} = N - S$. Clearly, $s \in S$ and $t \notin \overline{S}$. Since the search algorithm cannot reach any node in $\overline{S}$ and it can reach each node in S, we know that $r_{ij} = 0$ for each $(i, j) \in (S, \overline{S})$. Recall that $r_{ij} = (u_{ij} - x_{ij}) + x_{ji}$, $x_{ij} \leq u_{ij}$, and $x_{ji} \geq 0$. If $r_{ij} = 0$, then $x_{ij} = u_{ij}$ and $x_{ji} = 0$. Since $r_{ij} = 0$ for each $(i, j) \in (S, \overline{S})$, by substituting these flow values in expression (6), we find that $v = u[S, \overline{S}]$. Therefore, the value of the current flow x equals the capacity of the cut. Lemma 1 implies that x is a maximum flow and $[S, \overline{S}]$ is a minimum cut. This conclusion establishes the correctness of the generic augmenting path algorithm and, as a byproduct, proves the following *max-flow min-cut theorem*.

Theorem 2 *The maximum value of the flow from a source node s to a sink node t in a capacitated network equals the minimum capacity among all $s - t$-cuts.*

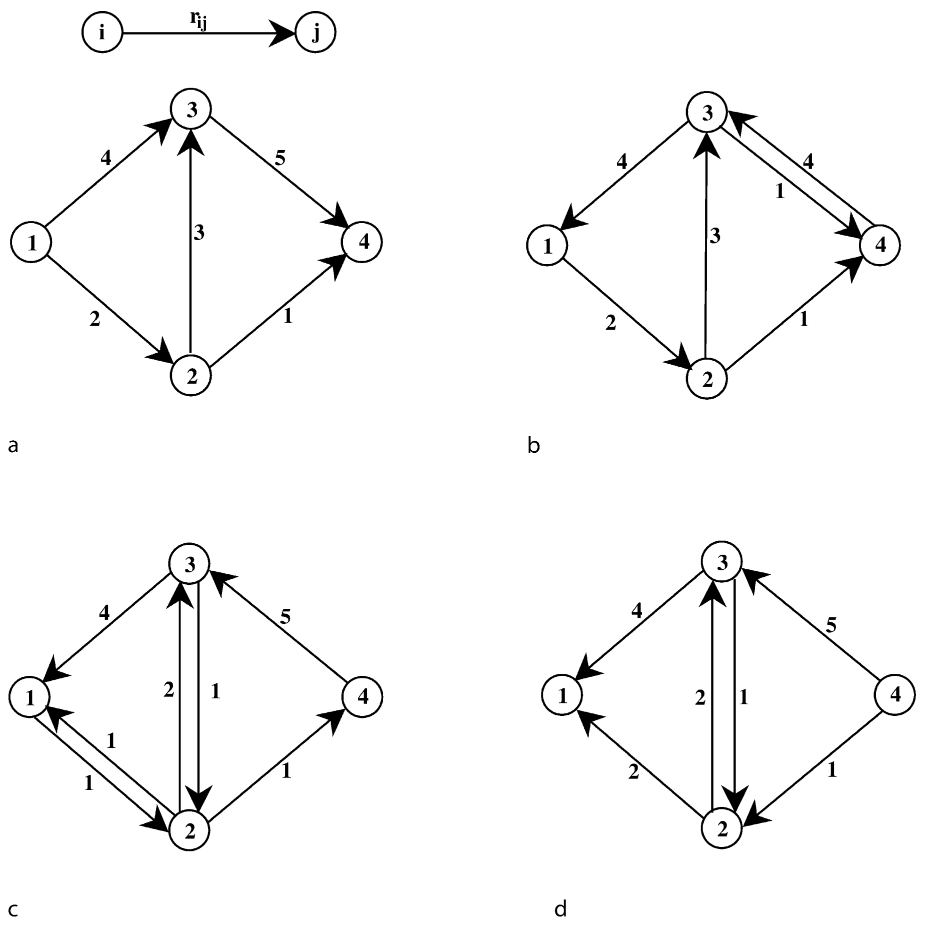
The proof of the max-flow min-cut theorem shows that when the augmenting path algorithm terminates, it also discovers a minimum cut $[S, \overline{S}]$, with S defined as the set of all nodes reachable from the source node in the residual network corresponding to the maximum flow. For our previous numerical example, the algorithm finds the minimum cut in the network, which is $[S, \overline{S}]$ with $S = \{1\}$.

The augmenting path algorithm also establishes another important result, the *integrality theorem*:

Theorem 3 *If all arc capacities are integer, then the maximum flow problem always has an integer maximum flow.*

This result follows from the facts that the initial (zero) flow is integer and all arc capacities are integer; consequently, all initial residual capacities will be integer. Since subsequently all arc flows change by integer amounts (because residual capacities are integer), the residual capacities remain integer throughout the algorithm. Further, the final integer residual capacities determine an integer maximum flow. The integrality theorem does not imply that every optimal solution of the maximum flow problem is integer. The maximum flow problem might have noninteger solutions and, most often, it has such solutions. The integrality theorem shows that the problem always has at least one integer optimal solution.

What is the worst-case running time of the algorithm? An augmenting path is a directed path in $G(x)$ from node s to node t. We have seen earlier that each iteration of the algorithm requires $O(m)$ time. In each iteration, the algorithm augments a positive integer amount of flow from the source node to the sink node. To bound the number of iterations, we will determine a bound on the maximum flow value. By definition, U denotes the largest arc capacity, and so the capacity of the cut $(\{s\}, S - \{s\})$ is at most nU. Since the value of any



Maximum Flow Problem, Figure 3
Illustrating the augmented path algorithm: a) the residual network *G*(*x*) for *x* = 0; b) the residual network after augmenting four units along the path (1 — 3 — 4); c) the residual network after augmenting one unit along the path (1 — 2 — 3 — 4); d) the residual network after augmenting one unit along the path (1 — 2 — 4)

flow can never exceed the capacity of any cut in the network, we obtain a bound of nU on the maximum flow value and also on the number of iterations performed by the algorithm. Consequently, the running time of the algorithm is $O(nmU)$, which is a pseudopolynomial time bound. We summarize the preceding discussion with the following theorem.

Theorem 4 *The generic augmenting path algorithm solves the maximum flow problem in O(nmU) time.*

The augmenting path algorithm is possibly the simplest algorithm for solving the maximum flow problem. Empirically, the algorithm performs reasonably well. However, the worst-case bound on the number of iterations is poor for large values of U. For example, if $U = 2^n$, the bound is exponential in the number of nodes. Moreover, as shown by known examples, the algorithm can indeed perform these many iterations. A second drawback of the augmenting path algorithm is that if the capacities are irrational, the algorithm might not terminate. For some pathological instances of the maximum flow problem, the augmenting path algorithm does not terminate in a finite number of iterations and although the successive flow values converge to some value, they might converge to a value strictly less than the maximum flow value. (Note, however, that the max-flow min-cut theorem is valid even if arc capacities are irrational.) Therefore, if the augmenting path algorithm is to be guaranteed to be effective in all situations, it must select augmenting paths carefully.

Researchers have developed specific implementations of the generic augmenting path algorithms that overcome these drawbacks. Of these, the following three implementations are particularly noteworthy:

i) the maximum capacity augmenting path algorithm which always augments flow along a path in the residual network with the maximum residual capacity and can be implemented to run in $O(m^2 \log U)$ time;

ii) the capacity scaling algorithm which uses a scaling technique on arc capacities and can be implemented to run in $O(nm \log U)$ time;

iii) the shortest augmenting path algorithm which augments flow along a shortest path (as measured by the number of arcs) in the residual network and runs in $O(n^2 m)$ time.

These algorithms are due to J. Edmonds and R.M. Karp [6], H.N. Gabow [9], and E.A. Dinic [5], respectively. L.R. Ford and D.R. Fulkerson [8] and P. Elias, A. Fenstein and C.E. Shannon [7] independently developed the basic augmenting path algorithm.

Generic Preflow-Push Algorithm

Another class of algorithms for solving the maximum flow problem, known as *preflow-push algorithms*, is more decentralized than augmenting path algorithms. Augmenting path algorithms send flow by augmenting along a path. This basic operation further decomposes into the more elementary operation of sending flow along individual arcs. Sending a flow of δ units along a path of k arcs decomposes into k basic operations of sending a flow of δ units along each of the arcs of the path. We shall refer to each of these basic operations as a *push*. The preflow-push algorithms push flows on individual arcs instead of on augmenting paths.

A path augmentation has one advantage over a single push: it maintains conservation of flow at all nodes. The preflow-push algorithms violate conservation of flow at all steps except at the very end, and instead maintain a 'preflow' at each iteration. A *preflow* is a vector x satisfying the flow bound constraints and the following relaxation of the mass balance constraints (2):

$$\sum_{\{j:\ (i,j)\in A\}} x_{ij} - \sum_{\{j:\ (j,i)\in A\}} x_{ji} \geq 0 \quad \text{for all } i \in N - \{s,t\}. \tag{7}$$

Each element of a preflow vector is either a real number or equals $+\infty$. The preflow-push algorithms maintain a preflow at each intermediate stage. For a given preflow x, we define the excess for each node $i \in \mathrm{N} - \{s, t\}$ as

$$e(i) = \sum_{\{j:\ (j,i)\in A\}} x_{ji} - \sum_{\{j:\ (j,i)\in A\}} x_{ij}.$$

We refer to a node with positive excess as an active node. We adopt the convention that the source and sink nodes are never active. In a preflow-push algorithm, the presence of an active node indicates that the solution is infeasible. Consequently, the basic operation in this algorithm is to select an active node i and try to remove the excess by pushing flow out of it. When we push flow out of an active node, we need to do it carefully. If we just push flow to an adjacent node in an arbitrary manner and the other nodes do the same, then it is conceivable that some nodes keep pushing flow among themselves resulting in an infinite loop, which is not a desirable situation. Since ultimately we want to send the flow to the sink node, it seems reasonable for an active node to push flow to another node that is 'closer' to the sink. If all nodes maintain this rule, then the algorithm could never encounter an infinite loop. The concept of distance labels defined next allows us to implement this algorithmic strategy.

The preflow-push algorithms maintain a *distance label* $d(i)$ with each node in the network. The distance labels are nonnegative (finite) integers defined with respect to the residual network $G(x)$. We say that distance labels are valid with respect to a flow x if they satisfy the following two conditions:

$$d(t) = 0, \tag{8}$$

$$d(i) \leq d(j) + 1 \quad \text{for every arc } (i,j) \text{ in the residual network } G(x). \tag{9}$$

We refer to the conditions (8) and (9) as the *validity conditions*. It is easy to demonstrate that $d(i)$ is a lower bound on the length of any directed path (as measured by number of arcs) from node i to node t in the residual network, and thus is a lower bound on the length of the shortest path between nodes i and j. Let $i = i_1 - \cdots - i_k - t$ be any path of length k in the residual network from node i to node t. The validity conditions (8), (9) imply that $d(i) = d(i_1) \leq d(i_2) + 1, d(i_2) \leq d(i_3) + 1, \ldots, d(i_k) \leq d(t) + 1 = 1$. Adding these inequalities shows that $d(i) \leq k$ for any path of length k in the residual network, and therefore any (shortest) path from node i to node t contains at least $d(i)$ arcs. We say that an arc (i, j) in the residual network is *admissible* if it satisfies the condition $d(i) = d(j) + 1$; we refer to all other arcs as *inadmissible*.

The basic operation in the preflow-push algorithm is to select an active node i and try to remove the excess by pushing flow to a node with smaller distance label. (We will use the distance labels as estimates of the length of the shortest path to the sink node.) If node i has an admissible arc (i, j), then $d(j) = d(i) - 1$ and the algorithm sends flow on admissible arcs to relieve the node's excess. If node i has no admissible arc, then the algorithm increases the distance label of node i so that node i has an admissible arc. The algorithm terminates when the network contains no active nodes, that is, excess resides only at the source and sink nodes. The next algorithm describes the generic preflow-push algorithm.

```
BEGIN
    set $x := 0$ and $d(j) := 0$ for all $j \in \mathbb{N}$;
    set $x_{sj} = u_{sj}$ for each arc $(s, j) \in A(s)$;
    $d(s) := n$;
    WHILE residual network $G(x)$ contains an active node
    DO
        BEGIN
            select an active node $I$;
            push/relabel($i$);
        END;
END;
```

```
procedure push/relabel($i$);
BEGIN
    IF network contains an admissible arc $(i, j)$
    THEN push $\delta := \min\{e(i), r_{i,j}\}$ units of flow
    from node $i$ to node $j$
    ELSE replace $d(i)$ by
        $\min\{d(j) + 1 : (i, j) \in A(i), r_{ij} > 0\}$;
END;
```

The generic preflow-push algorithm

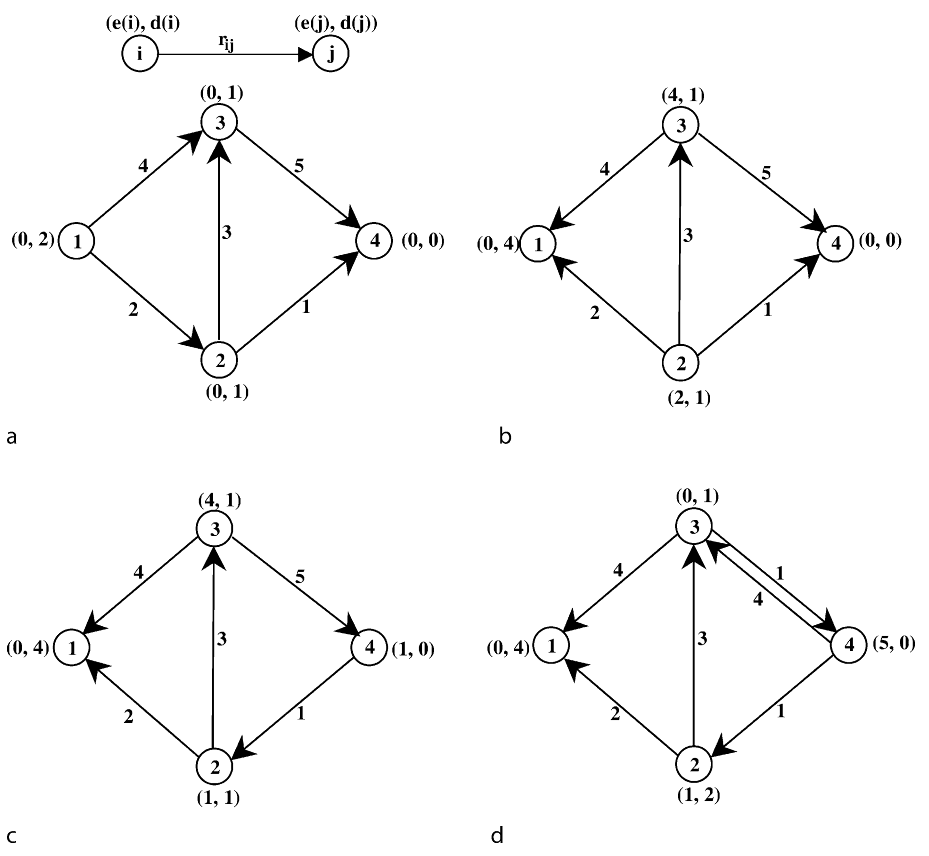
The algorithm first saturates all arcs emanating from the source node; then each node adjacent to node s has a positive excess, so that the algorithm can begin pushing flow from active nodes. Since the preprocessing operation saturates all the arcs incident to node s, none of these arcs is admissible and setting $d(s) = n$ will satisfy the validity condition (8), (9). But then, since $d(s) = n$, and a distance label is a lower bound on the length of the shortest path from that node to node t, the residual network contains no directed path from s to t. The subsequent pushes maintain this property and drive the solution toward feasibility. Consequently, when there are no active nodes, the flow is a maximum flow.

A push of δ units from node i to node j decreases both the excess $e(i)$ of node i and the residual r_{ij} of arc (i, j) by δ units and increases both $e(j)$ and r_{ji} by δ units. We say that a push of δ units of flow on an arc (i, j) is *saturating* if $d = r_{ij}$ and is *nonsaturating* otherwise. A nonsaturating push at node i reduces $e(i)$ to zero. We refer to the process of increasing the distance label of a node as a *relabel* operation. The purpose of the relabel operation is to create at least one admissible arc on which the algorithm can perform further pushes.

It is instructive to visualize the generic preflow-push algorithm in terms of a physical network: arcs represent flexible water pipes, nodes represent joints, and the distance function measures how far nodes are above the ground. In this network, we wish to send water from the source to the sink. We visualize flow in an admissible arc as water flowing downhill. Initially, we move the source node upward, and water flows to its neighbors. Although we would like water to flow downhill toward the sink, occasionally flow becomes trapped locally at a node that has no downhill neighbors. At this point, we move the node upward, and again water flows downhill toward the sink.

Eventually, no more flow can reach the sink. As we continue to move nodes upward, the remaining excess flow eventually flows back toward the source. The algorithm terminates when all the water flows either into the sink or flows back to the source.

To illustrate the generic preflow-push algorithm, we use the example given in Fig 4. Figure 4a) specifies the initial residual network. We first saturate the arcs emanating from the source node, node 1, and set $d(1) = n = 4$. Fig 4b) shows the residual graph at this stage. At



Maximum Flow Problem, Figure 4
Illustrating the preflow-push algorithm: a) the residual network *G*(*x*) for *x* = 0; b) the residual network after saturating arcs emanating from the source; c) the residual network after pushing flow on arc (2, 4); d) the residual network after pushing flow on arc (3, 4)

this point, the network has two active nodes, nodes 2 and 3. Suppose that the algorithm selects node 2 for the push/relabel operation. Arc (2, 4) is the only admissible arc and the algorithm performs a saturating push of value $\delta = \min\{e(2), r_{24}\} = \min\{2, 1\} = 1$. Fig 4c) gives the residual network at this stage. Suppose the algorithm again selects node 2. Since no admissible arc emanates from node 2, the algorithm performs a relabel operation and gives node 2 a new distance label $d(2) = \min\{d(3)+1, d(1)+1\} = \min\{2, 5\} = 2$. The new residual network is the same as the one shown in Fig 4c) except that $d(2) = 2$ instead of 1. Suppose this time the algorithm selects node 3. Arc (3, 4) is the only admissible arc emanating from node 3, and so the algorithm performs a nonsaturating push of value $\delta = \min\{e(3), r_{34}\} = \min\{4, 5\} = 4$. Fig 4d) specifies the residual network at the end of this iteration. Using this process for a few more iterations, the algorithm will determine a maximum flow.

The analysis of the computational (worst-case) complexity of the generic preflow-push algorithm is somewhat complicated. Without examining the details, we might summarize the analysis as follows. It is possible to show that the preflow-push algorithm maintains valid distance labels at all steps of the algorithm and increases the distance label of any node at most $2n$ times. The algorithm performs $O(nm)$ saturating pushes and $O(n^2m)$ nonsaturating pushes. The nonsaturating pushes are the limiting computational operation of the algorithm and so it runs in $O(n^2m)$ time.

The preflow-push algorithm has several attractive features, particularly its flexibility and its potential for further improvements. Different rules for selecting active nodes for the push/relabel operations create many different versions of the generic algorithm, each with different worst-case complexity. As we have noted, the bottleneck operation in the generic preflow-push algorithm is the number of nonsaturating pushes and many specific rules for examining active nodes can produce substantial reductions in the number of nonsaturating pushes. The following specific implementations of the generic preflow-push algorithms are noteworthy:

i) the FIFO preflow-push algorithm examines the active nodes in the first-in, first-out (FIFO) order and runs in $O(n^3)$ time;
ii) the highest label preflow-push algorithm pushes flow from an active node with the highest value of a distance label and runs in $O(n^2 m^{1/2})$ time; and
iii) the excess-scaling algorithm uses the scaling of arc capacities to attain a time bound of $O(nm + n^2 \log U)$.

These algorithms are due to A.V. Goldberg and R.J. Tarjan [10], J. Cheriyan and S.N. Maheshwari [4], and R.K. Ahuja and J.B. Orlin [3], respectively. These preflow-push algorithms are more general, more powerful, and more flexible than augmenting path algorithms. The best preflow-push algorithms currently outperform the best augmenting path algorithms in theory as well as in practice (see, for example, [1]).

Combinatorial Implications of the Max–Flow Min–Cut Theorem

The max-flow min-cut theorem has far reaching consequences. It can be used to prove several important results in combinatorics that appear to be difficult to prove using other means. We will illustrate the use of the max-flow min-cut theorem to prove two such important results.

Network Connectivity

Given a directed network $G = (N, A)$ and two specified nodes s and t, we are interested in the following two questions:

i) what is the maximum number of arc-disjoint (directed) paths from node s to node t; and
ii) what is the minimum number of arcs that we should remove from the network so that it contains no directed paths from node s to node t.

We will show that these two questions are closely related. The second question shows how robust a network, for example, a telecommunications network, is to the failure of its arcs.

In the network G, let us define the capacity of each arc as equal to one. Consider any feasible flow x of value v in the resulting unit capacity network. We can decompose the flow x into flows along v directed paths from node s to node t, each path carrying a unit flow. Now consider any $s-t$-cut $[S, \overline{S}]$ in the network. The capacity of this cut is $\left|(S, \overline{S})\right|$ that is, equals the number of forward arcs in the cut. Since each path joining nodes s and t contains at least one arc in the set $(S, \overline{S})$, the removal of all the arcs in $(S, \overline{S})$ disconnects all paths from node s to node t. Consequently, the network contains a disconnecting set of arcs of cardinality equal to the capacity of any $s-t$-cut $[S, \overline{S}]$. The max-flow min-cut theorem immediately implies the following result:

Corollary 5 *The maximum number of arc-disjoint paths from s to t in a directed network equals the minimum number of arcs whose removal will disconnect all paths from node s to node t.*

Matchings and Covers

The max-flow min-cut theorem also implies a max-min result concerning matchings and node covers in a directed bipartite network $G = (N_1 \cup N_2, A)$, with arc set $A \subseteq N_1 \times N_2$. In the network G, a subset $M \subseteq A$ is a *matching* if no two arcs in M have an endpoint in common. A subset $C \subseteq N_1 N_2$ is a *node cover* of G if every arc in A has at least one endpoint in the node set C. Suppose we create the network G' from G by adding two new nodes s and t, as well as arcs (s, i) of capacity 1 for each $i \in N_1$ and arcs (j, t) of capacity 1 for each $j \in N_2$. All other arcs in G' correspond to the arcs in G and have infinite capacity. It is possible to show that each matching of cardinality v defines a flow of value v in G', and each $s-t$ cut of capacity v induces a corresponding node cover with v nodes. Consequently, the max-flow min-cut theorem establishes the following result:

Corollary 6 *In a bipartite network $G = (N_1 \cup N_2, A)$, the maximum cardinality of any matching equals the minimum cardinality of any node cover of G.*

These two examples illustrate important relationships between maximum flows, minimum cuts, and many other problems in the field of combinatorics. The maximum flow problem is of interest because it provides a unifying tool for viewing many such results, because it arises directly in many applications, and because it has been a rich arena for developing new results concerning the design and analysis of algorithms.

See also

- ▶ Auction Algorithms
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Generalized Networks
- ▶ Minimum Cost Flow Problem
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Nonoriented Multicommodity Flow Problems
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Ahuja RK, Kodialam M, Mishra AK, Orlin JB (1997) Computational investigations of maximum flow algorithms. Europ J Oper Res 97:509–542
2. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs, NJ
3. Ahuja RK, Orlin JB (1989) A fast and simple algorithm for the maximum flow problem. Oper Res 37:748–759
4. Cheriyan J, Maheshwari SN (1989) Analysis of preflow-push algorithms for maximum network flow. SIAM J Comput 18:1057–1086
5. Dinic EA (1970) Algorithm for solution of a problem of maximum flow in networks with power estimation. Soviet Math Dokl 11:1277–1280
6. Edmonds J, Karp RM (1972) Theoretical improvements in algorithmic efficiency for network flow problems. J ACM 19:248–264
7. Elias P, Feinstein A, Shannon CE (1956) Note on maximum flow through a network. IRE Trans Inform Theory IT-2:117–119
8. Ford LR, Fulkerson DR (1956) Maximal flow through a network. Canad J Math 8:399–404
9. Gabow HN (1985) Scaling algorithms for network problems. J Comput Syst Sci 31:148–168
10. Goldberg AV, Tarjan RE (1988) A new approach to the maximum flow problem. J ACM 35:921–940, also: Proc. 19th ACM Symp. Theory of Computing, pp 136–146

Maximum Likelihood Detection via Semidefinite Programming

MIKALAI KISIALIOU, ZHI-QUAN LUO
Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, USA

MSC2000: 65Y20, 68W25, 90C27, 90C22, 49N15

Article Outline

Abstract

Maximum-likelihood detection is a generic NP-hard problem in digital communications which requires efficient solution in practice. Some existing quasi-maximum-likelihood detectors achieve polynomial complexity with significant bit-error-rate performance degradation (e. g. LMMSE Detector), while others exhibit near-maximum-likelihood bit-error-rate performance with exponential complexity (e. g. Sphere Decoder and its variants). We present an efficient suboptimal detector based on a semidefinite relaxation, called SDR Detector, which enjoys near-maximum-likelihood bit-error-rate with worst-case polynomial complexity.

SDR Detector can be implemented with recently developed Interior-Point methods for convex optimization problems. For large systems SDR Detector provides a constant factor approximation for the maximum-likelihood detection problem. In high signal-to-noise ratio region SDR Detector can solve the maximum-likelihood detection problem exactly. Efficient implementations of SDR Detector empirically deliver a near-optimal bit-error-rate with running time that scales well to large problems and in any signal-to-noise ratio region.

Keywords and Phrases

Maximum-likelihood detection; Multiple-input multiple-output systems; Multiuser detection; Semidefinite relaxation

Introduction

Maximum-Likelihood (ML) detection is a fundamental problem in digital communications. Under the mild assumption of equiprobable transmitted signals ML Detector achieves the best Bit-Error-Rate (BER). In general, the ML detection problem is NP-hard due to the discrete nature of a signal constellation. The exhaustive search can be applied for small problem sizes, however this strategy is not practical for large systems. Large communication systems often arise in schemes with efficient rate and diversity utilization, e. g. the systems based on Linear Dispersion Codes [6]. Various suboptimal detectors that have been developed to approximate ML Detector can be divided into two major categories:

- Accelerated versions of ML Detector with exponential complexity (e. g. versions of Sphere Decoder [3,16]),
- Polynomial complexity detectors with significant degradation in the BER performance (e. g. Linear Minimum Mean Square Error (LMMSE) Detector, Matched Filter, Decorrelator, etc.).

We focus on an alternative detector which is based on a semidefinite relaxation of the ML detection problem. This detector, called SDR Detector hereafter, enjoys a worst-case polynomial complexity while delivering a near-optimal BER performance. In the next subsection we will introduce notations and a system model used throughout the text.

Formulation

System Model

Consider a vector communication channel with n transmit and m receive antennas. In wireless communications a Rayleigh fading model is widely used in scenarios with significantly attenuated line-of-sight signal component. An abundant research is based on this model which is used in profound theoretical results on channel capacity, diversity and multiplexing gain. Define a fading coefficient from the ith transmit antenna to the kth receive antenna to be a Gaussian zero-mean unit-variance, $\mathcal{N}(0,1)$, variable H_{ki}, with a Rayleigh distributed amplitude $|H_{ki}|$ and a uniformly distributed phase $\phi(H_{ki})$. The coefficients H_{ki} are assumed to be spatially and temporarily independent and identically distributed (i.i.d.). The transmitted signals $\mathbf{s} = [s_1, \dots, s_n]^T$ are drawn from a discrete n-dimensional complex set C^n. The communication system is operating at an average Signal-to-Noise Ratio (SNR) denoted by ρ. Noise samples at each receive antenna, $\nu_k, k = 1, \dots, m$, are modelled as i.i.d. $\mathcal{N}(0,1)$ random variables. With these notations a Rayleigh memoryless vector channel can be represented by:

$$\mathbf{y} = \sqrt{\rho/n}\,\mathbf{H}\,\mathbf{s} + \mathbf{v}\,. \tag{1}$$

The coefficient $\sqrt{\rho/n}$ ensures that the expected value of SNR at each receive antenna is equal to ρ independent of problem dimension n. Channel model (1) is quite generic and can be used to describe other communication systems, for example, a synchronous CDMA multi-access channel, where n denotes the number of users in the system.

In the sequel, we will assume that the receiver has perfect information of the fading matrix $\mathbf{H}$. In practice $\mathbf{H}$ is estimated by sending training signals which are known to the receiver. Given the vector of received signals $\mathbf{y}$ and the channel state $\mathbf{H}$, the optimal detector computes an estimate of transmitted signals such that the probability of an erroneous decision is minimized. For equiprobable input signals the minimal error probability is achieved by ML Detector given by:

$$\mathbf{s}_{\mathrm{ML}} = \arg\max_{\mathbf{s}\in C^n} p(\mathbf{y}|\mathbf{s},\mathbf{H})\,,$$

where $p(\cdot|\cdot)$ is a conditional probability density function and $\mathbf{s}_{\mathrm{ML}}$ denotes the ML estimate of transmitted signals. For Gaussian noise this optimization problem can be stated in the form of the Integer Least Squares (ILS) problem:

$$\mathbf{s}_{\text{ML}} = \arg\min_{\mathbf{s}\in C^n} \|\mathbf{y} - \sqrt{\rho/n}\,\mathbf{H}\,\mathbf{s}\|^2 \,. \tag{2}$$

In general, this optimization problem is NP-hard and the discrete constraint set C^n of dimension n is the source of intractability. We are interested in an efficient polynomial time approximation algorithm for (2) with theoretical performance guarantees. In the next section we will briefly discuss common approaches to solving problem (2).

Connection with Unconstrained Optimization

Several strategies have been developed to overcome high computational complexity of ML Detector. Some detectors achieve polynomial complexity by relaxing the integer constraint in the ML detection problem (2), e. g. LMMSE Detector, Decorrelator, and Matched Filter [5]. From the perspective of optimization theory these detectors can be jointly treated by dropping the discrete constraint in (2) and imposing a penalty function instead. For the BPSK constellation the relaxed problem can be written as:

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}\in\mathbb{R}^n} \|\mathbf{y} - \sqrt{\rho/n}\,\mathbf{H}\,\mathbf{s}\|^2 + \gamma\,\|\mathbf{s}\|^2 \,. \tag{3}$$

The modified optimization problem is usually followed by a rounding procedure which projects the optimal solution of the relaxed problem onto set C^n. Selecting proper values for γ, we can specialize (3) to LMMSE Detector, Decorrelator, or Matched Filter. An appealing advantage of this approach is that it can be solved analytically:

$$\hat{\mathbf{s}} = \operatorname{sign}\left(\left(\frac{\rho}{n}\,\mathbf{H}^T\mathbf{H} + \gamma\mathbf{I}\right)^{-1}\mathbf{H}^T\mathbf{y}\right) \,. \tag{4}$$

This strategy achieves complexity $\mathcal{O}(n^3)$ while sacrificing the BER performance.

Another type of detectors preserves the near-ML BER while reducing the high complexity of the exhaustive search. The work originates in [3,16] with the algorithm to find the shortest vector on a lattice, known as the so-called Sphere Decoder. The algorithm reduces the exhaustive search to an ellipse centered at the zero-forcing estimate of the transmitted signals:

$$\mathbf{s}_{ZF} = \sqrt{n/\rho}\,\left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{y} \,.$$

Different variants of this approach use various intelligent strategies of the radius selection and ordering of points to be searched inside the ellipse. In high SNR region for small problem sizes Sphere Decoder empirically demonstrates fast running time [7]. However, a thorough theoretical analysis [9,10] has shown that both the worst-case and expected complexity of this algorithm is still exponential.

Semidefinite Relaxation Strategy

We consider an alternative approach to solve (2) which is based on a convex relaxation of the ML detection problem. Convexity of an optimization problem is a good indicator of problem tractability. Efficient and powerful algorithms with complexity $\mathcal{O}(n^{3.5})$ have recently been developed to solve convex optimization problems (e. g. Interior-Point methods). These algorithms make efficient use of theoretically computable stopping criteria, enjoy robustness, and offer the certificate of infeasibility when no solution exists. All these properties render convex optimization methods a primary tool for various fields of engineering.

There are several generic types of convex problems, the simplest one being a Linear Program (LP), i. e. the optimization problem with a linear objective function and linear constraints. An LP allows natural generalization of the notion of an inequality constraint to a so-called Linear Matrix Inequality (LMI). Instead of the regular componentwise meaning of the inequality in LP, LMI $\mathbf{X} \succeq 0$ implies that $\mathbf{X}$ belongs to the cone of symmetric positive semidefinite matrices, i. e. all eigenvalues of $\mathbf{X}$ are non-negative. Such generalization leads us to a generic class of Semi-Definite Programs (SDP), which can be written in the standard form as follows:

$$\begin{aligned} \min\quad & \mathbf{Q}\bullet\mathbf{X} \\ \text{s.t.}\quad & \mathbf{A}_k\bullet\mathbf{X} = b_k, \quad k = 1,\dots,K, \\ & \mathbf{X}\succeq 0\,, \end{aligned} \tag{5}$$

where $(\bullet)$ denotes inner product in the matrix space: $\mathbf{Q}\bullet\mathbf{X} = \operatorname{Tr}(\mathbf{QX})$. The class of SDP problems (5) includes Linear Programs as well as Second Order Cone Programs as special cases. It is quite remarkable that any problem (5) in the broad class of SDP problems can be solved in polynomial time, which makes it a valuable asset for solving engineering problems, including filter design, control, VLSI circuit layout design, etc. [2].

In addition to application in numerical solvers, SDP formulation (5) is widely used for analysis and design of approximation algorithms for NP-hard problems. Traditional approaches involve relaxation of an NP-hard problem to an LP, which can be easily solved in polynomial time. With the invent of Interior-Point methods for non-linear convex optimization problems some approximation algorithms have been significantly improved [4]. Such advanced non-linear approximation algorithms use weaker relaxations, thereby preserving most of the structure of the original NP-hard problem. The class of SDP problems represents a perfect candidate for design of approximation algorithms since the SDP form is quite generic. The solution to the original NP-hard problem is generated from the solution of the relaxed SDP problem by a randomized or deterministic rounding procedure. For example, as will be shown later, the ML detection problem can be formulated as

$$\begin{aligned} f_{\mathrm{ML}} := \min \quad & \mathbf{Q} \bullet \mathbf{X} \\ \text{s.t.} \quad & X_{i,i} = 1, \quad i = 1, \dots, n+1 \\ & \mathbf{X} \succeq 0 \\ & \mathbf{X} \text{ is rank-1} \,. \end{aligned} \tag{6}$$

Relaxing the rank constraint of $\mathbf{X}$ reduces the problem to the standard SDP form (5):

$$\begin{aligned} f_{\mathrm{SDP}} := \min \quad & \mathbf{Q} \bullet \mathbf{X} \\ \text{s.t.} \quad & X_{i,i} = 1, \quad i = 1, \dots, n+1 \\ & \mathbf{X} \succeq 0 \,. \end{aligned} \tag{7}$$

A subsequent rounding procedure generates an estimate of the transmitted signals with an objective value denoted f_{SDR} based on the optimal solution $\mathbf{X}_{\mathrm{opt}}$ of this SDP problem.

Since SDR Detector outputs an estimate that belongs to the feasible set of the ML detection problem, the optimal objective value f_{SDR} of SDR Detector satisfies $f_{\mathrm{ML}} \le f_{\mathrm{SDR}}$. Let f_{opt} (f_{apr}) denote the optimal objective value of an NP-hard problem (approximation algorithm) in minimization form, then the approximation algorithm with ratio $c \ge 1$ guarantees to provide a solution with objective value f_{apr} such that $f_{\mathrm{apr}} \le c f_{\mathrm{opt}}$. The quality of SDR Detector can be measured in terms of approximation ratio c such that:

$$f_{\mathrm{ML}} \le f_{\mathrm{SDR}} \le c f_{\mathrm{ML}}, \quad c \ge 1 \,,$$

where c is independent of problem size.

Relaxation (5) was first applied to combinatorial optimization in [4] where the authors relaxed MAX-CUT problem to an SDP problem in the standard form (5). This strategy resulted in a substantial improvement of the approximation ratio for MAX-CUT problem, as compared to the classical relaxation to an LP. Unfortunately, we can not pursue this approach because the ML detection problem involves minimization instead of maximization (for a positive semidefinite matrix $\mathbf{Q}$) used in the formulation of MAX-CUT problem. Moreover, the ML detection problem does not allow a constant factor approximation algorithm for the worst case realizations of $\mathbf{H}$ and $\mathbf{v}$. However, from the perspective of digital communications we are interested in the average performance of SDR Detector over many channel and noise realizations. It turns out that SDR Detector allows a probabilistic approximation ratio for the random channel model (1). In high SNR region a typical behavior of the detection error probability is

$$P_e \simeq e^{-\gamma(\rho)} \,,$$

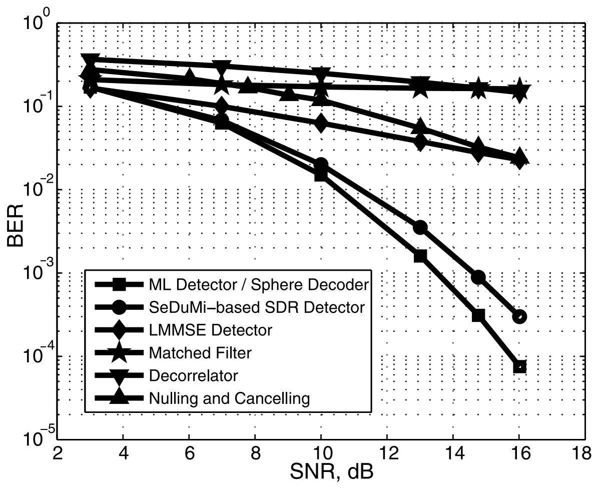
where function $\gamma(\rho)$ varies for different detectors. For example, $\gamma_{\mathrm{ml}}(\rho) = \mathcal{O}(\rho)$ for ML Detector, and $\gamma_{\mathrm{lmmse}}(\rho) = \mathcal{O}(\sqrt{\rho})$ for LMMSE Detector [5]. When a suboptimal detector is deployed instead of ML Detector, the incurred BER deterioration can be expressed in terms of the log-likelihood ratio:

$$\frac{\log(P_e(\mathrm{sdr}))}{\log(P_e(\mathrm{ml}))} = \frac{\gamma_{\mathrm{sdr}}(\rho)}{\gamma_{\mathrm{ml}}(\rho)} \le c(\rho) \,.$$

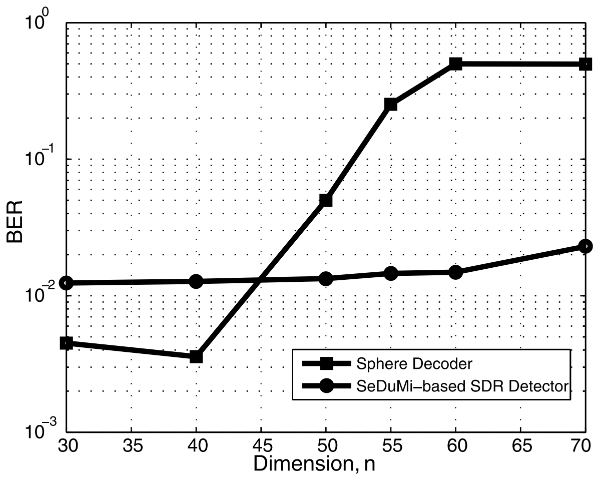
Therefore, the approximation ratio $c(\rho)$ is an essential step in bounding the SNR gap between two detectors. Before we proceed with the probabilistic analysis of the performance, let us consider the empirical BER performance of SDR Detector in numerical simulations for channel model (1).

Bit-Error-Rate Performance

The detector based on a semidefinite relaxation (SDR) consists of two parts: a solver of relaxation (7) and a randomized rounding procedure. The SDP in (7) can



Maximum Likelihood Detection via Semidefinite Programming, Figure 1
Bit-Error-Rate as a function of Signal-to-Noise Ratio for different detectors



Maximum Likelihood Detection via Semidefinite Programming, Figure 2
BER degradation due to the limit on detection time. Simulation parameters: BPSK modulation, SNR = 10 dB and time limit per bit = 6.3 ms

be efficiently solved using Interior Point (IP) methods with complexity $\mathcal{O}(n^{3.5})$. For this purpose we use SeDuMi optimization toolbox for Matlab. The randomized rounding procedure projects the solution of the SDP (7) onto the original discrete constraint set and will be discussed in details in the next section.

Figure 1 shows a comparison of the BER performance of the SeDuMi-based SDR Detector [13], LMMSE Detector, Matched Filter, Decorrelator, Nulling and Cancelling strategy, Sphere Decoder, and ML Detector. We observe a significant BER improvement of SDR Detector compared to other polynomial complexity detectors. Sphere Decoder with adjustable radius search [16] delivers the BER performance of ML Detector (with probability 1) with running time that scales exponentially [9] with problem size.

In many real-time/embedded applications a detection latency is upper bounded and, in general, premature decisions cause significant BER degradation. For simulation purposes we suppose that an engineering system is designed with BPSK modulation, operates at SNR = 10 dB and allows 6.3 ms per bit detection latency. Figure 2 demonstrates the BER performance of this system under the upper bound on the detection latency. The exponential complexity of Sphere Decoder reveals itself between dimensions 40 and 60 where we observe a rapid BER degradation because the running time of Sphere Decoder exceeds the fixed detection time threshold for most channel realizations. At the same time, the running time of SDR Detector scales gracefully with problem size and, in most cases, the detector completes detection in time. As a result, SDR Detector does not suffer any significant BER degradation even for large problem sizes. In fact, the number of late detections for SDR Detector does not exceed 1% for all dimensions shown in Fig. 2. For different values of SNR and latency per bit we obtain essentially similar curves for both detectors. Such behavior is indicative of the exponentially growing computational effort of Sphere Decoder and comparably modest computational power required by SDR Detector.

In the next section we will discuss the details of the SDP relaxation (11) and the randomized rounding procedure. After that we present theoretical guarantees that substantiate the observed empirical behavior of SDR Detector.

Method

SDR Detector consists of two components: an SDP solver and a randomized rounding procedure.

SDP Solver

A transformation of the original ML detection problem (2) into the standard SDP form (5) will help us localize the place in (2) that makes the problem NP-hard. We start with homogenizing the objective function:

$$\begin{aligned}&\|\mathbf{y} - \sqrt{\rho/n}\,\mathbf{Hs}\|^2\\ &= [\mathbf{s}\ \ 1]^T \begin{bmatrix} (\rho/n)\,\mathbf{H}^T\mathbf{H} & -\sqrt{\rho/n}\,\mathbf{H}^T\mathbf{y} \\ -\sqrt{\rho/n}\,\mathbf{y}^T\mathbf{H} & \|\mathbf{y}\|^2 \end{bmatrix} \begin{bmatrix} \mathbf{s} \\ 1 \end{bmatrix}\\ &= \rho\,\mathrm{Tr}(\mathbf{Qxx}^T)\,,\end{aligned}$$

where matrix $\mathbf{Q} \in \mathbb{R}^{(n+1)\times(n+1)}$ and vector $\mathbf{x} \in \mathbb{R}^{n+1}$ are defined as

$$\mathbf{Q} = \begin{bmatrix} (1/n)\,\mathbf{H}^T\mathbf{H} & -\sqrt{1/n\rho}\,\mathbf{H}^T\mathbf{y} \\ -\sqrt{1/n\rho}\,\mathbf{y}^T\mathbf{H} & \|\mathbf{y}\|^2/\rho \end{bmatrix}, \mathbf{x} = \begin{bmatrix} \mathbf{s} \\ 1 \end{bmatrix} \tag{8}$$

Notice, that matrix $\mathbf{Q}$ is composed of the parameters that are known at the receiver. We linearize the objective function by introducing a variable matrix $\mathbf{X}$ to comply with the standard SDP form (5):

$$\begin{aligned} f_{\mathrm{ML}} := \min\quad & \mathrm{Tr}(\mathbf{QX})\\ \text{s.t.}\quad & \mathbf{X} = \mathbf{xx}^T\\ & X_{i,i} = 1, \quad i = 1,\dots,n+1\,. \end{aligned} \tag{9}$$

In this problem formulation we discarded constraint $x_{n+1} = 1$ on the last entry of vector $\mathbf{x}$ because the problem is not sensitive to the sign of vector $\mathbf{x}$. If $\hat{x}_{n+1} = -1$ we output $-\hat{\mathbf{x}}$ as the solution to (9). Constraint $\mathbf{X} = \mathbf{xx}^T$ is equivalent to the set $\{\mathbf{X} \succeq 0,\ \mathrm{rank}(\mathbf{X}) = 1\}$, where notation $\mathbf{X} \succeq 0$ implies that matrix $\mathbf{X}$ is symmetric positive semidefinite. Thus, we complete the transformation of the original ML Detection problem over BPSK constellation to the equivalent form stated in (6):

$$\begin{aligned} f_{\mathrm{ML}} := \min\quad & \mathrm{Tr}(\mathbf{QX})\\ \text{s.t.}\quad & X_{i,i} = 1, \quad i = 1,\dots,n+1\\ & \mathbf{X} \succeq 0\\ & \mathbf{X}\text{ is rank-1}\,. \end{aligned} \tag{10}$$

The rank-1 constraint is the only non-convex constraint in (10) which makes the above problem intractable. SDR Detector relaxes the rank constraint and solves the following convex optimization problem:

$$\begin{aligned} f_{\mathrm{SDP}} := \min\quad & \mathrm{Tr}(\mathbf{QX})\\ \text{s.t.}\quad & X_{i,i} = 1, \quad i = 1,\dots,n+1\\ & \mathbf{X} \succeq 0\,. \end{aligned} \tag{11}$$

To reveal the difference between this relaxation and the one in (3) we can take one step further by relaxing the set of constraints $\{X_{i,i} = 1,\ i = 1,\dots,n+1\}$ into $\{\mathrm{Tr}(\mathbf{X}) = n+1\}$ while keeping constraint $\mathbf{X} \succeq 0$ intact. This extra relaxed problem can be solved analytically and leads to the solution

$$\hat{\mathbf{s}} = \frac{\rho}{n}\left(\frac{\rho}{n}\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{y}\,,$$

which is exactly the soft output of Decorrelator (4) with $\gamma = 0$. The relaxation in (11) compares favorably to the relaxations in (3) because it requires less modifications of the ML problem, although complexity $\mathcal{O}(n^{3.5})$ of (11) is higher than $\mathcal{O}(n^3)$ for the detectors in (3).

Since we dropped the rank constraint in (11), a solution $\mathbf{X}_{\mathrm{opt}}$ of (11) is no longer rank-1, hence, we need to project $\mathbf{X}_{\mathrm{opt}}$ onto the feasible set of the original ML detection problem. Such projection is usually done by a rounding procedure which can be either deterministic like in (4) or randomized [13]. It can also vary depending on the processing power available for the algorithm. In the next section we will consider a randomized rounding procedure based on the principal eigenvector of matrix $\mathbf{X}_{\mathrm{opt}}$.

Randomized Rounding Procedure

There are various rounding procedures that can be used to extract a rank-1 approximation of $\mathbf{X}_{\mathrm{opt}}$. Widely used approaches and their analysis can be found in [4,13,14]. For our purposes we consider the randomized strategy based on the principal eigenvector of matrix $\mathbf{X}_{\mathrm{opt}}$. Notice that in the noise-free case, we have $\mathbf{v} = \mathbf{0}$ and a transmitted vector $\mathbf{s}$ belongs to the kernel of matrix $\mathbf{Q}$ which is defined in (8). The optimal objective function is 0 and is achieved by the vector of transmitted signals $\mathbf{s}$. Thus, in the noise-free case, the optimal solution of problem (11) is a rank-1 matrix:

$$\mathbf{X}_{\mathrm{opt}} = \begin{bmatrix} \mathbf{s} \\ 1 \end{bmatrix} [\mathbf{s}^T\ \ 1]\,.$$

The structure of the optimal matrix $\mathbf{X}_{\mathrm{opt}}$ in the noise-free case suggests that the principal component of the eigen-decomposition contains most reliable information on the transmitted signals in high SNR region. It turns out that the optimal matrix $\mathbf{X}_{\mathrm{opt}}$ has a strong principal component even in low SNR region, justifying the randomized rounding procedure presented below:

- *INPUT*: Solution $\mathbf{X}_{\text{opt}}$ of (11), and number D of randomized rounding tries.
- *OUTPUT*: Quasi-ML estimate $\mathbf{s}_{\text{SDR}}$ and the best achieved objective value f_{SDR}.
- *RANDOMIZED ROUNDING PROCEDURE*:

1. Take a spectral decomposition $\mathbf{X}_{\text{opt}} = \sum_{i=1}^{n+1} \lambda_i \mathbf{u}_i \mathbf{u}_i^T$ and set $\mathbf{v}_i = \sqrt{\lambda_i}\mathbf{u}_i$, $i = 1, \dots, n+1$.
2. Pick $\mathbf{v}_i$ corresponding to the principal eigenvector $\mathbf{v}^{max} = \arg\max_{1 \le i \le n+1} \{\|\mathbf{v}_i\|\}$.
3. For each entry x_i define Bernoulli distribution:
$$\begin{aligned} \Pr\{x_i = +1\} &= (1 + v_i^{max})/2, \\ \Pr\{x_i = -1\} &= (1 - v_i^{max})/2\,, \end{aligned} \tag{12}$$
where v_i^{max} denotes the ith entry of vector $\mathbf{v}^{max}$.
4. Generate a fixed number D of i.i.d. (n+1)-dimensional vector samples $\bar{\mathbf{x}}_d$, $d = 1, \dots, D$, such that each entry of $(\bar{\mathbf{x}}_d)_i$, $i = 1, \dots, n+1$, is drawn from distribution (12).
5. For all D samples, set $\bar{\mathbf{x}}_d := -\bar{\mathbf{x}}_d$ if (n+1)-st entry of $\bar{\mathbf{x}}_d$ is equal to -1.
6. Pick $\mathbf{x}_{\text{SDR}} := \arg\min_d \bar{\mathbf{x}}_d^T \mathbf{Q} \bar{\mathbf{x}}_d$ and set the best achieved objective value $f_{\text{SDR}} := \mathbf{x}_{\text{SDR}}^T \mathbf{Q} \mathbf{x}_{\text{SDR}}$.
7. Return f_{SDR} and $\mathbf{s}_{\text{SDR}}$ which is given by vector $\mathbf{x}_{\text{SDR}}$ with the last bit discarded.

This randomized rounding procedure is designed to ensure that output $\mathbf{s}_{\text{SDR}}$ is equal to the vector of transmitted signals with high probability. Whenever there is an error, the procedure selects $\mathbf{s}_{\text{SDR}}$ to reduce the number of bits in error.

Cases

Performance of SDR Detector

Constant Factor Optimality of SDR Detector The core component of SDR Detector is an approximation algorithm based on the convex relaxation (11) of the original ML detection problem. In this section we analyze the approximation ratio of this algorithm.

A technique pioneered in [4] is widely used in optimization literature to derive a constant factor optimality for SDP-based relaxations. After the optimal solution $\mathbf{X}_{\text{opt}}$ of problem (11) has been obtained the randomized rounding procedure used in [4] defines Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{X}_{\text{opt}})$ (compare with (12)) and implements the n-dimensional sign$(\cdot)$ operator with uniformly generated cutting hyperplanes:

- Generate D i.i.d. samples $\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_D$ from Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{X}_{\text{opt}})$.
- Let $\mathbf{x}_i = \text{sign}(\bar{\mathbf{x}}_i)$ and set the solution $\mathbf{x}_{\text{SDR}}$ that achieves minimum:
$$f_{\text{SDR}} := \mathbf{x}_{\text{SDR}}^T \mathbf{Q} \mathbf{x}_{\text{SDR}} = \min_i \mathbf{x}_i^T \mathbf{Q} \mathbf{x}_i\ .$$

The best objective value f_{SDR} achieved with this randomized rounding procedure can be upper bounded as follows [4]:

$$\begin{aligned} E\{f_{\text{SDR}}\} &= E\left\{\mathbf{x}_{\text{SDR}}^T \mathbf{Q} \mathbf{x}_{\text{SDR}}\right\} \\ &\le^P E\left\{\mathbf{x}_i^T \mathbf{Q} \mathbf{x}_i\right\} \\ &= \text{Tr}\left(\mathbf{Q} E\left\{\mathbf{x}_i \mathbf{x}_i^T\right\}\right) \\ &= \frac{2}{\pi} \text{Tr}\left(\mathbf{Q} \arcsin(\mathbf{X}_{\text{opt}})\right)\ , \end{aligned} \tag{13}$$

where the inequality above holds in probability for sufficiently many samples D, and the last equality follows from that fact that for any scalar random samples $\bar{x}_i$ and $\bar{x}_j$ drawn from $\mathcal{N}(0, 1)$ we have:

$$E\{\text{sign}(\bar{x}_i)\,\text{sign}(\bar{x}_j)\} = \frac{2}{\pi} \arcsin\left(E\left\{\bar{x}_i \bar{x}_j\right\}\right)\ .$$

By taking Taylor expansion of $\arcsin(\mathbf{Y})$, we can see that for any matrix $\mathbf{Y}$, such that $\mathbf{Y} \succeq 0$, $Y_{ii} = 1$ the following inequality holds:

$$\arcsin(\mathbf{Y}) \succeq \mathbf{Y}\ . \tag{14}$$

Suppose that $\mathbf{Q} \preceq 0$, then we have the following upper bound:

$$\text{Tr}\left(\mathbf{Q} \arcsin(\mathbf{X}_{\text{opt}})\right) \le \text{Tr}(\mathbf{Q}\mathbf{X}_{\text{opt}})\ , \tag{15}$$

which allows us to bound f_{SDR} as a constant factor away from f_{ML}:

$$\begin{aligned} f_{\text{ML}} \le E\{f_{\text{SDR}}\} &\le^P \frac{2}{\pi} \text{Tr}(\mathbf{Q}\mathbf{X}_{\text{opt}}) \\ &= \frac{2}{\pi} f_{\text{SDP}} \le \frac{2}{\pi} f_{\text{ML}}\ , \end{aligned}$$

where the first inequality holds because an output of SDR Detector belongs to the feasible set of the ML problem (10), the second inequality follows from (13) combined with (15), the third equality is the definition of f_{SDP}, and the last inequality holds because the SDP problem (11) is a relaxation of the ML problem (10). Therefore, given $\mathbf{Q} \preceq 0$, we obtain a $2/\pi$-approximation ratio for the algorithm. Unfortunately, for ML detection problem the reverse inequality takes place (8):

$$\mathbf{Q} = \begin{bmatrix} (1/n)\,\mathbf{H}^T\mathbf{H} & -\sqrt{1/n\rho}\,\mathbf{H}^T\mathbf{y} \\ -\sqrt{1/n\rho}\,\mathbf{y}^T\mathbf{H} & \|\mathbf{y}\|^2/\rho \end{bmatrix} \succeq 0\,.$$

We can attempt to cure the problem with inequality similar to (14) in the reverse direction for some constant c:

$$\arcsin(\mathbf{Y}) \preceq c\mathbf{Y},\ \text{for all } \mathbf{Y} \succeq 0,\ \text{with } Y_{ii} = 1\,.$$

For this inequality to hold, c must be growing linearly with problem dimension n. Hence, in the limit $n \to \infty$ the constant c together with the approximation ratio of the algorithm grow unbounded. That is, we can not obtain a constant factor approximation by applying the standard technique of [4] to the analysis of the SDP relaxation in (11).

The technique presented above applies to any negative semidefinite matrix $\mathbf{Q}$, hence, in the context of suboptimal detection it attempts to obtain a constant factor optimality for the worst-case channel realization. However, from the perspective of digital communications, we are interested in the average performance of SDR Detector over many channel realizations. Unlike the technique we have discussed above, a probabilistic analysis of Karush–Kuhn–Tucker (KKT) optimality conditions of the semidefinite problem (11) allows us to claim a constant factor optimality for SDR Detector in probability [11].

The optimal objective value f_{SDR} achieved by SDR Detector is within a constant factor $c(\rho,\gamma)$ away from the optimal ML objective value in probability:

$$\lim_{\substack{n,m\to\infty \\ m/n\to\gamma\geq 1}} P\left\{\frac{f_{\mathrm{SDR}}}{f_{\mathrm{ML}}} \leq c(\rho,\gamma)\right\} = 1, \tag{16}$$

$$\text{where } c(\rho,\gamma) = 1 + \frac{2(1+\sqrt{\gamma})^2\beta}{\gamma\rho^\alpha - 1}\,,$$

and $\{\alpha,\beta\}$ are given by

$$\alpha = \begin{cases} \frac{1}{3}, & \text{if } \gamma = 1 \\ \frac{1}{2}, & \text{if } \gamma > 1 \end{cases} \qquad \beta = \begin{cases} 4\sqrt[3]{4}, & \text{if } \gamma = 1 \\ 4\sqrt{\frac{\gamma}{\gamma-1}}, & \text{if } \gamma > 1 \end{cases}$$

The statement implies that the log-likelihood ratio of SDR and ML Detectors is bounded in probability by a constant which is fully specified by SNR only.

Performance of SDR Detector in High SNR Region

We have argued in Sect. "Randomized Rounding Procedure" that the selected randomized rounding procedure provides the optimal solution in the noise-free case. The optimality condition can be extended to the case of large finite SNR: for sufficiently high SNR SDR Detector solves ML detection problem in polynomial time.

For given system dimension n and SNR ρ (both finite), the solution $\mathbf{X}_{\mathrm{opt}}$ of the relaxed problem (11) is rank-1 if channel matrix $\mathbf{H}$ and noise $\mathbf{v}$ realizations satisfy:

$$\lambda_{min}(\mathbf{H}^T\mathbf{H}) > \sqrt{\frac{n}{\rho}}\,\|\mathbf{H}^T\mathbf{v}\|_1\,. \tag{18}$$

Since random matrix $\mathbf{H}^T\mathbf{H}$ is full rank with probability 1, this claim can also be interpreted as follows: for any given n there exists a sufficiently high (finite) SNR level such that (18) holds and $\mathbf{X}_{\mathrm{opt}}$ is rank-1. In general, if (18) does not hold $\mathbf{X}_{\mathrm{opt}}$ may still be rank-1. Notice that if condition (18) is satisfied the solution of the SDP problem (11) belongs to the feasible set of (10), thus, $\mathbf{X}_{\mathrm{opt}}$ is also the solution of the ML detection problem. Hence, under the specified conditions SDR Detector solves the original ML detection problem.

The asymptotic performance of SDR Detector for fixed problem size and $\rho \to \infty$ has been analyzed in [8], where it is shown that for Rayleigh fading $\mathbf{H}$ SDR Detector achieves maximum diversity, i. e.

$$\lim_{\rho\to\infty} \frac{\log P\{\mathbf{s}_{\mathrm{sdr}} \neq \mathbf{s}\}}{\log\rho} = \lim_{\rho\to\infty} \frac{\log P\{\mathbf{s}_{\mathrm{ml}} \neq \mathbf{s}\}}{\log\rho} = -\frac{n}{2}\,.$$

Simulation Results

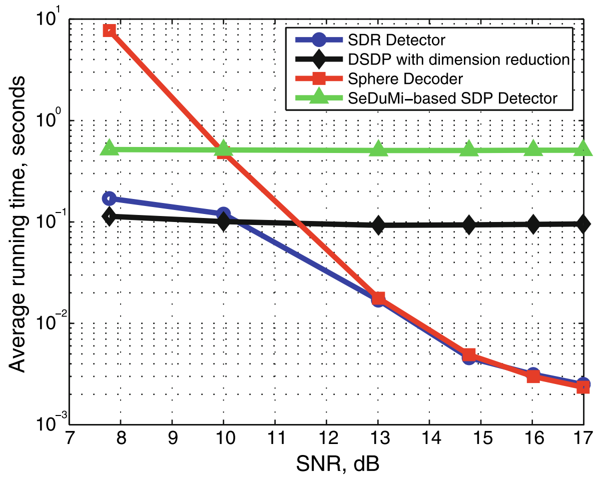
In this section we compare the running time and the BER performance of various implementations of the detectors based on the semidefinite relaxation (11) and that of Sphere Decoder:

- SDP detector [13] implemented with SeDuMi toolbox [15] for convex optimization problems.
- SDR Detector that is based on a dual-scaling interior-point method (DSDP implementation [1]) and a dimension reduction strategy [12].

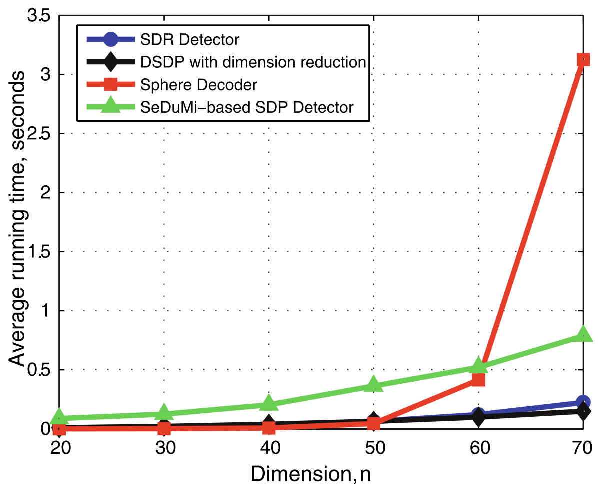
- SDR Detector [12], implemented with a dual-scaling interior-point method, a dimension reduction strategy, and warm start with a truncated version of Sphere Decoder.
- Sphere Decoder [16].

Figures 3 and 4 demonstrate the average running time and the BER performance achieved by the above detectors for problem size $n = 60$. Notice, the running time of DSDP-based (SeDuMi-based) detector is insensitive to SNR, and the BER performance shows 1 dB (2-dB) SNR loss. Sphere Decoder is faster than the semidefinite relaxation-based detectors in high SNR regime but becomes significantly slower for SNR lower than 10 dB. SDR Detector matches the speed of Sphere Decoder in high SNR region, matches the running time of other semidefinite relaxation-based detectors in low SNR regime, and enjoys the near-ML BER performance.

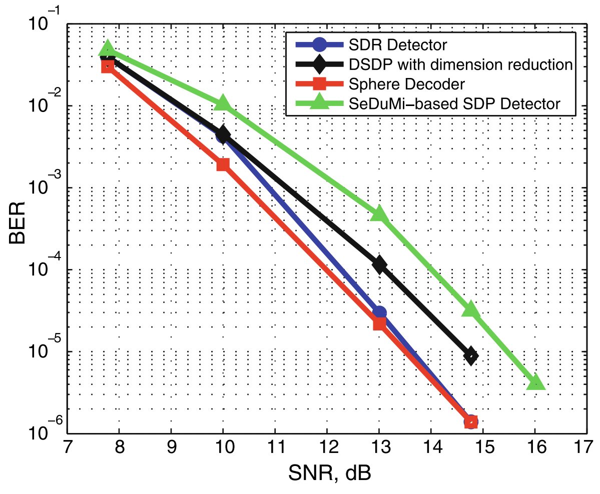
Figures 5 and 6 compare the average running time for large problems and in low SNR region. The running time of polynomial complexity detectors (SDR



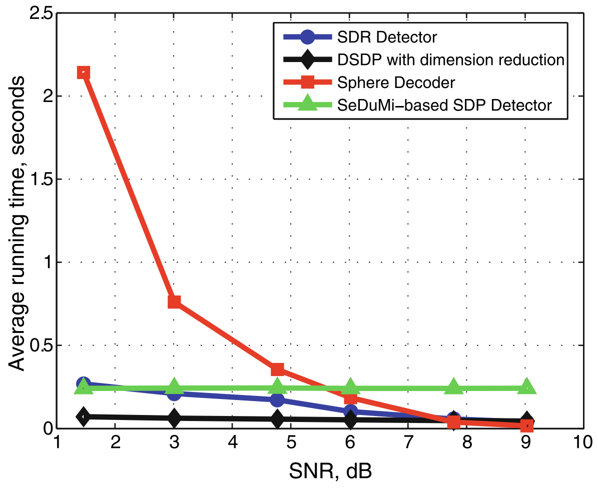
Maximum Likelihood Detection via Semidefinite Programming, Figure 3
Running time comparison, $n = 60$



Maximum Likelihood Detection via Semidefinite Programming, Figure 5
Running time for large problems, $\rho = 10$ dB



Maximum Likelihood Detection via Semidefinite Programming, Figure 4
Bit-error-rate comparison, $n = 60$



Maximum Likelihood Detection via Semidefinite Programming, Figure 6
Running time in low SNR regime, $n = 40$

Detector, SeDuMi and DSDP-based) scales well in both regimes, remaining in the sub-second region, while the running time of Sphere Decoder deteriorates in both scenarios.

Conclusions

We have considered the maximum likelihood detection problem. Among various quasi-ML detectors SDR Detector offers a near-optimal BER performance with the worst-case polynomial complexity. We have analyzed the underlying structure of the SDP relaxation which is the core of SDR Detector. For a given SNR SDR Detector delivers a constant factor approximation of the log-likelihood ratio for the original ML detection problem in probability, where the constant factor is independent of problem size. SDR Detector solves ML detection problem exactly in high SNR region. Numerical simulations of BER and running time empirically demonstrate the advantages of SDR Detector as compared to the computationally expensive ML Detector.

References

1. Benson SJ, Ye Y (2007) DSDP5: Software for semidefinite programming. ACM Trans Math Soft 34(3)
2. Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, New York
3. Fincke U, Pohst M (1985) Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Math Comput 44:463–471
4. Goemans MX, Williamson DP (1995) Improved approximation algorithms for maximum cut and satisfiability problem using semidefinite programming. J ACM 42:1115–1145
5. Guo D, Verdú S (2005) Randomly spread CDMA: asymptotics via statistical physics. IEEE Trans Inf Theory 51:1982–2010
6. Hassibi B, Hochwald BM (2002) High-rate codes that are linear in space and time. IEEE Trans Inf Theory 48(7): 1804–1824
7. Hassibi B, Vikalo H (2001) On the expected complexity of sphere decoding. Thirty-Fifth Asilomar Conference on Signals. Syst Comput 2:1051–1055
8. Jalden J (2006) Detection for multiple input multiple output channels. Ph.D. Thesis, KTH, School of Electrical Engineering, Stockholm
9. Jalden J, Ottersten B (2004) An exponential lower bound on the expected complexity of sphere decoding. Proc ICASSP '04, vol 4, pp IV 393–IV 396
10. Jalden J, Ottersten B (2005) On the complexity of sphere decoding in digital communications. IEEE Trans Signal Process 53(4):1474–1484
11. Kisialiou M, Luo Z-Q (2005) Performance analysis of quasi-maximum-likelihood detector based on semi-definite programming. Proc ICASSP '05, vol 3, pp III 433–III 436
12. Kisialiou M, Luo Z-Q (2007) Efficient implementation of a quasi-maximum-likelihood detector based on semidefinite relaxation. Proc ICASSP '07, vol 4, pp IV 1329–IV 1332
13. Ma WK, Davidson TN, Wong KM, Luo Z-Q, Ching PC (2002) Quasi-maximum-likelihood multiuser detection using semidefinite relaxation. IEEE Trans Signal Process 50(4):912–922
14. Nesterov YE (1997) Quality of semi-difinite relaxation for nonconvex quadratic optimization. CORE Discussion Paper, no 9719
15. Sturm JF (1999) Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim Meth Soft 11–12:625–653
16. Viterbo E, Boutros J (1999) A universal lattice code decoder for fading channels. IEEE Trans Inf Theory 45(5): 1639–1642

Maximum Partition Matching

MPM

JIANER CHEN
Texas A&M University, College Station, USA

MSC2000: 05A18, 05D15, 68M07, 68M10, 68Q25, 68R05

Article Outline

Keywords

Maximum matching; Greedy algorithm; Star network; Parallel routing algorithm

The maximum partition matching problem was introduced recently in the study of routing schemes on interconnection networks [2]. In this article, we study the basic properties of the problem. An efficient algorithm for the maximum partition matching problem is presented.

Definitions and Motivation

Let $\mathbf{S} = \{C_1, \ldots, C_k\}$ be a collection of subsets of the universal set $U = \{1, \ldots, n\}$ such that $\cup_{i=1}^{k} C_i = U$, and $C_i \cap C_j = \emptyset$ for all $i \neq j$. A partition $(\mathbf{A}, \mathbf{B})$ of $\mathbf{S}$ pairs two elements a and b in U if a is contained in a subset in $\mathbf{A}$ and b is contained in a subset in $\mathbf{B}$. A *partition matching* (of order m) of $\mathbf{S}$ consists of two ordered subsets $L = \{a_1, \ldots, a_m\}$ and $R = \{b_1, \ldots, b_m\}$ of m elements of U (the subsets L and R may not be disjoint), together with a sequence of m distinct partitions of $\mathbf{S}$: $(\mathbf{A}_1, \mathbf{B}_1), \ldots, (\mathbf{A}_m, \mathbf{B}_m)$ such that for all $i = 1, \ldots, m$, the partition $(\mathbf{A}_i, \mathbf{B}_i)$ pairs the elements a_i and b_i. The *maximum partition matching problem* is to construct a partition matching of order m for a given collection $\mathbf{S}$ with m maximized.

The maximum partition matching problem arises in connection with the parallel routing problem in interconnection networks. In particular, in the study of the star networks [1], which are attractive alternatives to the popular hypercubes networks. It can be shown that constructing an optimal parallel routing scheme in the star networks can be effectively reduced to the maximum partition matching problem. Readers interested in this connection are referred to [2] for a detailed discussion.

The maximum partition matching problem can be formulated in terms of the *3-dimensional matching problem* as follows: given an instance $\mathbf{S} = \{C_1, \ldots, C_k\}$ of the maximum partition matching problem, we construct an instance M for the 3-dimensional matching problem such that a triple (a, b, P) is contained in M if and only if the partition P of $\mathbf{S}$ pairs the elements a and b. However, since the number of partitions of the collection $\mathbf{S}$ can be as large as 2^n and the 3-dimensional matching problem is *NP*-hard [4], this reduction does not hint a polynomial time algorithm for the maximum partition matching problem.

In the rest of this article, we study the basic properties for the maximum partition matching problem, and present an algorithm of running time $O(n^2 \log n)$ for the problem. We first introduce necessary terminologies that will be used in our discussion.

Let $\pi = \langle L, R, (\mathbf{A}_1, \mathbf{B}_1), \ldots, (\mathbf{A}_m, \mathbf{B}_m)\rangle$ be a partition matching of the collection $\mathbf{S}$, where $L = \{a_1, \ldots, a_m\}$ and $R = \{b_1, \ldots, b_m\}$. We will say that the partition $(\mathbf{A}_i, \mathbf{B}_i)$ *left-pairs* the element a_i and *right-pairs* the element b_i. An element a is said to be *left-paired* if it is in the set L. Otherwise, the element a is *left-unpaired*. Similarly we define *right-paired* and *right-unpaired* elements. The collections $\mathbf{A}_i$ and $\mathbf{B}_i$ are called the *left-collection* and *right-collection* of the partition $(\mathbf{A}_i, \mathbf{B}_i)$. The partition matching π may also be written as $\pi[(a_1, b_1), \ldots, (a_m, b_m)]$ if the corresponding partitions are implied.

For the rest of this paper, we assume that $U = \{1, \ldots, n\}$ and that $\mathbf{S} = \{C_1, \ldots, C_k\}$ is a collection of pairwise disjoint subsets of U such that $\cup_{i=1}^{k} C_i = U$.

Case I. Via Pre-Matching when $\|\mathbf{S}\|$ is Large

A necessary condition for two ordered subsets $L = \{a_1, \ldots, a_m\}$ and $R = \{b_1, \ldots, b_m\}$ of U to form a partition matching for the collection $\mathbf{S}$ is that a_i and b_i belong to different subsets in the collection $\mathbf{S}$, for all $i = 1, \ldots, m$. We say that the two ordered subsets L and R of U form a *pre-matching* $\sigma = \{(a_i, b_i) \colon 1 \le i \le m\}$ if a_i and b_i do not belong to the same subset in the collection $\mathbf{S}$, for all $i = 1, \ldots, m$. The pre-matching σ is *maximum* if m is the largest among all pre-matchings of $\mathbf{S}$.

A maximum pre-matching can be constructed efficiently by the *algorithm pre-matching* given below, where we say that a set is *singular* if it consists of a single element. See [3] for a proof for the correctness of the algorithm.

```
Input :   the collection $\mathbf{S} = \{C_1, \ldots, C_k\}$ of subsets
          of $U$
Output :  a maximum pre-matching $\sigma$ in $\mathbf{S}$
1.        $\mathbf{T} = \mathbf{S}$; $\sigma = \emptyset$;
2.        WHILE $\mathbf{T}$ contains more than one set but
          does not consist of exactly three singular
          sets
          DO
   2.1.   pick two sets $C$ and $C'$ of largest cardinal-
          ity in $\mathbf{T}$;
   2.2.   pick an element $a$ in $C$ and an element $b$
          in $C'$;
   2.3.   $\sigma = \sigma \cup \{(a, b), (b, a)\}$;
   2.4.   $C = C - \{a\}$; $C' = C' - \{b\}$;
   2.5.   if $C$ or $C'$ is empty now, delete it from $\mathbf{T}$;
3.        IF $\mathbf{T}$ consists of exactly three singular sets
          $C_1 = \{a_1\}$, $C_2 = \{a_2\}$, and $C_3 = \{a_3\}$
          THEN
          $\sigma = \sigma \cup \{(a_1, a_2), (a_2, a_3), (a_3, a_1)\}$.
```

Algorithm pre-matching

In the following, we show that when the cardinality of the collection **S** is large enough, a maximum partition matching of **S** can be constructed from the maximum pre-matching σ produced by the algorithm pre-matching.

Suppose that the collection **S** consists of k subsets $C_1, \dots, C_k$ and $2^k \geq 4n$. The pre-matching σ contains at most n pairs. Let (a, b) be a pair in σ and let C and C' be two arbitrary subsets in **S** such that C contains a and C' contains b. Note that the number of partitions $(\mathbf{A}, \mathbf{B})$ of **S** such that C is in **A** and C' is in **B** is equal to $2^{k-2} \geq n$. Therefore, at least one such partition can be used to left-pair a and right-pair b. This observation results in the following theorem.

Theorem 1 *Let* $\mathbf{S} = \{C_1, \dots, C_k\}$ *be a collection of nonempty subsets of the universal set* $U = \{1, \dots, n\}$ *such that* $\cup_{i=1}^{k} C_i = U$ *and* $C_i \cap C_j = \emptyset$*, for* $i \neq j$*. If* $2^k \geq 4n$*, then a maximum partition matching in* **S** *can be constructed in time* $O(n^2)$*.*

Proof Consider the following *algorithm partition-matching-I*.

```
Input:    the collection $\mathbf{S} = \{C_1, \dots, C_k\}$ of subsets of $U$
Output:   a partition matching $\pi$ in $\mathbf{S}$
1.        construct a maximum pre-matching $\sigma$ of $\mathbf{S}$;
2.        FOR each pair $(a, b)$ in $\sigma$ DO
              use an unused partition of $\mathbf{S}$ to pair $a$ and $b$.
```

Algorithm partition-matching-I

Suppose the pre-matching σ constructed in step 1 is $\sigma = \{(a_1, b_1), \dots, (a_m, b_m)\}$. According to the above discussion, for each pair (a_i, b_i) in σ, there is always an unused partition of **S** that left-pairs a and right-pairs b. Therefore, step 2 of the algorithm partition-matching-I is valid and constructs a partition matching π for the collection **S**. Since each partition matching for **S** induces a pre-matching in **S** and σ is a maximum pre-matching, we conclude that the partition matching π is a maximum partition matching for the collection **S**.

By carefully organizing the elements in U and the partitions of **S**, we can show that the algorithm partition-matching-I runs in time $O(n^2)$. See [3].

Case II. Via Greedy Method when $\|\mathbf{S}\|$ is Small

Now we consider the case $2^k < 4n$. Since the number 2^k of partitions of the collection **S** is small, we can apply a greedy strategy that expands a current partition matching by trying to add each of the unused partitions to the partition matching. We show in this section that a careful use of this greedy method constructs a maximum partition matching for the given collection.

Suppose we have a partition matching $\pi = \pi[(a_1, b_1), \dots, (a_h, b_h)]$ and want to expand it. The partitions of the collection **S** then can be classified into two classes: h of the partitions are used to pair the h pairs (a_i, b_i), $i = 1, \dots, h$, and the rest $2^k - h$ partitions are unused. Now if there is an unused partition $P = (\mathbf{A}, \mathbf{B})$ such that there is a left-unpaired element a in **A** and a right-unpaired element b in **B**, then we simply pair the element a with the element b using the partition P, thus expanding the partition matching π.

Now suppose that there is no such unused partition, i. e., for all unused partitions $(\mathbf{A}, \mathbf{B})$, either **A** contains no left-unpaired elements or **B** contains no right-unpaired elements. This case may not necessarily imply that the current partition matching is the maximum. For example, suppose that $(\mathbf{A}, \mathbf{B})$ is an unused partition such that there is a left-unpaired element a in **A** but no right-unpaired elements in **B**. Assume further that there is a used partition $(\mathbf{A}', \mathbf{B}')$ that pairs elements (a', b'), such that the element b' is in **B** and there is a right-unpaired element b in $\mathbf{B}'$. Then we can let the partition $(\mathbf{A}', \mathbf{B}')$ pair the elements (a', b), and then let the partition $(\mathbf{A}, \mathbf{B})$ pair the elements (a, b'), thus expanding the partition matching π. An explanation of this process is that the used partitions have been incorrectly used to pair elements, thus in order to construct a maximum partition matching, we must re-pair some of the elements. To further investigate this relation, we need to introduce a few notations.

For a used partition P of **S**, we put an underline on a set in the left-collection (resp. the right-collection) of P to indicate that an element in the set is left-paired (resp. right-paired) by the partition P. The sets will be called the *left-paired set* and the *right-paired set* of the partition P, respectively.

Definition 2 A used partition P is *directly left-reachable* from a partition $P_1 = (\mathbf{A}_1, \mathbf{B}_1)$ if the left-paired set of P is contained in $\mathbf{A}_1$ (the partition P_1

can be either used or unused). The partition P is *directly right-reachable* from a partition $P_2 = (\mathbf{A}_2, \mathbf{B}_2)$ if the right-paired set of P is contained in $\mathbf{B}_2$. A partition P_s is *left-reachable* (resp. *right-reachable*) from a partition P_1 if there are partitions $P_2, \ldots, P_{s-1}$ such that P_i is directly left-reachable (resp. directly right-reachable) from P_{i-1}, for all $i = 2, \ldots, s$.

The left-reachability and the right-reachability are transitive relations.

Let $P_1 = (\mathbf{A}_1, \mathbf{B}_1)$ be an unused partition such that there are no left-unpaired elements in $\mathbf{A}_1$, and let $P_s = (\mathbf{A}_s, \mathbf{B}_s)$ be a partition left-reachable from P_1 and there is a left-unpaired element a_s in $\mathbf{A}_s$. We show how we can use a *chain justification* to make a left-unpaired element for the collection $\mathbf{A}_1$.

By the definition, there are used partitions $P_2, \ldots, P_{s-1}$ such that P_i is directly left-reachable from P_{i-1}, for $i = 2, \ldots, s$. We can further assume that P_i is not directly left-reachable from P_{i-2} for $i = 3, \ldots, s$ (otherwise we simply delete the partition P_{i-1} from the sequence). Thus, these partitions can be written as

$$\begin{aligned}
P_1 &= (\{C_1\} \cup \mathbf{A}'_1, \mathbf{B}_1),\\
P_2 &= (\{\underline{C_1}, C_2\} \cup \mathbf{A}'_2, \mathbf{B}_2),\\
P_3 &= (\{\underline{C_2}, C_3\} \cup \mathbf{A}'_3, \mathbf{B}_3),\\
&\ddots\\
P_{s-1} &= (\{\underline{C_{s-2}}, C_{s-1}\} \cup \mathbf{A}'_{s-1}, \mathbf{B}_{s-1}),\\
P_s &= (\{\underline{C_{s-1}}, C_s\} \cup \mathbf{A}'_s, \mathbf{B}_s),
\end{aligned}$$

where $\mathbf{A}_1', \ldots, \mathbf{A}_s'$ are subcollections of $\mathbf{S}$ without an underlined set.

We can assume that the left-unpaired element a_s in $\mathbf{A}_s = \{\underline{C_{s-1}}, C_s\} \cup \mathbf{A}'_s$ is in a nonunderlined set C_s in $\mathbf{A}_s$ (otherwise we consider the sequence $P_1, \ldots, P_{s-1}$ instead). We modify the partition sequence into

$$\begin{aligned}
P_1 &= (\{C_1\} \cup \mathbf{A}'_1, \mathbf{B}_1),\\
P_2 &= (\{C_1, \underline{C_2}\} \cup \mathbf{A}'_2, \mathbf{B}_2),\\
P_3 &= (\{C_2, \underline{C_3}\} \cup \mathbf{A}'_3, \mathbf{B}_3),\\
&\vdots\\
P_{s-1} &= (\{C_{s-2}, \underline{C_{s-1}}\} \cup \mathbf{A}'_{s-1}, \mathbf{B}_{s-1}),\\
P_s &= (\{C_{s-1}, \underline{C_s}\} \cup \mathbf{A}'_s, \mathbf{B}_s).
\end{aligned}$$

The interpretation is as follows: we use the partition P_s to left-pair the left-unpaired element a_s (the right-paired element in the right-collection $\mathbf{B}_s$ is unchanged). Thus, the element a_{s-1} in the set C_{s-1} of the partition P_s used to left-pair becomes left-unpaired. We then use the partition P_{s-1} to left-pair the element a_{s-1} and leave an element a_{s-2} in the set C_{s-2} left-unpaired, then we use the partition P_{s-2} to left-pair a_{s-2}, etc. At the end, we use the partition P_2 to left-pair an element a_2 in the set C_2 and leave an element a_1 in the set C_1 left-unpaired. Therefore, this process makes an element in the left-collection $\mathbf{A}_1 = \{C_1\} \cup \mathbf{A}_1'$ of the partition P_1 left-unpaired.

The above process will be called a *left-chain justification*. Thus, given an unused partition $P_1 = (\mathbf{A}_1, \mathbf{B}_1)$ in which the left-collection $\mathbf{A}_1$ has no left-unpaired elements and given a used partition $P_s = (\mathbf{A}_s, \mathbf{B}_s)$ left-reachable from P_1 such that the left-collection $\mathbf{A}_s$ of P_s has a left-unpaired element, we can apply the left-chain justification that keeps all used partitions in the partition matching π and makes a left-unpaired element for the partition P_1. A process called *right-chain justification* for right-collections of the partitions can be described similarly.

A greedy method based on the left-chain and right-chain justifications is presented in the following *algorithm greedy-expanding*.

```
Input:    the collection $\mathbf{S} = \{C_1, \ldots, C_k\}$ of subsets of $U$
Output:   a partition matching $\pi_{\text{exp}}$ in $\mathbf{S}$
1.        $\pi_{\text{exp}} = \emptyset$;
2.        repeat until no more changes
            IF there is an unused partition $P = (\mathbf{A}, \mathbf{B})$ that has a left-unpaired element $a$ in $\mathbf{A}$ and a right-unpaired element $b$ in $\mathbf{B}$
            THEN pair the elements $(a, b)$ by the partition $P$ and add $P$ to the matching $\pi_{\text{exp}}$
            ELSE IF a left-chain justification or a right-chain justification (or both) is applicable to make an unused partition $P = (\mathbf{A}, \mathbf{B})$ to have a left-unpaired element in $\mathbf{A}$ and a right-unpaired element in $\mathbf{B}$
            THEN apply the left-chain justification and/or the right-chain justification
```

Algorithm greedy-expanding

In case $2^k < 4n$, a careful organization of the elements and the partitions can make the running time of the algorithm greedy-expanding bounded by $O(n^2 \log n)$. Briefly speaking, we construct a graph G of 2^k vertices in which each vertex represents a partition of **S**. The direct left- and right- reachabilities of partitions are given by the edges in the graph G, so that checking left- and right- reachabilities and performing left- and right- chain justifications can be done efficiently. Interested readers are referred to [3] for a detailed description.

After execution of the algorithm greedy-expanding, we obtain a partition matching π_{exp}. For each partition $P = (\mathbf{A}, \mathbf{B})$ not included in π_{exp}, either **A** has no left-unpaired elements and no used partition left-reachable from P has a left-unpaired element in its left-collection, or **B** has no right-unpaired elements and no used partition right-reachable from P has a right-unpaired element in its right-collection.

Definition 3 Define L_{free} to be the set of partitions P not used by π_{exp} such that the left-collection of P has no left-unpaired elements and no used partition left-reachable from P has a left-unpaired element in its left-collection, and define R_{free} to be the set of partitions P' not used by π_{exp} such that the right-collection of P' has no right-unpaired elements and no used partition right-reachable from P' has a right-unpaired element in its right-collection.

According to the algorithm greedy-matching, each partition not used by π_{exp} is either in the set L_{free} or in the set R_{free}. The sets L_{free} and R_{free} may not be disjoint.

Definition 4 L_{reac} to be the set of partitions in π_{exp} that are left-reachable from a partition in L_{free}, and define R_{reac} to be the set of partitions in π_{exp} that are right-reachable from a partition in R_{reac}.

According to the definitions, if a used partition P is in the set L_{reac}, then all elements in its left-collection are left-paired, and if a used partition P is in the set R_{reac}, then all elements in its right-collection are right-paired.

We first show that if L_{reac} and R_{reac} are not disjoint, then we can construct a maximum partition matching from the partition matching π_{exp} constructed by the algorithm greedy-expanding. For this, we need the following technical lemma.

Lemma 5 *If the sets L_{reac} and R_{reac} contain a common partition and the partition matching π_{exp} has less than n pairs, then there is a set C_0 in **S**, $|C_0| \le n/2$, such that either all elements in each set $C \ne C_0$ are left-paired and every used partition whose left-paired set is not C_0 is contained in L_{reac}, or all elements in each set $C \ne C_0$ are right-paired and every used partition whose right-paired set is not C_0 is contained in R_{reac}.*

For a proof, see [3].

Theorem 6 *If L_{reac} and R_{reac} have a common partition, then the collection **S** has a maximum partition matching of n pairs, which can be constructed in linear time from the partition matching π_{exp}.*

Proof If π_{exp} has n pairs, then π_{exp} is already a maximum partition matching. Thus we assume that π_{exp} has less than n pairs. According to the above lemma, we can assume, without loss of generality, that all elements in each set C_i, $i = 2, \ldots, k$, are left-paired, and that every used partition whose left-paired set is not C_1 is in L_{reac}. Moreover, $|C_1| \le \sum_{i=2}^{k} |C_i|$.

Let $t = \sum_{i=2}^{k} |C_i|$ and $d = |C_1|$. Then we can assume that the partition matching π_{exp} consists of the partitions

$$P_1, \ldots, P_t, P_{t+1}, \ldots, P_{t+h}$$

where $P_1, \ldots, P_t$ are used by π_{exp} to left-pair the elements in $\cup_{i=2}^{k} C_i$, and $P_{t+1}, \ldots, P_{t+h}$ are used by π_{exp} to left-pair the elements in C_1, $h < d$. Moreover, all partitions $P_1, \ldots, P_t$ are in the set L_{reac}. Thus, the set C_1 must be contained in the right-collection in each of the partitions $P_1, \ldots, P_t$.

We ignore the partitions $P_{t+1}, \ldots, P_{t+h}$ and use the partitions $P_1, \ldots, P_t$ to construct a maximum partition matching of n pairs. Note that $\{P_1, \ldots, P_t\}$ also forms a partition matching in the collection **S**.

For a partition $(\mathbf{A}, \mathbf{B})$ of **S**, we say that the partition $(\mathbf{B}, \mathbf{A})$ is obtained by *flipping* the partition $(\mathbf{A}, \mathbf{B})$. In the following *algorithm partition-flipping*, we show that a maximum partition matching of n pairs can be constructed by flipping d partitions in the partitions $P_1, \ldots, P_t$.

```
Input:  a partition matching $\{P_1, \ldots, P_t\}$ that left-pairs all elements in $\cup_{i=2}^{k} C_i$, $t = \sum_{i=2}^{k} |C_i|$,
        and the set $C_1$ is contained in the right-collection of each partition $P_i$, $i = 1, \ldots, t$,
        $d = |C_1| \le t$
Output: a maximum partition matching in $\mathbf{S}$ with $n$ pairs.
1.      if not all elements in the set $C_1$ are right-paired by $P_1, \ldots, P_t$, replace a
        proper number of right-paired elements in $\cup_{i=2}^{k} C_i$ by the right-unpaired elements
        in $C_1$ so that all elements in $C_1$ are right-paired by $P_1, \ldots, P_t$;
2.      suppose that the partitions $P_1, \ldots, P_{t-d}$ right-pair $t - d$ elements $b_1, \ldots, b_{t-d}$ in
        $\cup_{i=2}^{k} C_i$, and that $P_{t-d+1}, \ldots, P_t$ right-pair the $d$ elements in $C_1$;
3.      suppose that $\overline{P}_1, \ldots, \overline{P}_{t-d}$ are the $t - d$ partitions in $\{P_1, \ldots, P_t\}$ that left-pair the
        elements $b_1, \ldots, b_{t-d}$;
4.      flip each of the $d$ partitions in $\{P_1, \ldots, P_t\} - \{\overline{P}_1, \ldots, \overline{P}_{t-d}\}$ to get $d$
        partitions $P'_1, \ldots, P'_d$ to left-pair the $d$ elements in $C_1$. The right paired element
        of each $P'_i$ is the left-paired element before the flipping;
5.      $\{P_1, \ldots, P_t, P'_1, \ldots, P'_d\}$ is a partition matching of $n$ pairs.
```

Algorithm partition-flipping

Step 1 of the algorithm is always possible: since C_1 is contained in the right-collection of each partition P_i, $i = 1, \ldots, t$, and $t \geq d$, for each right-unpaired element b in C_1, we can always pick a partition P_i that right-pairs an element in $\cup_{i=2}^{k} C_i$, and let P_i right-pair the element b. We keep doing this replacement until all d elements in C_1 get right-paired. At this point, the number of partitions in $\{P_1, \ldots, P_t\}$ that right-pair elements in $\cup_{i=2}^{k} C_i$ is exactly $t- d$. Step 3 is always possible since the partitions $P_1, \ldots, P_t$ left-pair all elements in $\cup_{i=2}^{k} C_i$.

Now we verify that the constructed sequence $\{P_1, \ldots, P_t, P_1', \ldots, P_d'\}$ is a partition matching in $\mathbf{S}$. No two partitions P_i and P_j can be identical since $\{P_1, \ldots, P_t\}$ is supposed to be a partition matching in $\mathbf{S}$. No two partitions P_i' and P_j' can be identical since they are obtained by flipping two different partitions in $\{P_1, \ldots, P_t\}$. No partition P_i is identical to a partition P_j' because P_i has C_1 in its right-collection while P_j' has C_1 in its left-collection. Therefore, the partitions $P_1, \ldots, P_t, P_1', \ldots, P_d'$ are all distinct.

Each of the partitions $P_1, \ldots, P_t$ left-pairs an element in $\cup_{i=2}^{k} C_i$, and each of the partitions $P_1', \ldots, P_d'$ left-pairs an element in C_1. Thus, all elements in the universal set U get left-paired in $\{P_1, \ldots, P_t, P_1', \ldots, P_d'\}$.

Finally, the partitions $P_1, \ldots, P_t$ right-pair all elements in C_1 and the elements $b_1, \ldots, b_{t-d}$ in $\cup_{i=2}^{k} C_i$. Now by our selection of the partitions, the partitions $P_1', \ldots, P_d'$ precisely right-pair all the elements in $\cup_{i=2}^{k} C_i - \{b_1, \ldots, b_{t-d}\}$. Thus, all elements in U also get right-paired in $\{P_1, \ldots, P_t, P_1', \ldots, P_d'\}$.

This concludes that the constructed sequence $\{P_1, \ldots, P_t, P_1', \ldots, P_d'\}$ is a maximum partition matching in the collection $\mathbf{S}$. The running time of the algorithm partition-flipping is obviously linear.

Now we consider the case when the sets L_{reac} and R_{reac} have no common partitions.

Theorem 7 *If L_{reac} and R_{reac} have no common partitions, then the partition matching π_{exp} is a maximum partition matching.*

Proof Let W_{other} be the set of used partitions in π_{exp} that belong to neither L_{reac} nor R_{reac}. Then $L_{\text{free}} \cup R_{\text{free}} \cup L_{\text{reac}} \cup R_{\text{reac}} \cup W_{\text{other}}$ is the set of all partitions of the collection $\mathbf{S}$, and $L_{\text{reac}} \cup R_{\text{reac}} \cup W_{\text{other}}$ is the set of partitions contained in the partition matching π_{exp}. Since all sets L_{reac}, R_{reac}, and W_{other} are pairwise disjoint, the number of partitions in π_{exp} is precisely $|L_{\text{reac}}| + |R_{\text{reac}}| + |W_{\text{other}}|$.

Now consider the set $W_L = L_{\text{free}} \cup L_{\text{reac}}$. Let U_L be the set of elements that appears in the left-collection of a partition in W_L. We have

- Every $P \in L_{\text{reac}}$ left-pairs an element in U_L;
- Every element in U_L is left-paired;
- If an element a in U_L is left-paired by a partition P, then $P \in L_{\text{reac}}$.

Therefore, the partitions in L_{reac} precisely left-pair the elements in U_L. This gives $|L_{\text{reac}}| = |U_L|$. Since there are only $|U_L|$ elements that appear in the left-collections in partitions in $L_{\text{free}} \cup L_{\text{reac}}$, we conclude that the partitions in $W_L = L_{\text{free}} \cup L_{\text{reac}}$ can be used to left-pair at most $|U_L| = |L_{\text{reac}}$ elements in any partition matching in $\mathbf{S}$.

Similarly, the partitions in the set $W_R = R_{\text{free}} \cup R_{\text{reac}}$ can be used to right-pair at most $|R_{\text{reac}}|$ elements in any partition matching in **S**.

Therefore, any partition matching in the collection **S** can include at most $|L_{\text{reac}}|$ partitions in the set W_L, at most $|R_{\text{reac}}|$ partitions in the set W_R, and at most all partitions in the set W_{other}. Consequently, a maximum partition matching in **S** consists of at most $|L_{\text{reac}}| + |R_{\text{reac}}| + |W_{\text{other}}|$ partitions. Since the partition matching π_{exp} constructed by the algorithm greedy-expanding contains just this many partitions, π_{exp} is a maximum partition matching in the collection **S**.

Now it is clear how the maximum partition matching problem is solved.

Theorem 8 *The maximum partition matching problem is solvable in time $O(n^2 \log n)$.*

Proof Suppose that we are given a collection $\mathbf{S} = \{C_1, \dots, C_k\}$ of pairwise disjoint subsets of $U = \{1, \dots, n\}$.

In case $2^k \geq 4n$, we can call the algorithm partition-matching-I to construct a maximum partition matching in time $O(n^2)$.

In case $2^k < 4n$, we first call the algorithm greedy-expanding to construct a partition matching π_{exp} and compute the sets L_{reac} and R_{reac}. If L_{reac} and R_{reac} have no common partition, then according to the previous theorem, π_{exp} is already a maximum partition matching. Otherwise, we call the algorithm partition-flipping to construct a maximum partition matching. All these can be done in time $O(n^2 \log n)$. A detailed analysis of this algorithm can be found in [3].

See also

- ▶ Assignment and Matching
- ▶ Assignment Methods in Clustering
- ▶ Bi-objective Assignment Problem
- ▶ Communication Network Assignment Problem
- ▶ Frequency Assignment Problem
- ▶ Quadratic Assignment Problem

References

1. Akers SB, Krishnamurthy B (1989) A group-theoretic model for symmetric interconnection networks. IEEE Trans Comput 38:555–565
2. Chen C-C, Chen J (1997) Optimal parallel routing in star networks. IEEE Trans Comput 48:1293–1303
3. Chen C-C, Chen J (1999) The maximum partition matching problem with applications. SIAM J Comput 28:935–954
4. Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, New York

Maximum Satisfiability Problem

MAX-SAT

Roberto Battiti
Dip. Mat., Universitá Trento, Povo (Trento), Italy

MSC2000: 03B05, 68Q25, 90C09, 90C27, 68P10, 68R05, 68T15, 68T20, 94C10

Article Outline

Keywords

Maximum satisfiability; Local search; Approximation algorithms; History-sensitive heuristics

In the maximum satisfiability (MAX-SAT) problem one is given a *Boolean formula in conjunctive normal form*, i. e., as a conjunction of clauses, each clause being a disjunction. The task is to find an assignment of truth values to the variables that satisfies the maximum number of clauses.

Let n be the number of variables and m the number of clauses, so that a formula has the following form:

$$\bigwedge_{1\le i\le m} \left(\bigvee_{1\le k\le |C_i|} l_{ik} \right),$$

where $|C_i|$ is the number of literals in clause C_i and l_{ik} is a *literal*, i. e., a propositional variable u_j or its negation $\overline{u_j}$, for $1 \le j \le n$. The set of clauses in the formula is denoted by $\mathcal{C}$. If one associates a weight w_i to each clause C_i one obtains the *weighted MAX-SAT problem*, denoted as MAX W-SAT: one is to determine the assignment of truth values to the n variables that maximizes the sum of the weights of the satisfied clauses. In the literature one often considers problems with different numbers k of literals per clause, defined as MAX-k-SAT, or MAX W-k-SAT in the weighted case. In some papers MAX-k-SAT instances contain up to k literals per clause, while in other papers they contain exactly k literals per clause. We consider the second option unless otherwise stated.

MAX-SAT is of considerable interest not only from the theoretical side but also from the practical one. On one hand, the decision version SAT was the first example of an *NP*-complete problem [16], moreover MAX-SAT and related variants play an important role in the characterization of different approximation classes like APX and PTAS [5]. On the other hand, many issues in mathematical logic and artificial intelligence can be expressed in the form of satisfiability or some of its variants, like constraint satisfaction. Some exemplary problems are consistency in expert system knowledge bases [46], integrity constraints in databases [4,23], approaches to inductive inference [35,40], asynchronous circuit synthesis [32]. An extensive review of algorithms for MAX-SAT appeared in [9].

M. Davis and H. Putnam [19] started in 1960 the investigation of useful strategies for handling resolution in the satisfiability problem. Davis, G. Logemann and D. Loveland [18] avoid the memory explosion of the original DP algorithm by replacing the resolution rule with the *splitting rule*. A recent review of advanced techniques for resolution and splitting is presented in [31].

The MAX W-SAT problem has a natural integer linear programming formulation. Let $y_j = 1$ if Boolean variable u_j is 'true', $y_j = 0$ if it is 'false', and let the Boolean variable $z_i = 1$ if clause C_i is satisfied, $z_i = 0$ otherwise. The integer linear program is:

$$\max \sum_{i=1}^{m} w_i z_i$$

subject to the constraints:

$$\begin{cases} \displaystyle\sum_{j \in U_i^+} y_j + \sum_{j \in U_i^-} (1 - y_j) \geq z_i, \\ \qquad i = 1, \ldots, m, \\ y_j \in \{0,1\}, \qquad j = 1, \ldots, n, \\ z_i \in \{0,1\}, \qquad i = 1, \ldots, m, \end{cases}$$

where U_i^+ and U_i^- denote the set of indices of variables that appear unnegated and negated in clause C_i, respectively. If one neglects the objective function and sets all z_i variables to 1, one obtains an integer programming feasibility problem associated to the SAT problem [11].

The integer linear programming formulation of MAX-SAT suggests that this problem could be solved by a branch and bound method (cf. also ▶ Integer programming: Branch and bound methods). A usable method uses Chvátal cuts. In [35] it is shown that the resolvents in the propositional calculus correspond to certain *cutting planes* in the integer programming model of inference problems.

Linear programming relaxations of integer linear programming formulations of MAX-SAT have been used to obtained upper bounds in [27,33,55]. A linear programming and rounding approach for MAX-2-SAT is presented in [13]. A method for strengthening the generalized set covering formulation is presented in [47], where Lagrangian multipliers guide the generation of cutting planes.

The first approximation algorithms with a 'guaranteed' quality of approximation [5] were proposed by D.S. Johnson [38] and use greedy construction strategies. The original paper [38] demonstrated for both of them a performance ratio 1/2. In detail, let k be the minimum number of variables occurring in any clause of the formula, $m(x, y)$ the number of clauses satisfied by the feasible solution y on instance x, and $m^*(x)$ the maximum number of clauses that can be satisfied.

For any integer $k \geq 1$, the first algorithm achieves a feasible solution y of an instance x such that

$$\frac{m(x,y)}{m^*(x)} \geq 1 - \frac{1}{k+1},$$

while the second algorithm obtains

$$\frac{m(x,y)}{m^*(x)} \geq 1 - \frac{1}{2^k}.$$

Recently (1997) it has been proved [12] that the second algorithm reaches a performance ratio 2/3. There are formulas for which the second algorithm finds a truth assignment such that the ratio is 2/3. Therefore this bound cannot be improved [12].

One of the most interesting approaches in the design of new algorithms is the use of *randomization*.

During the computation, random bits are generated and used to influence the algorithm process. In many cases randomization allows to obtain better (expected) performance or to simplify the construction of the algorithm. Two randomized algorithms that achieve a performance ratio of 3/4 have been proposed in [27] and [55]. Moreover, it is possible to derandomize these algorithms, that is, to obtain deterministic algorithms that preserve the same bound 3/4 for every instance. The approximation ratio 3/4 can be slightly improved [28]. T. Asano [2] (following [3]) has improved the bound to 0.77. For the restricted case of MAX-2-SAT, one can obtain a more substantial improvement (performance ratio 0.931) with the technique in [21]. If one considers only satisfiable MAX W-SAT instances, L. Trevisan [54] obtains a 0.8 approximation factor, while H. Karloff and U. Zwick [41] claim a 0.875 performance ratio for satisfiable instances of MAX W-3-SAT. A strong negative result about the approximability can be found in [36]: Unless $P = NP$ MAX W-SAT cannot be approximated in polynomial time within a performance ratio greater than 7/8.

MAX-SAT is among the problems for which local search has been very successful: in practice, local search and its variations are the only efficient and effective method to address large and complex real-world instances. Different variations of local search with randomness techniques have been proposed for SAT and MAX-SAT starting from the late 1980s, see for example [30,52], motivated by previous applications of 'min-conflicts' heuristics in the area of artificial intelligence [44].

The general scheme is based on generating a starting point in the set of admissible solution and trying to improve it through the application of basic moves. The search space is given by all possible truth assignments. Let us consider the elementary changes to the current assignment obtained by changing a single truth value. The definitions are as follows.

Let U be the discrete search space: $U = \{0, 1\}^n$, and let f be the number of satisfied clauses. In addition, let $U^{(t)} \in U$ be the current configuration along the *search trajectory* at iteration t, and $N(U^{(t)})$ the neighborhood of point $U^{(t)}$, obtained by applying a set of basic moves μ_i $(1 \le i \le n)$, where μ_i complements the ith bit u_i of the string: $\mu_i\ (u_1, \dots, u_i, \dots, u_n) = (u_1, \dots, 1 - u_i, \dots, u_n)$:

$$N(U^{(t)}) = \left\{U \in U\colon\ U = \mu_i,\ U^{(t)},\ i = 1, \dots, n\right\}.$$

The version of local search that we consider starts from a random initial configuration $U^{(0)} \in U$ and generates a search trajectory as follows:

$$V = \text{BESTNEIGHBOR}(N(U^{(t)})), \tag{1}$$

$$U^{(t+1)} = \begin{cases} V & \text{if } f(V) > f(U^{(t)}), \\ U^{(t)} & \text{if } f(V) \le f(U^{(t)}) \end{cases} \tag{2}$$

where BESTNEIGHBOR selects $V \in N(U^{(t)})$ with the best f value and ties are broken randomly. V in turn becomes the new current configuration if f improves. Other versions are satisfied with an improving (or nonworsening) neighbor, not necessarily the best one. Clearly, local search stops as soon as the first *local optimum* point is encountered, when no improving moves are available, see (2). Let us define as LS$^+$ a modification of LS where a specified number of iterations are executed and the candidate move obtained by BESTNEIGHBOR is always accepted even if the f value remains equal or worsens.

Properties about the number of clauses satisfied at a *local optimum* have been demonstrated. Let m^* be the best value and k the minimum number of literals contained in the problem clauses. Let m_{loc} be the number of satisfied clauses at a local optimum of any instance of MAX-SAT with at least k literals per clause. m_{loc} satisfies the following bound [34]:

$$m_{\text{loc}} \ge \frac{k}{k+1} m$$

and the bound is sharp. Therefore, if m_{loc} is the number of satisfied clauses at a local optimum, then:

$$m_{\text{loc}} \ge \frac{k}{k+1} m^*. \tag{3}$$

State-of-the-art heuristics for MAX-SAT are obtained by complementing local search with schemes that are capable of producing better approximations beyond the locally optimal points. In some cases, these schemes generate a sequence of points in the set of admissible solutions in a way that is fixed before the search starts. An example is given by *multiple runs* of local search starting from different random points. The algorithm does not take into account the *history* of the previous phase of the search when the next points are generated. The term 'memory-less' denotes this lack of feedback from the search history.

In addition to the cited multiple-run local search, these techniques are based on Markov processes (simulated annealing; cf. also ► Simulated annealing methods in protein folding), 'plateau' search and 'random noise' strategies, or combinations of randomized constructions and local search. The use of a Markov process to generate a stochastic search trajectory is adopted, for example in [53].

The *Gsat algorithm* was proposed in [52] as a *model-finding procedure*, i. e., to find an interpretation of the variables under which the formula comes out 'true'. Gsat consists of multiple runs of LS^+, each run consisting of a number of iterations that is typically proportional to the problem dimension n. An empirical analysis of Gsat is presented in [24,25]. Different 'noise' strategies to escape from attraction basins are added to Gsat in [50,51].

A hybrid algorithm that combines a randomized greedy construction phase to generate initial candidate solutions, followed be a local improvement phase is the *GRASP* scheme proposed in [48] for the SAT and generalized for the MAX W-SAT problem in [49]. GRASP is an iterative process, with each iteration consisting of two phases, a construction phase and a local search phase.

Different *history-sensitive heuristics* have been proposed to continue local search schemes beyond local optimality. These schemes aim at intensifying the search in promising regions and at diversifying the search into uncharted territories by using the information collected from the previous phase (the history) of the search. Because of the internal feedback mechanism, some algorithm parameters can be modified and tuned in an *on-line* manner, to reflect the characteristics of the *task* to be solved and the *local* properties of the configuration space in the neighborhood of the current point. This tuning has to be contrasted with the *off-line* tuning of an algorithm, where some parameters or choices are determined for a given problem in a preliminary phase and they remain fixed when the algorithm runs on a specific instance.

Tabu search is a *history-sensitive heuristic* proposed by F. Glover [26] and, independently, by P. Hansen and B. Jaumard, that used the term 'SAMD' (steepest ascent mildest descent) and applied it to the MAX-SAT problem in [34]. The main mechanism by which the history influences the search in tabu search is that, at a given iteration, some neighbors are *prohibited*, only a nonempty subset $N_A(U^{(t)}) \subset N(U^{(t)})$ of them is *allowed*. The general way of generating the search trajectory that we consider is given by:

$$N_A(U^{(t)}) = \text{allow}(N(U^{(t)}), \dots, U^{(t)}), \tag{4}$$

$$U^{(t+1)} = \text{BESTNEIGHBOR}(N_A(U^{(t)})). \tag{5}$$

The set-valued function allow selects a nonempty subset of $N(U^{(t)})$ in a manner that depends on the entire previous history of the search $U^{(0)}, \dots, U^{(t)}$. A specialized tabu search heuristic is used in [37] to speed up the search for a solution (if the problem is satisfiable) as part of a branch and bound algorithm for SAT, that adopts both a relaxation and a decomposition scheme by using polynomial instances, i. e., 2-SAT and Horn-SAT.

Different methods to generate prohibitions produce *discrete dynamical systems* with qualitatively different *search trajectories*. In particular, prohibitions based on a list of *moves* lead to a faster escape from a locally optimal point than prohibitions based on a list of visited *configurations* [6]. In detail, the function allow can be specified by introducing a *prohibition parameter* T (also called *list size*) that determines how long a move will remain prohibited after its execution. The *fixed tabu search* algorithm is obtained by fixing T throughout the search [26]. A neighbor is allowed if and only if it is obtained from the current point by applying a move that has not been used during the last T iterations. In detail, if $\mathrm{LU}(\mu)$ is the last usage time of move μ ($\mathrm{LU}(\mu) = -\infty$ at the beginning):

$$N_A(U^{(t)}) = \left\{ U = \mu U^{(t)} : \ \mathrm{LU}(\mu) < (t - T) \right\}.$$

The *reactive tabu search* algorithm of [10], defines simple rules to determine the prohibition parameter by reacting to the repetition of previously-visited configurations. One has a repetition if $U^{(t+R)} = U^{(t)}$ for $R \geq 1$. The prohibition period T depends on the iteration t and a *reaction equation* is added to the dynamical system:

$$T^{(t)} = \text{react}(T^{(t-1)}, U^{(0)}, \ldots, U^{(t)}).$$

An algorithm that combines local search and *nonoblivious local search* [8], the use of prohibitions, and a reactive scheme to determine the prohibition parameter is the *Hamming-reactive tabu search* algorithm proposed in [7], which contains also a detailed experimental analysis.

Given the hardness of the problem and the relevancy for applications in different fields, the emphasis on the experimental analysis of algorithms for the MAX-SAT problem has been growing in recent years (as of 2000).

In some cases the experimental comparisons have been executed in the framework of 'challenges,' with support of electronic collection and distribution of software, problem generators and test instances. An example is the the Second DIMACS algorithm implementation challenge on cliques, coloring and satisfiability, whose results have been published in [39]. Practical and industrial MAX-SAT problems and benchmarks, with significant case studies are also presented in [20]. Some basic problem models that are considered both in theoretical and in experimental studies of MAX-SAT algorithms are described in [31].

Different algorithms demonstrate a different degree of effort, measured by number of elementary steps or CPU time, when solving different kinds of instances. For example, in [45] it is found that some distributions used in past experiments are of little interest because the generated formulas are almost always very easy to satisfy. It also reports that one can generate very hard instances of k-SAT, for $k \geq 3$. In addition, it reports the following observed behavior for random fixed length 3-SAT formulas: if r is the ratio r of clauses to variables ($r = m/n$), almost all formulas are satisfiable if $r < 4$, almost all formulas are unsatisfiable if $r > 4.5$. A rapid transition seems to appear for $r \approx 4.2$, the same point where the computational complexity for solving the generated instances is maximized, see [17,42] for reviews of experimental results.

Let κ be the least real number such that, if r is larger than κ, then the probability of $\mathcal{C}$ being satisfiable converges to 0 as n tends to infinity. A notable result found independently by many people, including [22] and [14] is that

$$\kappa \leq \log_{\frac{8}{7}} 2 = 5.191.$$

A series of theoretical analyses aim at approximating the *unsatisfiability threshold* of random formulas [1,15,29,43].

See also

- ▶ Greedy Randomized Adaptive Search Procedures
- ▶ Integer Programming

References

1. Achlioptas D, Kirousis LM, Kranakis E, Krinzac D (1997) Rigorous results for random (2 + p)-SAT. In: Proc. Work. on Randomized Algorithms in Sequential, Parallel and Distributed Computing (RALCOM 97), Santorini, Greece, pp 1–10
2. Asano T (1997) Approximation algorithms for MAX-SAT: Yannakakis vs. Goemans–Williamson. In: Proc. 3rd Israel Symp. on the Theory of Computing and Systems, Ramat Gan, Israel, pp 24–37
3. Asano T, Ono T, Hirata T (1996) Approximation algorithms for the maximum satisfiability problem. Proc. 5th Scandinavian Work. Algorithms Theory, pp 110–111
4. Asirelli P, de Santis M, Martelli A (1985) Integrity constraints in logic databases. J Logic Programming 3:221–232
5. Ausiello G, Crescenzi P, Protasi M (1995) Approximate solution of NP optimization problems. Theoret Comput Sci 150:1–55
6. Battiti R (1996) Reactive search: Toward self-tuning heuristics. In: Rayward-Smith VJ, Osman IH, Reeves CR, Smith GD (eds) Modern Heuristic Search Methods. Wiley, New York, pp 61–83
7. Battiti R, Protasi M (1997) Reactive search, a history-sensitive heuristic for MAX-SAT. ACM J Experimental Algorithmics 2:2
8. Battiti R, Protasi M (1997) Solving MAX-SAT with non-oblivious functions and history-based heuristics. In: Du D-Z, Gu J, Pardalos PM (eds) Satisfiability Problem: Theory and Applications. DIMACS, vol 35. Amer. Math. Soc. and ACM, Providence, RI, pp 649–667
9. Battiti R, Protasi M (1998) Approximate algorithms and heuristics for MAX-SAT. In: Du D-Z and Pardalos PM (eds) Handbook Combinatorial Optim. Kluwer, Dordrecht, pp 77–148
10. Battiti R, Tecchiolli G (1994) The reactive tabu search. ORSA J Comput 6(2):126–140
11. Blair CE, Jeroslow RG, Lowe JK (1986) Some results and experiments in programming for propositional logic. Comput Oper Res 13(5):633–645
12. Chen J, Friesen D, Zheng H (1997) Tight bound on Johnson's algorithm for MAX-SAT. In: Proc. 12th Annual IEEE

Conf. Computational Complexity (Ulm, Germany), pp 274–281
13. Cheriyan J, Cunningham WH, Tuncel T, Wang Y (1996) A linear programming and rounding approach to MAX 2-SAT. In: Trick M, Johson DS (eds) Proc. Second DIMACS Algorithm Implementation Challenge on Cliques, Coloring and Satisfiability. DIMACS vol 26, pp 395–414
14. Chvátal V, Szemerédi E (1988) Many hard examples for resolution. J ACM 35:759–768
15. Chvátal V, Reed B (1992) Mick gets some (the odds are on his side). In: Proc. 33th Ann. IEEE Symp. on Foundations of Computer Sci., IEEE Computer Soc., pp 620–627
16. Cook SA (1971) The complexity of theorem-proving procedures. In: Proc. Third Annual ACM Symp. Theory of Computing, pp 151–158
17. Cook SA, Mitchell DG (1997) Finding hard instances of the satisfiability problem: A survey. In: Du D-Z Gu J, Pardalos PM (eds) Satisfiability Problem: Theory and Applications. DIMACS, vol 35. Amer. Math. Soc. and ACM, Providence, RI, pp 1–17
18. Davis M, Logemann G, Loveland D (1962) A machine program for theorem proving. Comm ACM 5:394–397
19. Davis M, Putnam H (1960) A computing procedure for quantification theory. J ACM 7:201–215
20. Du D-Z, Gu J, Pardalos PM (eds) (1997) Satisfiability problem: Theory and applications. DIMACS, vol 35. Amer. Math. Soc. and ACM, Providence, RI
21. Feige U, Goemans MX (1995) Approximating the value of two proper proof systems, with applications to MAX-2SAT and MAX-DICUT. In: Proc. Third Israel Symp. Theory of Computing and Systems, pp 182–189
22. Franco J, Paull M (1983) Probabilistic analysis of the Davis–Putnam procedure for solving the satisfiability problem. Discrete Appl Math 5:77–87
23. Gallaire H, Minker J, Nicolas JM (1984) Logic and databases: A deductive approach. Computing Surveys 16(2):153–185
24. Gent IP, Walsh T (1993) An empirical analysis of search in GSAT. J Artif Intell Res 1:47–59
25. Gent IP, Walsh T (1993) Towards an understanding of hill-climbing procedures for SAT. In: Proc. Eleventh Nat. Conf. Artificial Intelligence, AAAI Press/MIT, pp 28–33
26. Glover F (1989) Tabu search: Part I. ORSA J Comput 1(3):190–260
27. Goemans MX, Williamson DP (1994) New 3/4-approximation algorithms for the maximum satisfiability problem. SIAM J Discret Math 7(4):656–666
28. Goemans MX, Williamson DP (1995) Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J ACM 42(6):1115–1145
29. Goerdt A (1996) A threshold for unsatisfiability. J Comput Syst Sci 53:469–486
30. Gu J (1992) Efficient local search for very large-scale satisfiability problem. ACM SIGART Bull 3(1):8–12
31. Gu J, Purdom PW, Franco J, Wah BW (1997) Algorithms for the satisfiability (SAT) problem: A survey. In: Du D-Z Gu J, Pardalos PM (eds) Satisfiability Problem: Theory and Applications. DIMACS, vol 35. Amer. Math. Soc. and ACM, Providence, RI
32. Gu J, Puri R (1995) Asynchronous circuit synthesis with Boolean satisfiability. IEEE Trans Computer-Aided Design Integr Circuits 14(8):961–973
33. Hammer PL, Hansen P, Simeone B (1984) Roof duality, complementation and persistency in quadratic 0-1 optimization. Math Program 28:121–155
34. Hansen P, Jaumard B (1990) Algorithms for the maximum satisfiability problem. Computing 44:279–303
35. Hooker JN (1988) Resolution vs. cutting plane solution of inference problems: some computational experience. Oper Res Lett 7(1):1–7
36. Håstad J (1997) Some optimal inapproximability results. In: Proc. 28th Annual ACM Symp. on Theory of Computing, El Paso, Texas, pp 1–10
37. Jaumard B, Stan M, Desrosiers J (1996) Tabu search and a quadratic relaxation for the satisfiability problem. In: Trick M, Johson DS (eds) Proc. Second DIMACS Algorithm Implementation Challenge on Cliques, Coloring and Satisfiability. DIMACS, vol 26, pp 457–477
38. Johnson DS (1974) Approximation algorithms for combinatorial problems. J Comput Syst Sci 9:256–278
39. Johnson DS, Trick M (eds) (1996) Cliques, coloring, and satisfiability: Second DIMACS implementation challenge. DIMACS, vol 26. Amer Math Soc, Providence, RI
40. Kamath AP, Karmarkar NK, Ramakrishnan KG, Resende MG (1990) Computational exprience with an interior point algorithm on the satisfiability problem. Ann Oper Res 25:43–58
41. Karloff H, Zwick U (1997) A 7/8-approximation algorithm for MAX 3SAT? In: Proc. 38th Annual IEEE Symp. Foundations of Computer Sci., IEEE Computer Soc
42. Kirkpatrick S, Selman B (1994) Critical behavior in the satisfiability of random Boolean expressions. Science 264:1297–1301
43. Kirousis LM, Kranakis E, Krizanc D (Sept. 1996) Approximating the unsatisfiability threshold of random formulas. In: Proc. Fourth Annual European Symp. Algorithms. Springer, Berlin, pp 27–38
44. Minton S, Johnston MD, Philips AB, Laird P (1990) Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In: Proc. 8th Nat. Conf. Artificial Intelligence (AAAI-90), pp 17–24
45. Mitchell D, Selman B, Levesque H (1992) Hard and easy distributions of SAT problems. In: Proc. 10th Nat. Conf. Artificial Intelligence (AAAI-92), pp 459–465
46. Nguyen TA, Perkins WA, Laffrey TJ, Pecora D (1985) Checking an expert system knowledge base for consistency and completeness. In: Proc. Internat. Joint Conf. on Artificial Intelligence, pp 375–378

47. Nobili P, Sassano A (1996) Strengthening Lagrangian bounds for the MAX-SAT problem. Techn. Report Inst. Informatik Köln Univ., Germany, no. 96–230; Franco J, Gallo G, Kleine Buening H (eds) Proc. Work Satisfiability Problem, Siena, Italy
48. Resende MGC, Feo TA (1996) A GRASP for satisfiability. In: Trick M, Johson DS (eds) Proc. Second DIMACS Algorithm Implementation Challenge on Cliques, Coloring and Satisfiability. DIMACS, vol 26. Amer. Math. Soc., Providence, RI, pp 499–520
49. Resende MGC, Pitsoulis LS, Pardalos PM (1997) Approximate solution of weighted MAX-SAT problems using GRASP. In: Du D-Z Gu J, Pardalos PM (eds) Satisfiability Problem: Theory and Applications. DIMACS, vol 35. Amer Math Soc, Providence, RI
50. Selman B, Kautz H (1993) Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In: Proc. Internat. Joint Conf. Artificial Intelligence, pp 290–295
51. Selman B, Kautz HA, Cohen B (1996) Local search strategies for satisfiability testing. In: Trick M, Johson DS (eds) Proc. Second DIMACS Algorithm Implementation Challenge on Cliques, Coloring and Satisfiability. DIMACS, vol 26, pp 521–531
52. Selman B, Levesque H, Mitchell D (1992) A new method for solving hard satisfiability problems. In: Proc. 10th Nat. Conf. Artificial Intelligence (AAAI-92), pp 440–446
53. Spears WM (1996) Simulated annealing for hard satisfiability problems. In: Trick M, Johnson DS (eds) Proc. Second DIMACS Algorithm Implementation Challenge on Cliques, Coloring and Satisfiability. DIMACS vol 26, pp 533–555
54. Trevisan L (1997) Approximating satisfiable satisfiability problems. In: Proc. 5th Annual European Symp. Algorithms, Graz. Springer, Berlin, pp 472–485
55. Yannakakis M (1994) On the approximation of maximum satisfiability. J Algorithms 17:475–502

Medium-Term Scheduling of Batch Processes

STACY L. JANAK, CHRISTODOULOS A. FLOUDAS
Department of Chemical Engineering,
Princeton University, Princeton, USA

Article Outline

Introduction

In multiproduct and multipurpose batch plants, different products can be manufactured via the same or a similar sequence of operations by sharing available pieces of equipment, intermediate materials, and other production resources. They are ideally suited to manufacture products that are produced in small quantities or for which the production recipe or the customer demand pattern is likely to change. The inherent operational flexibility of this type of plant provides the opportunity for increased savings through the realization of an efficient production schedule which can reduce inventories, production and transition costs, and production shortfalls.

The problem of production scheduling and planning for multiproduct and multipurpose batch plants has received a considerable amount of attention during the last two decades. Extensive reviews have been written by Reklaitis [10], Pantelides [9], Shah [11] and more recently by Floudas and Lin [4,5]. Most of the work in the area of multiproduct batch plants has dealt with either the long-term planning problem or the short-term scheduling problem. Both planning and scheduling deal with the allocation of available resources over time to perform a set of tasks required to manufacture one or more products. However, long-term planning problems deal with longer time horizons (e. g., several months or years) and are focused on higher level decisions such as timing and location of additional facilities and levels of production. In contrast, short-term scheduling models address shorter time horizons (e. g., several days) and are focused on determining detailed sequencing of various operational tasks. The area of medium-term scheduling, however, which involves medium time horizons (e. g. several weeks) and still aims to determine detailed production schedules, can result in very large-scale problems and has received much less attention in the literature.

For medium-term scheduling, relatively little work has been presented in the literature. Medium-term scheduling can be quite computationally complex, thus it is common for mathematical programming techniques to be used in their solution. The most widely employed strategy to overcome the computational difficulty is based on the idea of decomposition. The decomposition approach divides a large and complex problem, which may be computationally expensive or even intractable when formulated and solved directly as a single MILP model, to smaller subproblems, which can be solved much more efficiently. There have been a wide variety of decomposition approaches proposed in the literature. In addition to decomposition techniques developed for general forms of MILP problems, various approaches that exploit the characteristics of specific process scheduling problems have also been proposed. In most cases, the decomposition approaches only lead to suboptimal solutions, however, they substantially reduce the problem complexity and the solution time, making MILP based techniques applicable for large, real-world problems.

In this chapter, we propose an enhanced State-Task Network MILP model for the medium-term production scheduling of a multipurpose, multiproduct industrial batch plant. The proposed approach extends the work of Ierapetritou and Floudas [6] and Lin et al. [8] to consider a large-scale production facility and account for various storage policies (UIS, NIS, ZW), variable batch sizes and processing times, batch mixing and splitting, sequence-dependent changeover times, intermediate due dates, products used as raw materials, and several modes of operation. The methodology consists of the decomposition of the whole scheduling period into successive short horizons of a few days. A decomposition model is implemented to determine each short horizon and the corresponding products to be included. Then, a novel continuous-time formulation for short-term scheduling of batch processes with multiple intermediate due dates is applied to each short horizon selected, leading to a large-scale mixed-integer linear programming (MILP) problem. The scheduling model includes over 80 pieces of equipment and can take into account the processing recipes of hundreds of different products. Several characteristics of the production plant are incorporated into the scheduling model and actual plant data are used to model all parameters.

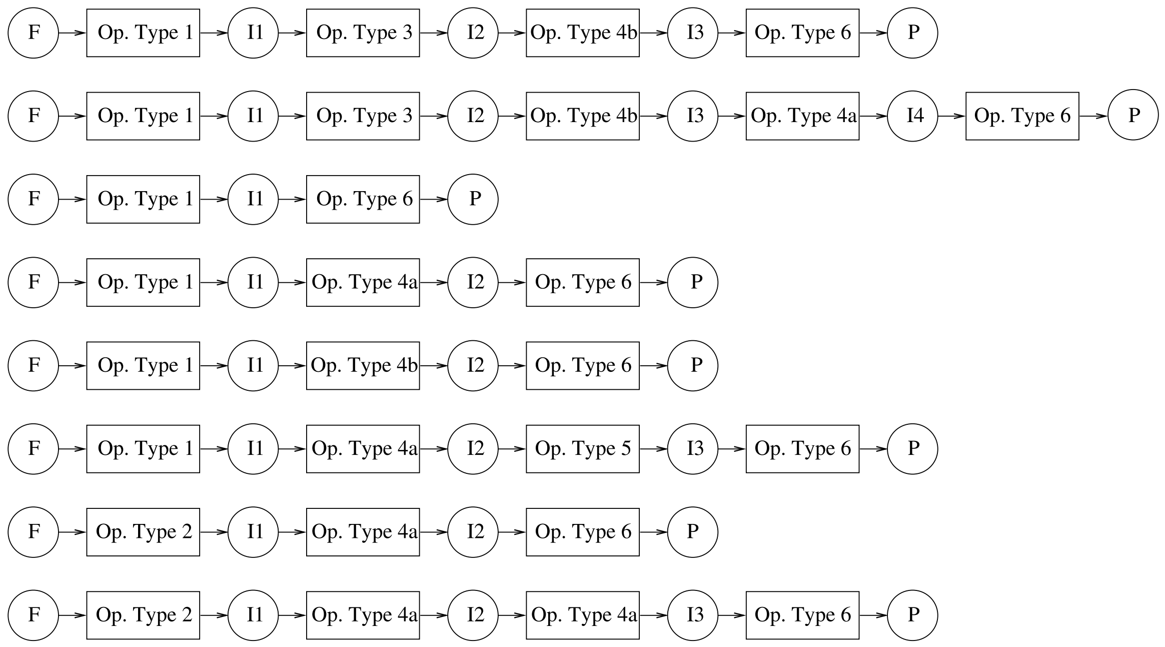
Problem Statement

In the multiproduct batch plant investigated, there are several different types of operations (or tasks) termed operation type 1 to operation type 6. The plant has many different types of units and over 80 are modeled explicitly. Hundreds of different products can be produced and for each of them, one of the processing recipes shown in Fig. 1 or a slight variation is applied. The recipes are represented in the form of State-Task Network (STN), in which the *state* node is denoted by a circle and the *task* node by a rectangle. The STN representation provides the flow of material through various tasks in the production facility to produce different types of final products and does not represent the actual connectivity of equipment in the plant.

For the first type of STN shown in Fig. 1, raw materials (or state F) are fed into a type 1 unit and undergo operation type 1 to produce an intermediate (or state I1). This intermediate then undergoes operation type 3 in a type 3 unit to produce another intermediate (or state I2). This second intermediate is then sent to a type 4b unit before the resulting intermediate material (or state I3) is sent to a type 6 unit to undergo an operation type 6 task to produce a final product (or state P). The information on which units are suitable for each product is given. All the units are utilized in a batch mode with the exception of the type 5 and 6 units, which operate in a continuous mode. The capacity limits of the type 1, type 2, and type 3 units vary from one product to another, while the capacity limits of the types 4a, 4b, 5 and 6 units are the same for all suitable products. The processing time or processing rate of each task in the suitable units is also specified. Also, some products require other products as their raw materials, creating very complicated state-task networks.

The time horizon considered for production scheduling is a few weeks or longer. Customer orders are fixed throughout the time horizon with specified amounts and due dates. There is no limitation on external raw materials and we apply the zero-wait storage condition or limited intermediate storage capacity for all materials based on actual plant data. There are two different types of products produced, category 1 and 2.

The sixth STN shown in Fig. 1 shows a special type of product, denoted as a campaign product. For this



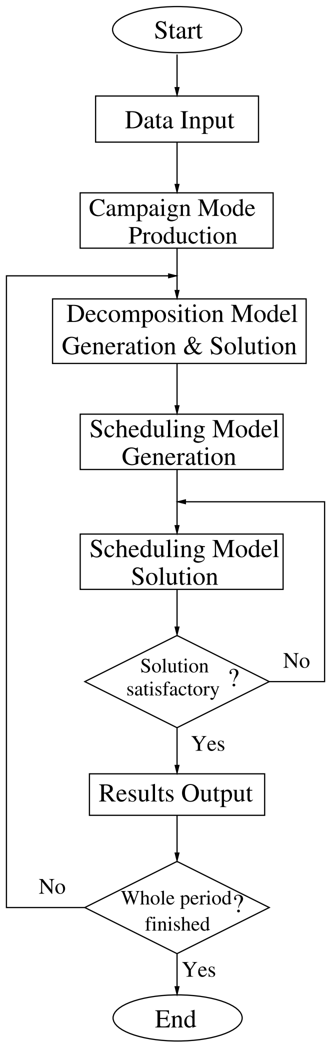
Medium-Term Scheduling of Batch Processes, Figure 1
State-task network (STN) representation of plant

type of product, raw materials are fed into up to three type 1 units and undergo operation type 1 to produce an intermediate, or state I1. This intermediate is then sent to one of two type 4a units before being processed in the type 5 unit, which is a continuous unit. Finally, the intermediate material (or state I3) is sent to a type 6 unit, producing a final campaign product (or state P). Because product changeovers in the type 5 unit can be undesirable, there was a need to introduce the ability to fix campaigns for continuous production of a single product in the type 5 unit, called campaign mode production.

Formulation

The overall methodology for solving the medium-range production scheduling problem is to decompose the large and complex problem into smaller short-term scheduling subproblems in successive time horizons [8]. The flowchart for this rolling horizon approach is shown in Fig. 2. The first step is to input relevant data into the formulation. Then, if necessary, campaign mode production is determined. Next, the overall medium-term scheduling problem is considered. A decomposition model is formulated and solved to determine the current time horizon and corresponding products that should be included in the current subproblem. According to the solution of the decomposition model, a short-term scheduling model is formulated using the information on customer orders, inventory levels, and processing recipes. The resulting MILP problem is a large-scale, complex problem which requires a large computational effort for its solution. When a satisfactory solution is determined, the relevant data is output and the next time horizon is considered. The above procedure is applied iteratively in an automatic fashion until the whole time horizon under consideration has been scheduled.

Note that the decomposition model determines how many days and products to consider in the shorter scheduling horizon subject to an upper limit on the complexity of the resulting mathematical model. Products are selected for the scheduling horizon if there is an order for the product, if the product has an order within a set amount of time into the future, if the product is used as a raw material for another product which is included, if the product was still processing in the previous scheduling horizon, or if the product is a campaign product and is included in a campaign for the current horizon.



Medium-Term Scheduling of Batch Processes, Figure 2
Flowchart of the rolling horizon approach

Models

A key component of the rolling horizon approach is the determination of the time horizon and the products which should be included for each short-term scheduling subproblem. We extend the two-level decomposition formulation of Lin et al. [8] which partitions the entire scheduling horizon into shorter subhorizons by taking into account the trade-off between demand satisfaction, unit utilization, and model complexity. In the first level, the number of days in the time horizon and the main products which should be included are determined. In the second level, additional products are added to the horizon so that each of the first-stage units, or type 1 units, are fully utilized.

Short-Term Scheduling Model

Once the decomposition model has determined the days in the time horizon and the products to be included, a novel continuous-time formulation for short-term scheduling with multiple intermediate due dates is applied to determine the detailed production schedule. This formulation is based on the models of Floudas and coworkers [6,7,8] and is expanded and enhanced in this work to take into account specific aspects of the problem under consideration. The proposed short-term scheduling formulation requires the following indices, sets, parameters and variables:

Indices:

d	days;
i	processing tasks;
j	units;
k	orders;
n	event points representing the beginning of a task;
s	states;

Sets:

D	days in the overall scheduling horizon;
D^{in}	days in the current scheduling horizon;
I	processing tasks;
I_j	tasks which can be performed in unit (j);
I_k	tasks which process order (k);
I_s^{c}	tasks which consume state (s);
I_s^{p}	tasks which produce state (s);
I^{in}	tasks which are included in the current scheduling horizon;
I^{T5}	tasks which are used to determine the type 5 unit campaign;
I^{T6b}	tasks which are used to perform operation type 6 for category 1 products;
J	units;
J_i	units which are suitable for performing task (i);
J^{p}	units which are suitable for performing only processing tasks, or operation type 1, 2, 3, and 5 tasks;
J^{T1}	units which are suitable for performing only operation type 1 tasks;

$\boldsymbol{J^{T4}}$	units which are suitable for performing only operation type 4a and 4b tasks;
$\boldsymbol{J^{T5}}$	units are which used to determine the type 5 unit campaign;
$\boldsymbol{J^{T6}}$	units which are suitable for performing only operation type 6 tasks;
$\boldsymbol{K}$	orders;
$\boldsymbol{K_i}$	orders which are processed by task (*i*);
$\boldsymbol{K_s}$	orders which produce state (*s*);
$\boldsymbol{K^{in}}$	orders which are included in the current scheduling horizon;
$\boldsymbol{N}$	event points within the time horizon;
$\boldsymbol{S}$	states;
$\boldsymbol{S_k}$	states which are used to satisfy order (*k*);
$\boldsymbol{S^{cat1}}$	states which are category 1 final products;
$\boldsymbol{S^{cat2}}$	states which are category 2 final products;
$\boldsymbol{S^{cpm}}$	states which have minimum or maximum storage limitations;
$\boldsymbol{S^{f}}$	states which are final products, after operation type 6;
$\boldsymbol{S^{i}}$	states which are intermediate products, before operation type 6;
$\boldsymbol{S^{in}}$	states which are included in the current scheduling horizon;
$\boldsymbol{S^{p}}$	states which are either final or intermediate products;
$\boldsymbol{S^{rw}}$	states which are products and are used as raw materials for other products;
$\boldsymbol{S^{st}}$	states which have no intermediate storage;
$\boldsymbol{S^{T5}}$	states which are used to determine the type 5 unit campaign;
$\boldsymbol{S^{unl}}$	states which have unlimited intermediate storage;
$\boldsymbol{S^{0}}$	states which are external raw materials;

Parameters:

$\boldsymbol{B_s^{max}}$	the maximum suitable batch size used to produce product state (*s*);
$\boldsymbol{B_s^{min}}$	the minimum suitable batch size used to produce product state (*s*);
$\boldsymbol{C}$	a large constant (e. g., 10000);
$\boldsymbol{cap_{ij}^{max}}$	maximum capacity for task (*i*) in unit (*j*);
$\boldsymbol{cap_{ij}^{min}}$	minimum capacity for task (*i*) in unit (*j*);
$\boldsymbol{dem_s}$	demand for state (*s*) in the current scheduling horizon;
$\boldsymbol{dem_s^{rw}}$	demand for raw material product state (*s*);
$\boldsymbol{dem_s^{tot}}$	total demand for state (*s*) in the overall horizon;
$\boldsymbol{duek_{ksd}}$	due date of order (*k*) for state (*s*) on day (*d*);
$\boldsymbol{ExtraTime_i}$	amount of time needed for operation type 3 task after processing task (*i*);
$\boldsymbol{FixedTime_{ij}}$	constant term of processing time for task (*i*) in unit (*j*);
$\boldsymbol{H}$	time horizon;
$\boldsymbol{mintasks}$	the minimum number of tasks that must occur in the first-stage processing units, J^{T1};
$\boldsymbol{N^{max}}$	the maximum number of event points in the scheduling horizon;
$\boldsymbol{praw_{ss'}}$	0-1 parameter to relate final product (*s*) to its raw material product (*s′*);
$\boldsymbol{price_s}$	price of state (*s*);
$\boldsymbol{prior_s}$	priority of product state (*s*);
$\boldsymbol{prior_s^{raw}}$	priority of raw material state (*s*);
$\boldsymbol{RateCT_{ij}}$	variable term of processing time for task (*i*) in unit (*j*);
$\boldsymbol{rk_{ksd}}$	amount of order (*k*) for state (*s*) on day (*d*);
$\boldsymbol{start_j}$	the time at which unit (*j*) first becomes available in the current scheduling horizon;
$\boldsymbol{stcap_s^{max}}$	maximum capacity for storage of state (*s*);
$\boldsymbol{stcap_s^{min}}$	minimum capacity for storage of state (*s*);
α	coefficient for the demand satisfaction of individual orders term;
β	coefficient for the due date satisfaction of individual orders term;
γ	coefficient for the overall demand satisfaction slack variable term;
δ	coefficient for the minimum inventory requirement in dedicated units term;
η	coefficient for the artificial demands on raw material states term;
κ	coefficient for the minimizing of binary variables term;
λ	coefficient for the minimizing of active start times term;
μ	a small constant (e. g., 0.01);
ρ_{is}^{c}	proportion of state (*s*) consumed by task (*i*);
ρ_{is}^{p}	proportion of state (*s*) produced by task (*i*);

$\tau_{ii'}$	sequence-dependent setup time between tasks (i) and (i');
ϕ	coefficient for the satisfaction of orders term;
ω	coefficient for the overall production term;

Continuous Variables:

$\boldsymbol{B(i,j,n)}$	amount of material undertaking task (i) in unit (j) at event point (n);
$\boldsymbol{D(s,n)}$	amount of state (s) delivered at event point (n);
$\boldsymbol{Df(s,n)}$	amount of state (s) delivered after the last event point;
$\boldsymbol{kD(k,s,n)}$	amount of state (s) delivered at event point (n) for order (k);
$\boldsymbol{kDf(k,s,n)}$	amount of state (s) delivered after the last event point for order (k);
$\boldsymbol{sla1(k,s,d)}$	amount of state (s) due on day (d) for order (k) that is not delivered;
$\boldsymbol{sla2(k,s,d)}$	amount of state (s) due on day (d) for order (k) that is over delivered;
$\boldsymbol{slcap(s,n)}$	amount of state (s) that is deficient in its dedicated storage unit at event point (n);
$\boldsymbol{sll(s)}$	amount of state (s) due in the current time horizon but not made;
$\boldsymbol{sll^{raw}(s)}$	amount of raw material product state (s) artificially due in the current time horizon but not made;
$\boldsymbol{slorder(k)}$	0-1 variable indicating if order (k) was met;
$\boldsymbol{slt1(k,s,d)}$	amount of time state (s) due on day (d) for order (k) is late;
$\boldsymbol{slt2(k,s,d)}$	amount of time state (s) due on day (d) for order (k) is early;
$\boldsymbol{ST(s,n)}$	amount of state (s) at event point (n);
$\boldsymbol{STF(s)}$	final amount of state (s) at the end of the current time horizon;
$\boldsymbol{STO(s)}$	initial amount of state (s) at the beginning of the current time horizon;
$\boldsymbol{T^{f}(i,j,n)}$	time at which task (i) finishes in unit (j) at event point (n);
$\boldsymbol{T^{s}(i,j,n)}$	time at which task (i) starts in unit (j) at event point (n);
$\boldsymbol{tot(s)}$	total amount of state (s) made in the current time horizon;
$\boldsymbol{tt^{s}(i,j,n)}$	starting time of the active task (i) in unit (j) at event point (n);

Binary Variables:

$\boldsymbol{wv(i,j,n)}$	assigns the beginning of task (i) in unit (j) at event point (n);
$\boldsymbol{y(i,k,n)}$	assigns the delivery of order (k) through task (i) at event point (n);

On the basis of this notation, the mathematical model for the short-term scheduling of an industrial batch plant with intermediate due dates involves the following constraints:

$$\sum_{i\in I^{\text{in}},I_j} wv(i,j,n) \le 1\,, \qquad \forall j\in J,\quad n\in N,\quad n\le N^{\max} \tag{1}$$

$$cap_{ij}^{\min}\cdot wv(i,j,n) \le B(i,j,n) \le cap_{ij}^{\max}\cdot wv(i,j,n)\,, \qquad \forall i\in I^{\text{in}},\quad j\in J_i,\quad n\in N,\quad n\le N^{\max} \tag{2}$$

$$st(s,n)=0,\quad \forall s\in S^{\text{in}},S^{\text{st}}\,, \quad s\notin S^{\text{cpm}},S^{\text{unl}},\quad n\in N,\quad n\le N^{\max} \tag{3}$$

$$st(s,n)\ge stcap_s^{\min}-slcap(s)\,, \qquad \forall s\in S^{\text{in}},S^{\text{cpm}},\quad n\in N,\quad n\le N^{\max} \tag{4}$$

$$st(s,n)\le stcap_s^{\max}\,, \qquad \forall s\in S^{\text{in}},S^{\text{cpm}},\quad n\in N,\quad n\le N^{\max} \tag{5}$$

$$\begin{aligned} ST(s,n) = {} & ST(s,n-1) - D(s,n) \\ & + \sum_{i\in I_s^{\text{p}}}\rho_{is}^{\text{p}}\sum_{j\in J_i} B(i,j,n-1) \\ & - \sum_{i\in I_s^{\text{c}}}\rho_{is}^{\text{c}}\sum_{j\in J_i} B(i,j,n)\,, \end{aligned} \qquad \forall s\in S^{\text{in}},\quad n\in N,\quad n>1,\quad n\le N^{\max} \tag{6}$$

$$\begin{aligned} ST(s,n) = {} & STO(s) - D(s,n) \\ & - \sum_{i\in I_s^{\text{c}}}\rho_{is}^{\text{c}}\sum_{j\in J_i} B(i,j,n)\,, \end{aligned} \qquad \forall s\in S^{\text{in}},\quad n\in N,\quad n=1 \tag{7}$$

$$\begin{aligned} STF(s) = {} & ST(s,n) - Df(s,n) \\ & + \sum_{i\in I_s^{\text{p}}}\rho_{is}^{\text{p}}\sum_{j\in J_i} B(i,j,n)\,, \end{aligned} \qquad \forall s\in S^{\text{in}},\quad n\in N,\quad n=N^{\max} \tag{8}$$

$$T^{\mathrm{f}}(i,j,n) = T^{\mathrm{s}}(i,j,n) + \mathit{FixedTime}_{ij} \cdot wv(i,j,n) + \mathit{RateCT}_{ij} \cdot B(i,j,n)\,, \quad \forall i \in I^{\mathrm{in}},\ j \in J^{\mathrm{p}} \cup J^{\mathrm{T6}}, J_i,\ n \in N,\ n \leq N^{\max} \tag{9}$$

$$T^{\mathrm{f}}(i,j,n) \geq T^{\mathrm{s}}(i,j,n)\,, \quad \forall i \in I^{\mathrm{in}},\ j \in J^{\mathrm{T4}}, J_i,\ n \in N,\ n \leq N^{\max} \tag{10}$$

$$T^{\mathrm{f}}(i,j,n) = H,\ \forall s \in S^{\mathrm{in}}, S^{\mathrm{st}},\ s \notin S^{\mathrm{unl}}\,, \quad i \in I^{\mathrm{in}}, I_s^{\mathrm{p}},\ j \in J^{\mathrm{T4}}, J_i,\ n \in N,\ n = N^{\max} \tag{11}$$

$$T^{\mathrm{s}}(i,j,n+1) \geq T^{\mathrm{f}}(i,j,n) + \mathit{ExtraTime}_i \cdot wv(i,j,n)\,, \quad \forall i \in I^{\mathrm{in}},\ j \in J_i,\ n \in N,\ n < N^{\max} \tag{12}$$

$$T^{\mathrm{s}}(i,j,n+1) \geq T^{\mathrm{f}}(i',j,n) + (\tau_{i'i} + \mathit{ExtraTime}_{i'}) \cdot wv(i',j,n) - H[1 - w(i',j,n)]\,, \quad \forall j \in J,\ i,i' \in I^{\mathrm{in}}, I_j,\ i \neq i',\ n \in N,\ n < N^{\max} \tag{13}$$

$$T^{\mathrm{s}}(i,j,n+1) \geq T^{\mathrm{f}}(i',j',n) - H[1 - wv(i',j',n)]\,, \quad \forall s \in S^{\mathrm{in}},\ i \in I^{\mathrm{in}}, I_s^{\mathrm{c}},\ i' \in I^{\mathrm{in}}, I_s^{\mathrm{p}}\,,\ j \in J_i,\ j' \in J_{i'},\ j \neq j',\ n \in N,\ n < N^{\max} \tag{14}$$

$$T^{\mathrm{s}}(i,j,n+1) \leq T^{\mathrm{f}}(i',j',n) + H[2 - wv(i',j',n) - wv(i,j,n+1)]\,, \quad \forall s \in S^{\mathrm{in}}, S^{\mathrm{st}}, s \notin S^{\mathrm{unl}},\ i \in I^{\mathrm{in}}, I_s^{\mathrm{c}},\ i' \in I^{\mathrm{in}}, I_s^{\mathrm{p}}\,,\ j \in J_i,\ j' \in J_{i'},\ j \neq j',\ n \in N,\ n < N^{\max} \tag{15}$$

The allocation constraints in (1) express the requirement that for each unit (j) and at each event point (n), only one of the tasks that can be performed in the unit (i. e., $i \in I_j$) should take place. The capacity constraints in (2) express the requirement for the batchsize of a task (i) processing in a unit (j) at event point (n), $B(i,j,n)$, to be greater than the minimum amount of material, $cap_{ij}^{\min}$, and less than the maximum amount of material, $cap_{ij}^{\max}$, that can be processed by task (i) in unit (j). The storage constraints in (3) enforce that those states with no intermediate storage have to be consumed by some processing task or storage task immediately after they are produced. Constraints (4) represent the minimum required storage for state (s) in a dedicated storage tank where this amount can be violated, if necessary, by an amount *slcap*(s) which is penalized in the objective function. Constraints (5) represent the maximum available storage capacity for state (s) based on the maximum storage capacity of the dedicated storage tank. According to the material balance constraints in (6), the amount of material of state (s) at event point (n) is equal to that at event point ($n-1$) increased by any amounts produced at event point ($n-1$), decreased by any amounts consumed at event point (n), and decreased by the amount required by the market at event point (n), $D(s,n)$. Constraints (7)–(8) represent the material balance on state (s) at the first and last event points, respectively. The duration constraints in (9) represent the relationship between the starting and finishing times of task (i) in unit (j) at event point (n) for all processing tasks (i. e., J^{p}) and all operation type 6 tasks (i. e., J^{T6}) where $\mathit{FixedTime}_{ij}$ are the fixed processing times for batch tasks and zero for continuous tasks and RateCT_{ij} are the inverse of processing rates for continuous tasks and zero for batch tasks, respectively. Constraints (10) also represent the relationship between the starting and finishing times of task (i) in unit (j) at event point (n), but for operation type 4a and 4b tasks (i. e., J^{T4}). They do not impose exact durations for tasks in these units but just enforce that all tasks must end after they start. Constraints (11) are written only for tasks in units which are processing a nonstorable state (i. e., S^{st} and not S^{unl}) and enforce that task (i) taking place at the last event point (n) must finish at the end of the horizon.

The sequence constraints in (12) state that task (i) starting at event point ($n+1$) should start after the end of the same task performed in the same unit (j) which has finished at the previous event point, (n) where extra time is added after task (i) at event point (n), if necessary. The constraints in (13) are written for tasks (i) and (i') that are performed in the same unit (j) at event points ($n+1$) and (n), respectively. If both tasks take place in the same unit, they should be at most consecutive. The third set of sequence constraints in (14) relate tasks (i) and (i') which are performed in different units (j) and (j') but take place consecutively according to the production recipe. The zero-wait constraints in (15) are written for different tasks (i) and (i') that take place consecutively with the intermediate state (s)

having no possible intermediate storage and thus subject to the zero-wait condition.

$$\sum_{\substack{i\in I^{\text{in}},\\ I_k, I^{\text{T6b}}}} \sum_{\substack{n\in N\\ n\le N^{\max}}} y(i,k,n) + slorder(k) \ge 1\,, \quad \forall k \in K^{\text{in}} \tag{16}$$

$$\sum_{i\in I^{\text{in}}, I_k, I^{\text{T6b}}} \sum_{\substack{n\in N\\ n\le N^{\max}}} y(k,i,n) \le \sum_{\substack{s\in S^{\text{in}}, S^{\text{cat1}}\\ sl_s>0}} \sum_{\substack{d\in D^{\text{in}}\\ rk_{ksd}>0}} \left\lceil \frac{rk_{ksd}}{B_s^{\min}} \right\rceil, \quad \forall k \in K^{\text{in}} \tag{17}$$

$$\sum_{k\in K^{\text{in}}, K_i} \sum_{j\in J_i} suit_{ij} \cdot y(i,k,n) \ge \sum_{j\in J_i, J^{\text{T6}}} wv(i,j,n)\,, \quad \forall i \in I^{\text{in}}, I^{\text{T6b}}, \quad n\in N, \quad n \le N^{\max} \tag{18}$$

$$\sum_{k\in K^{\text{in}}, K_i} y(i,k,n) \le \sum_{j\in J_i, J^{\text{T6}}} wv(i,j,n)\,, \quad \forall i \in I^{\text{in}}, I^{\text{T6b}}, \quad n\in N, \quad n \le N^{\max} \tag{19}$$

$$kD(k,s,n+1) + kDf(k,s,n+1) \ge \sum_{j\in J_i} B(i,j,n) - C\cdot(1-y(i,k,n))\,, \quad \forall s \in S^{\text{in}}, S^{\text{cat1}}, \quad k\in K^{\text{in}}, K_s\,, \quad i \in I_k, I^{\text{T6b}}, \quad n\in N, \quad n < N^{\max} \tag{20}$$

$$D(s,n) = \sum_{k\in K^{\text{in}}, K_s} kD(k,s,n)\,, \quad \forall s \in S^{\text{in}}, S^{\text{cat1}}, \quad n\in N, \quad n \le N^{\max} \tag{21}$$

$$Df(s,n) = \sum_{k\in K^{\text{in}}, K_s} kDf(k,s,n)\,, \quad \forall s \in S^{\text{in}}, S^{\text{cat1}}, \quad n\in N, \quad n = N^{\max} \tag{22}$$

$$\sum_{\substack{n\in N\\ n<N^{\max}}} \left[kD(k,s,n+1) + kDf(k,s,n+1)\right] + sla1(k,s,d) \ge rk_{ksd}\,, \quad \forall s \in S^{\text{in}}, S^{\text{cat1}}, \quad k\in K^{\text{in}}, K_s\,, \quad d\in D^{\text{in}}, rk_{ksd} > 0 \tag{23}$$

$$\sum_{\substack{n\in N\\ n<N^{\max}}} \left[kD(k,s,n+1) + kDf(k,s,n+1)\right] + stf(s) - sla2(k,s,d) \le rk_{ksd}\,, \quad \forall s \in S^{\text{in}}, S^{\text{cat1}}, \quad k\in K^{\text{in}}, K_s\,, \quad d\in D^{\text{in}}, rk_{ksd} > 0 \tag{24}$$

$$T^{\text{f}}(i,j,n) - slt1(k,s,d,n) \le duek(k,s,d) + H\cdot(2 - wv(i,j,n) - y(i,k,n))\,, \quad \forall s \in S^{\text{in}}, S^{\text{cat1}}, \quad k\in K^{\text{in}}, K_s, \quad i \in I^{\text{in}}, I_k, I^{\text{T6b}}, \quad j\in J_i, \quad n\in N, n\le N^{\max}, \quad d\in D^{\text{in}}, rk_{ksd} > 0 \tag{25}$$

$$T^{\text{f}}(i,j,n) + slt2(k,s,d,n) \ge (duek(k,s,d) - 24) - H\cdot(2 - wv(i,j,n) - y(i,k,n))\,, \quad \forall s \in S^{\text{in}}, S^{\text{cat1}}, \quad k\in K^{\text{in}}, K_s, \quad i \in I^{\text{in}}, I_k, I^{\text{T6b}}, \quad j\in J_i, \quad n\in N, n\le N^{\max}, \quad d\in D^{\text{in}}, rk_{ksd} > 0 \tag{26}$$

The order satisfaction constraints in (16)–(23) are written to ensure that all orders for category 1 products are met on-time and with the required amount. Both under and overproduction as well as early and late production are represented with slack variables that are penalized in the objective function. Note that these constraints can be modified to represent different requirements for production, if desired. Constraints (16) try to ensure that each order (k) is met at least one time with an operation type 6 task (i), where task (i) is suitable for order (k) if $i \in I_k$ and is a operation type 6 task for a category 1 product if $i \in I^{\text{T6b}}$. Similarly, constraints (17) enforce the condition that each order (k) for category 1 product state (s) on day (d) can be met with at most $\lceil rk_{ksd} / B_s^{\min} \rceil$ tasks. Constraints (18) and (19) link the delivery of order (k) through task (i) at event point (n) to the beginning of task (i) in any suitable unit (j) at event point (n) so that every category 1 operation type 6 task must be linked to at least one order delivery and vice versa. Thus, constraint (18) enforces that if a binary variable is activated for operation type 6 task (i), then at least one order delivery must be activated. Similarly, constraint (19) ensures that if no binary variables are activated for operation type 6 task (i) at event point (n), then no delivery variables can be activated. Constraints (20) relate the individual order delivery variables to the batch-size of the operation type 6 task used to satisfy the order. If an order (k) is met by task (i) at event point (n) (i. e., $y(i,k,n) = 1$), then at least one operation type 6 task is active for task (i) at event point (n) and thus at least one $B(i,j,n)$ variable is greater than zero. Constraints (21) and (22) relate the individual order delivery variables to the overall delivery variables used in the material balance constraints.

Constraints (23) and (24) determine the under and overproduction, respectively, of order (k) for state (s) on day (d). Constraints (23) try to enforce the individual order delivery variables to exceed the amount due for order (k) (i. e., rk_{ksd}) where slack variables $sla1(k,s,d)$ are activated in the case of underproduction. Similarly, constraints (24) try to enforce the individual order delivery variables plus any amount of the product state left at the end of the horizon not to exceed the amount due for order (k) where slack variables $sla2(k,s,d)$ are activated in the case of overproduction. Constraints (25) and (26) determine the late and early production, respectively, of order (k) for state (s) on day (d). Constraints (25) try to enforce the finishing time of task (i) used to satisfy order (k) at event point (n) to be less than the due date of order (k) where slack variables $slt1(k,s,d,n)$ are activated in the case of late production. Similarly, constraints (26) try to enforce the finishing time of task (i) used to fulfill order (k) at event point (n) to be greater than the beginning of the day (d) on which the order is due (i. e., $duek(k,s,d)-24$). Otherwise, slack variables $slt2(k,s,d,n)$ are activated indicating early production.

$$tot(s) = stf(s) + \sum_{\substack{n\in N\\ n\le N^{\max}}} \left[D(s,n) + Df(s,n)\right], \quad \forall s \in S^{\text{in}} \tag{27}$$

$$\sum_{\substack{n\in N\\ n\le N^{\max}}} \left[D(s,n) + Df(s,n)\right] + sll(s) \ge dem_s, \quad \forall s \in S^{\text{in}}, S^{\text{cat1}} \tag{28}$$

$$tot(s) + sll(s) \ge dem_s, \quad \forall s \in S^{\text{in}}, S^{\text{cat2}} \tag{29}$$

$$\sum_{i\in I^{\text{in}}, I_s^{\text{p}}} \sum_{j\in J_i} \sum_{\substack{n\in N\\ n\le N^{\max}}} B(i,j,n) + sll^{\text{raw}}(s) \ge dem_s^{\text{raw}}, \quad \forall s \in S^{\text{in}}, S^{\text{rw}},\; s' \in S^{\text{in}}, S^{\text{f}},\; praw_{s's} > 0,\; dem_{s'} > 0,\; dem_s^{\text{raw}} > 0 \tag{30}$$

Constraints (27)–(29) are used to determine the overall underproduction for both category 1 and 2 products in the current time horizon. First, constraints (27) determine the total production for all product states (s) (i. e., $tot(s)$) in the current horizon. Then, constraints (28) sum the overall delivery variables for category 1 products and activate the slack variables $sll(s)$ if the sum does not exceed the demand for category 1 product state (s). Similarly, constraints (29) calculate the amount of underproduction (i. e., $sll(s)$) for category 2 product state (s) based on it's overall demand in the time horizon. The slack variable $sll(s)$ is then penalized in the objective function where category 1 and 2 products can be penalized at different weights. Constraints (30) determine the amount of underproduction for intermediate product states (s) that are needed as raw materials for final product states (s').

The bound constraints are used to impose lower and upper bounds on the continuous variables including slack variables. They are also used to fix some binary and continuous variables to be zero when necessary.

$$T^{\text{f}}(i,j,n) \ge start_j, \quad \forall i \in I,\; j \in J_i,\; n \in N$$
$$T^{\text{s}}(i,j,n) \ge start_j, \quad \forall i \in I,\; j \in J_i,\; n \in N$$
$$T^{\text{f}}(i,j,n) \le H, \quad \forall i \in I,\; j \in J_i,\; n \in N$$
$$T^{\text{s}}(i,j,n) \le H, \quad \forall i \in I,\; j \in J_i,\; n \in N$$
$$STO(s) = 0, \quad \forall s \notin S^0$$
$$STF(s) \le dem_s^{\text{tot}}, \quad \forall s \in S^{\text{f}}$$
$$tot(s) \le dem_s^{\text{tot}}, \quad \forall s \in S^{\text{f}}$$
$$D(s,n), Df(s,n) = 0, \quad \forall s \notin S^{\text{p}} \;or\; n \in N, n > N^{\max}$$
$$D(s,n), Df(s,n) \le \sum_{d\in D^{\text{in}}} \sum_{k\in K^{\text{in}}} rk(k,s,d), \quad \forall s \in S^{\text{p}},\; n \in N, n \le N^{\max}$$
$$kD(k,s,n), kDf(k,s,n) = 0, \quad \forall k \notin K^{\text{in}} \;or\; s \notin S_k \;or\; n \in N, n > N^{\max}$$
$$kD(k,s,n), kDf(k,s,d) \le \sum_{d\in D^{\text{in}}} rk(k,s,d), \quad \forall s \in S_k,\; n \in N,\; n \le N^{\max}$$
$$slcap(s,n) \le stcap_s^{\min}, \quad \forall s \in S^{\text{cpm}}$$
$$sla1(k,s,n), sla2(k,s,n) = 0, \quad \forall k \notin K^{\text{in}} \;or\; s \notin S_k \;or\; d \notin D^{\text{in}} \;or\; rk(k,s,d) = 0$$
$$sla1(k,s,n) \le rk(k,s,d), \quad \forall k \in K^{\text{in}},\; s \in S_k,\; d \in D^{\text{in}}$$

$$
\begin{aligned}
&sla2(k,s,n) \leq dem_s^{\mathrm{tot}}, \quad \forall k \in K^{\mathrm{in}}, \\
&\quad s \in S_k, \quad d \in D^{\mathrm{in}} \\
&slt1(k,s,d,n), slt2(k,s,d,n) = 0, \quad \forall k \notin K^{\mathrm{in}} \\
&\quad or\ s \notin S_k\ or\ d \notin D^{\mathrm{in}}\ or \\
&rk(k,s,d) = 0\ or\ n \in N, \quad n > N^{\max} \\
&slt1(k,s,d,n) \leq H - duek(k,s,d), \quad \forall k \in K^{\mathrm{in}}, \\
&\quad s \in S_k, \quad d \in D^{\mathrm{in}}, \quad n \in N, \quad n \leq N^{\max} \\
&slt2(k,s,d,n) \leq duek(k,s,d), \quad \forall k \in K^{\mathrm{in}}, \\
&\quad s \in S_k, \quad d \in D^{\mathrm{in}}, \quad n \in N, \quad n \leq N^{\max} \\
&sll(s) \leq dem_s, \quad \forall s \in S^{\mathrm{p}} \\
&sll^{\mathrm{raw}}(s) \leq dem_s^{\mathrm{raw}}, \quad \forall s \in S^{\mathrm{rw}} \\
&wv(i,j,n), B(i,j,n) = 0, \quad \forall i \notin I^{\mathrm{in}}\ or\ j \notin J_i \\
&\quad or\ n \in N, \quad n > N^{\max} \\
&D(s,n), Df(s,n) = 0, \quad \forall s \notin S^{\mathrm{in}} \\
&\quad or\ n \in N, \quad n > N^{\max}
\end{aligned} \tag{31}
$$

There are several different objective functions that can be employed with a general short-term scheduling problem. In this work, we maximize the sale of final products while penalizing several other terms including the slack variables introduced previously. The overall objective function is as follows:

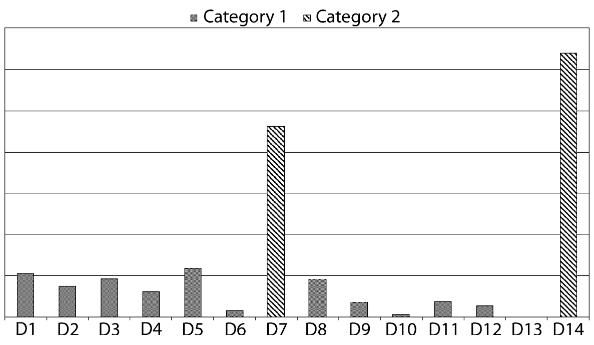
$$
\begin{aligned}
&Max\ \ \omega \cdot \sum_{s \in S^{\mathrm{in}}, S^{\mathrm{p}}} price_s \cdot tot(s) \\
&\quad - \lambda \cdot \sum_{i \in I^{\mathrm{in}}} \sum_{j \in J_i} \sum_{\substack{n \in N \\ n \leq N^{\max}}} tts(i,j,n) \\
&\quad - \kappa \cdot \left[\sum_{i \in I^{\mathrm{in}}} \sum_{j \in J_i} \sum_{\substack{n \in N \\ n \leq N^{\max}}} wv(i,j,n) \right. \\
&\qquad \left. + \sum_{k \in K^{\mathrm{in}}} \sum_{i \in I_k} \sum_{\substack{n \in N \\ n \leq N^{\max}}} y(i,k,n) \right] \\
&\quad - \gamma \cdot \sum_{s \in S^{\mathrm{f}}} prior_s \cdot sll(s) - \phi \cdot \sum_{k \in K^{\mathrm{in}}} slorder(k) \\
&\quad - \alpha \cdot \left[\sum_{k \in K^{\mathrm{in}}} \sum_{s \in S^{\mathrm{cat1}}} \sum_{d \in D^{\mathrm{in}}} sla1(k,s,d) \right. \\
&\qquad \left. + \mu \cdot sla2(k,s,d) \right] \\
&\quad - \beta \cdot \left[\sum_{k \in K^{\mathrm{in}}} \sum_{s \in S^{\mathrm{cat1}}} \sum_{d \in D^{\mathrm{in}}} \sum_{\substack{n \in N \\ n \leq N^{\max}}} \mu \cdot slt1(k,s,d,n) \right. \\
&\qquad \left. + slt2(k,s,d,n) \right] \\
&\quad - \eta \cdot \sum_{s \in S^{\mathrm{rw}}} prior_s^{\mathrm{raw}} \cdot sll^{\mathrm{raw}}(s) \\
&\quad - \delta \cdot \sum_{s \in S^{\mathrm{cpm}}} \sum_{\substack{n \in N \\ n \leq N^{\max}}} slcap(s,n)
\end{aligned} \tag{32}
$$

where each of the coefficients is used to balance the relative weight of each term in the overall objective function. The first term is the maximization of the value of the final products and is the main term of the objective function. The second term seeks to minimize the sum of the starting times of all active processing tasks. This is done to encourage all tasks to start as early as possible in the scheduling horizon. Note that this results in a bilinear term which can replaced with an equivalent linear term and set of constraints [3]. The third term seeks to minimize the number of active binary variables in the final production schedule. The fourth term seeks to minimize the slack variable that is activated when product state (s) does not meet its overall demand for the time horizon. Coefficient $prior_s$ allows the ability to assign different weights to different product states. The fifth term minimizes the number of category 1 orders (k) that are not filled in the time horizon. The sixth term minimizes the amount of over and underproduction of orders for category 1 products in the time horizon where the coefficient μ allows over and underproduction to be penalized by different amounts. The seventh term seeks to minimize the amount of early and late production of orders for category 1 products due in the time horizon where the coefficient μ allows early and late production to be penalized to different degrees. The eighth term minimizes the slack variables activated when insufficient raw material state (s) is produced during the time horizon where $prior_s^{\mathrm{raw}}$ allows different states to be penalized by different amounts. The ninth, and final, term seeks to minimize the slack variables activated when insufficient intermediate state (s) is stored in its dedicated storage tank at each event point. Typical values for each of the coefficients are as follows: $\omega = 1$, $\lambda = 1, \kappa = 10, \gamma = 1000, \phi = 1000, \alpha = 2000$, $\beta = 500, \mu = 0.01, \eta = 50, \delta = 10$.

Cases

In this section, an example problem is presented to demonstrate the effectiveness of the rolling horizon framework. The example utilizes the proposed framework to determine the medium range production schedule of an industrial batch plant for a two-week time period which satisfies customer orders for various products distributed throughout the time period. The example is implemented with GAMS 2.50 [1] and solved using CPLEX 9.0 [2] with a 3.20 GHz Linux workstation. The dual simplex method is used with best-bound search and strong branching. A relative optimality tolerance equal to 0.001% was used as the termination criterion along with a three hour time limit and an integer solution limit of 40.

The distribution of demands for the entire two-week time period is shown in Fig. 3 where the amounts are shown in relative terms. There are two categories of products, category 1 and 2, and a total of 67 different products have demands. There are two different campaign products that can be scheduled for campaign mode production and an additional eight intermediate products are used to make final products, even though they do not have demands. It is assumed that no final products are available at the beginning of the time horizon although some intermediate materials are available. Also, we assume no limitation on external raw materials and the zero-wait condition is applied to all intermediate materials unless they are used as raw materials for other final products. In this case, unlimited intermediate storage is allowed. Note that finite intermediate storage is effectively modeled for those intermediates that have a dedicated storage task with a given capacity limit. In addition, there are two types of connections made between each consecutive short-term scheduling horizon in the rolling horizon framework: the initial available time for each unit and the inventory of intermediate materials.



Medium-Term Scheduling of Batch Processes, Figure 3
Distribution of demands

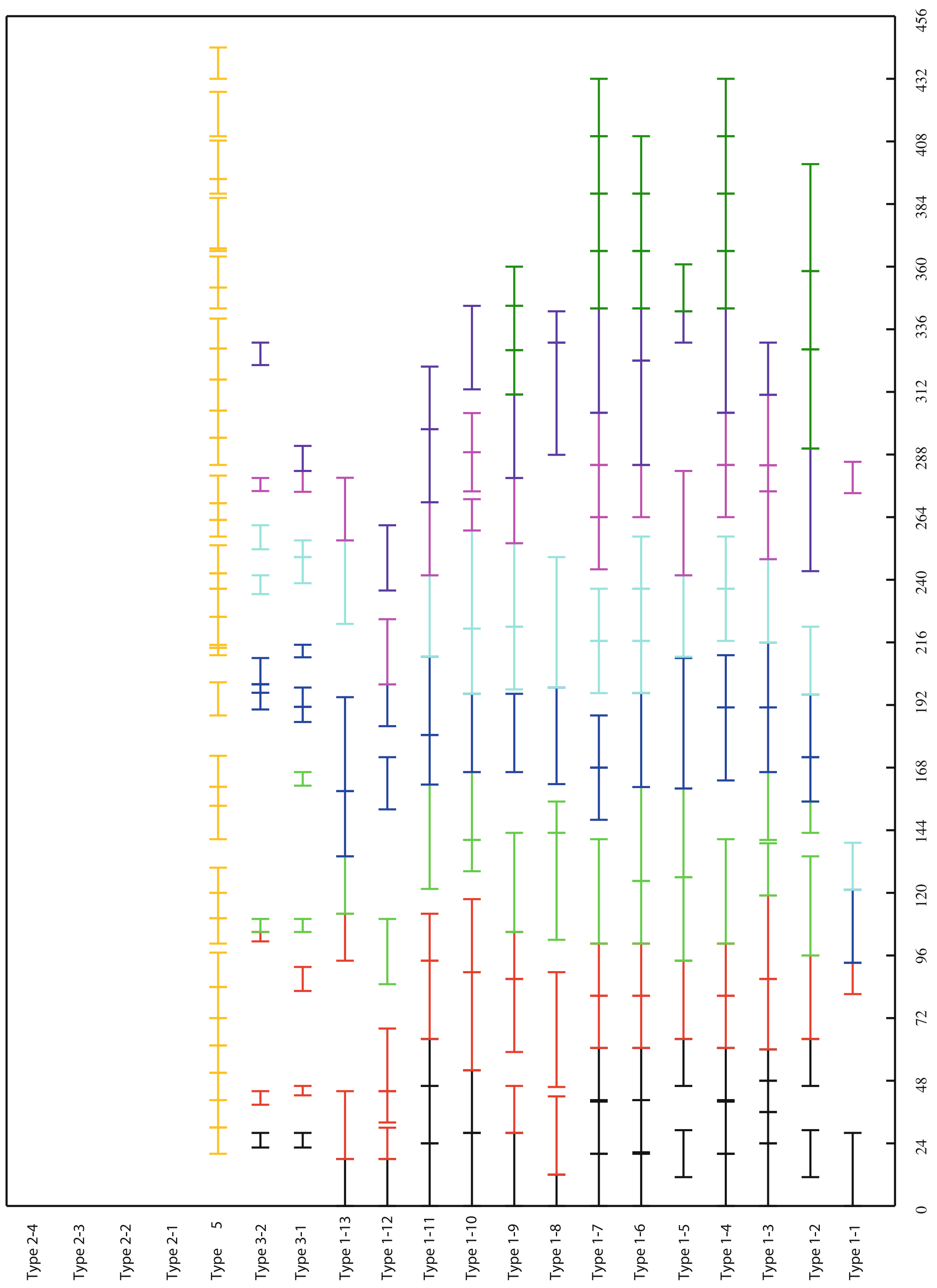
Case 1: Nominal Run without Campaign Mode Production

The example problem considers the production scheduling of an industrial batch plant where no type 5 unit campaign is imposed. Instead, demands for both campaign products are created throughout the time horizon with a total demand for each product equal to the production that would be imposed by a campaign. The total time period is 19 days, from D0 to D18. The rolling horizon framework decomposes the time horizon into 8 individual subhorizons, each with its own products and demands. The results of the decomposition for each time horizon can be seen in Table 1.

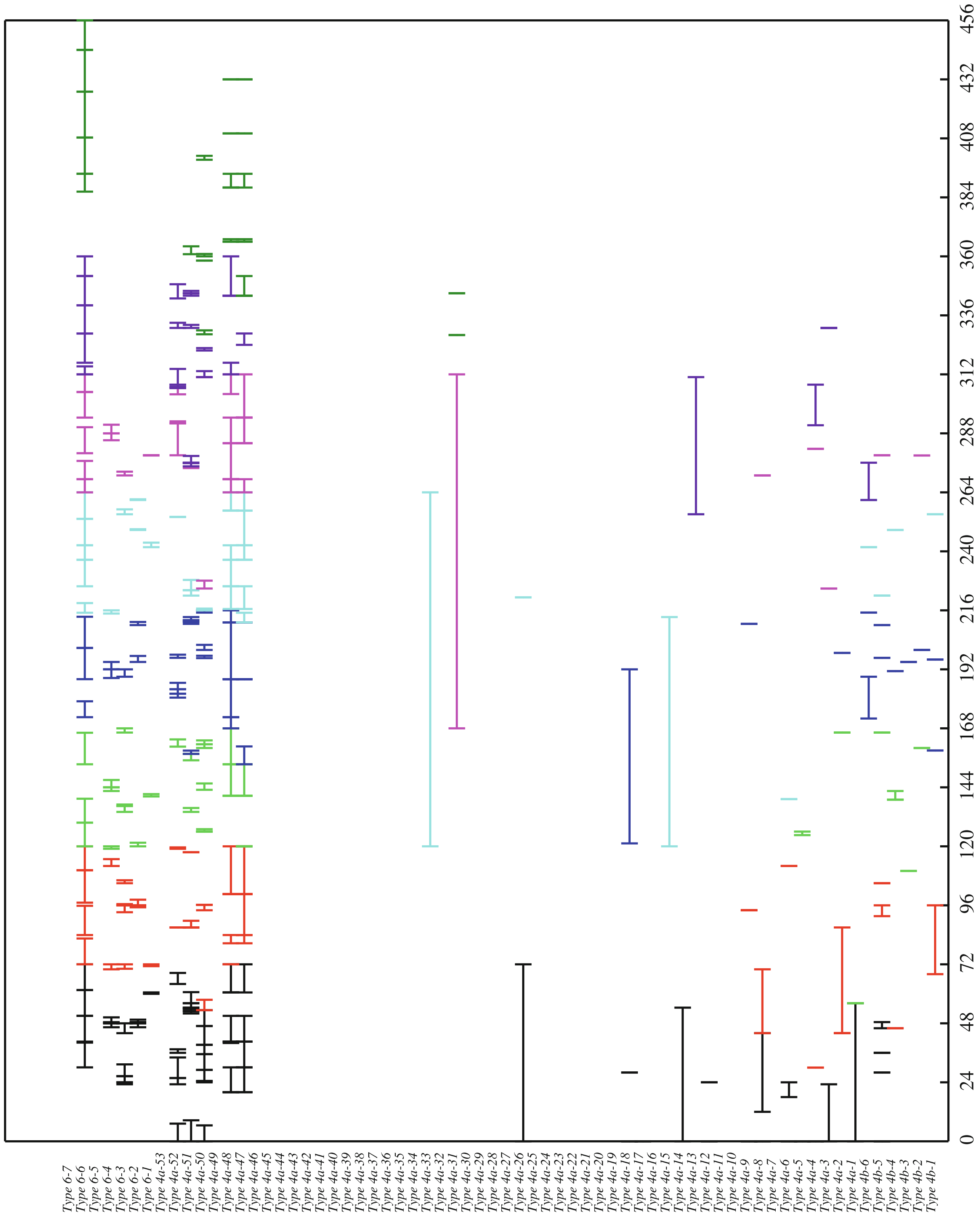
The final production schedule for the entire time period can be seen in Fig. 4 and 5 where the processing units (operation type 1, 2, 3, and 5) are shown in the first figure and the other units (operation type 4a, 4b, and 6) are shown in the second. Each short-term scheduling horizon is represented with a different color beginning with black for the first horizon, red for the second horizon, green for the third horizon, etc. The model and solution statistics for each short-term

Medium-Term Scheduling of Batch Processes, Table 1
Decomposition results for case 1

	Days	Main Products	Additional Products
H1	D0–D2	27	2
H2	D3–D4	31	0
H3	D5–D6	50	0
H4	D7–D8	49	0
H5	D9–D10	37	0
H6	D11–D12	49	0
H7	D13–D14	54	0
H8	D15–D18	45	0



Medium-Term Scheduling of Batch Processes, Figure 4
Overall production schedule for processing units for case 1



Medium-Term Scheduling of Batch Processes, Figure 5
Overall production schedule for non-processing units for case 1

Medium-Term Scheduling of Batch Processes, Table 2
Model and solutions statistics for case 1

	Days	Event Points	Objective Function	Binary Variables	Continuous Variables	Constraints
H1	D0–D2	8	14, 001.69	4880	33,064	187,833
H2	D3–D4	6	4135.24	3660	24,923	125,374
H3	D5–D6	6	−105, 854.81	5478	32,621	258,852
H4	D7–D8	6	−5496.19	5376	32,167	255,696
H5	D9–D10	6	−15, 352.37	4296	27,613	175,939
H6	D11–D12	6	−11, 326.13	5490	32,637	272,802
H7	D13–D14	6	−19, 401.39	5568	32,955	282,632
H8	D15–D18	10	−37, 054.00	7430	46,827	321,162

scheduling horizon can be seen in Table 2 where each horizon runs for the time limit of three hours.

The total demand for the entire 14-day period is 2323.545 and the total production is 2744.005, where 51.674 of the demands are not met. The production schedules obtained satisfy demands for almost all the products, though some due dates are relaxed, and also produce 18.10% more material than the demands require. Many of the processing units are not fully utilized, as shown in Table 3, indicating the potential for even more production in the given time period. Also, note that the processing units become more idle towards the end of the overall time horizon. This is because no demands are specified for the days following day D14 including days D15 to D18. Additional demands at the end of the overall time horizon or in the following days would generate a more heavily utilized production schedule.

Medium-Term Scheduling of Batch Processes, Table 3
Unit utilization statistics for case 1

Unit	Time Used (h)	TimeLeft (h)	Percent Utilized
Type 1–1	98.00	358.00	21.49%
Type 1–2	341.00	115.00	74.78%
Type 1–3	329.60	126.40	72.15%
Type 1–4	396.00	60.00	80.92%
Type 1–5	283.20	172.80	62.06%
Type 1–6	402.00	54.00	88.16%
Type 1–7	408.00	48.00	89.47%
Type 1–8	281.00	175.00	61.62%
Type 1–9	322.00	134.00	70.61%
Type 1–10	322.20	133.80	70.66%
Type 1–11	312.20	143.80	68.46%
Type 1–12	177.00	279.00	38.82%
Type 1–13	201.00	255.00	44.08%
Type 5	362.04	93.96	79.39%

Conclusions

In this paper, a *unit-specific* event-based continuous-time formulation is presented for the medium-term production scheduling of a large-scale, multipurpose industrial batch plant. The proposed formulation takes into account a large number of processing recipes and units and incorporates several features including various storage policies (UIS, NIS, ZW), variable batch sizes and processing times, batch mixing and splitting, sequence-dependent changeover times, intermediate due dates, products used as raw materials, and several modes of operation. The scheduling horizon is several weeks or longer, however longer time periods can be addressed with the proposed framework. A key feature of the proposed formulation is the use of a decomposition model to split the overall scheduling horizon into smaller subhorizons which are scheduled in a sequential fashion. Also, new constraints are added to the short-term scheduling model in order to model the delivery of orders at intermediate due dates. The effectiveness of the proposed approach is demonstrated with an industrial case study. Results indicate that the rolling horizon approach is effective at solving large-scale, medium-term production scheduling problems.

References

1. Brooke A, Kendrick D, Meeraus A, Raman R (2003) GAMS: A User's Guide. South San Francisco
2. CPLEX (2005) ILOG CPLEX 9.0 User's Manual
3. Floudas CA (1995) Nonlinear and Mixed-Integer Optimization. Oxford University Press, Oxford

4. Floudas CA, Lin X (2004) Continuous-Time versus Discrete-Time Approaches for Scheduling of Chemical Processes: A Review. Comput Chem Eng 28:2109
5. Floudas CA, Lin X (2005) Mixed Integer Linear Programnubg in Process Scheduling: Modeling, Algorithms, and Applications. Ann Oper Res 139:131
6. Ierapetritou MG, Floudas CA (1998) Effective Continuous-Time Formulation for Short-Term Scheduling: 1. Multipurpose Batch Processes. Ind Eng Chem Res 37:4341
7. Janak SL, Lin X, Floudas CA (2004) Enhanced Continuous-Time Unit-Specific Event-Based Formulation for Short-Term Scheduling of Multipurpose Batch Processes: Resource Constraints and Mixed Storage Policies. Ind Eng Chem Res 43:2516
8. Lin X, Floudas CA, Modi S, Juhasz NM (2002) Continuous-Time Optimization Approach for Medium-Range Production Scheduling of a Multiproduct Batch Plant. Ind Eng Chem Res 41:3884
9. Pantelides CC (1993) Unified Frameworks for Optimal Process Planning and Scheduling. In: Rippin DWT, Hale JC, Davis J (eds) Proceedings of the Second International Conference on Foundations of Computer-Aided Process Operations. CACHE. Crested Butte, Austin, pp 253–274
10. Reklaitis GV (1992) Overview of Scheduling and Planning of Batch Process Operations. In: Presented at NATO Advanced Study Institute – Batch Process Systems Engineering. Antalya
11. Shah N (1998) Single- And Multisite Planning and Scheduling: Current Status and Future Challenges. In: Pekny JF, Blau GE (eds) Proceedings of the Third International Conference on Foundations of Computer-Aided Process Operations. CACHE-AIChE. Snowbird, New York, pp 75–90

Metaheuristic Algorithms for the Vehicle Routing Problem

YANNIS MARINAKIS
Department of Production Engineering and Management, Decision Support Systems Laboratory, Technical University of Crete, Chania, Greece

MSC2000: 90B06, 90C59

Article Outline

Introduction

The **vehicle routing problem (VRP)** or the **capacitated vehicle routing problem (CVRP)** is often described as a problem in which vehicles based at a central depot are required to visit geographically dispersed customers in order to fulfill known customer demands. Let $G = (V, E)$ be a graph where $V = \{i_0, i_1, i_2, \dots i_n\}$ is the vertex set ($i_i = i_0$ refers to the depot and the customers are indexed $i_i = i_1, \dots, i_n$) and $E = \{(i_l, i_{l_1})\colon i_l, i_{l_1} \in V\}$ is the edge set. Each customer must be assigned to exactly one of the k vehicles and the total size of deliveries for customers assigned to each vehicle must not exceed the vehicle capacity (Q_k). If the vehicles are homogeneous, the capacity for all vehicles is equal and denoted by Q. A demand q_{i_l} and a service time st_{i_l} are associated with each customer node i_l. The travel cost between customers i_l and i_{l_1} is $c_{i_l i_{l_1}}$. The problem is to construct a low cost, feasible set of routes – one for each vehicle. A route is a sequence of locations that a vehicle must visit along with the indication of the service it provides. The vehicle must start and finish its tour at the depot. The most important variants of the vehicle routing problem can be found in [12,13,39,54,84].

The vehicle routing problem was first introduced by Dantzig and Ramser [21]. As it is an NP-hard problem, the instances with a large number of customers cannot be solved in optimality within reasonable time. Due to the general inefficiency of the exact methods and their inability to solve large scale VRP instances, a large number of approximation techniques have been proposed. These techniques are classified into two main categories, the classical heuristics that were developed mostly between 1960 and 1990 and the metaheuristics that were developed in the last fifteen years.

In the 1960s and 1970s the first attempts to solve the vehicle routing problem focused on route building, route improvement and two-phase heuristics. In the 1980s a number of mathematical programming procedures were proposed for the solution of the problem. The most important of them can be found in [6,18,19,22,28,29,33,62,88].

Metaheuristic Algorithms for the Vehicle Routing Problem

The last fifteen years an incremental amount of metaheuristic algorithms have been proposed. Simulated annealing, genetic algorithms, neural networks, tabu search, ant algorithms, together with a number of hybrid techniques are the main categories of the metaheuristic procedures. These algorithms have the ability to find their way out of local optima. Surveys in metaheuristic algorithms have been published by [27,31,32,49,50,79].

A number of metaheuristic algorithms have been proposed for the solution of the Capacitated Vehicle Routing Problem. The most important algorithms published for each metaheuristic algorithm are given in the following:

- **Simulated Annealing** (**SA**) [1,3,47,72] plays a special role within local search for two reasons. First, they appear to be quite successful when applied to a broad range of practical problems. Second, some threshold accepting algorithms such as SA have a stochastic component, which facilitates a theoretical analysis of their asymptotic convergence. Simulated Annealing [2] is a stochastic algorithm that allows random uphill jumps in a controlled fashion in order to provide possible escapes from poor local optima. Gradually the probability allowing the objective function value to increase is lowered until no more transformations are possible. Simulated Annealing owes its name to an analogy with the annealing process in condensed matter physics, where a solid is heated to a maximum temperature at which all particles of the solid randomly arrange themselves in the liquid phase, followed by cooling through careful and slow reduction of the temperature until the liquid is frozen with the particles arranged in a highly structured lattice and minimal system energy. This ground state is reachable only if the maximum temperature is sufficiently high and the cooling sufficiently slow. Otherwise a meta-stable state is reached. The metastable state is also reached with a process known as quenching, in which the temperature is instantaneously lowered. Its predecessor is the so-called Metropolis filter. Simulated Annealing algorithms for the VRP are presented in [14,31,63].
- **Threshold Accepting Method** is a modification of the Simulated Annealing, which together with **record to record travel** [25,26] are known as **Deterministic Annealing** methods. These methods leave out the stochastic element in accepting worse solutions by introducing a deterministic threshold denoted by $Th_m > 0$, and accept a worse solution if $\sigma = c(S') - c(S) \leq Th_m$, where c is the cost of the solution. This is the move acceptance criterion and the subscript m is an iteration index. Dueck and Scheurer [26] were the first to propose the Threshold Accepting Method for the VRP. Tarantilis et al. [81,82] proposed two very efficient algorithms belonging to this class: the **Backtracking Adaptive Threshold Accepting** (**BATA**) and the **List-Based Threshold Accepting** (**LBTA**). Other Deterministic Annealing methods were proposed by Golden et al. [40], the **Record-to-Record Travel Method** and by Li et al. [51].
- **Tabu search (TS)** was introduced by Glover [34,35] as a general iterative metaheuristic for solving combinatorial optimization problems. Computational experience has shown that TS is a well established approximation technique, which can compete with almost all known techniques and which, by its flexibility, can beat many classic procedures. It is a form of local neighbor search. Each solution S has an associated set of neighbors $N(S)$. A solution $S' \in N(S)$ can be reached from S by an operation called a *move*. TS can be viewed as an iterative technique which explores a set of problem solutions, by repeatedly making moves from one solution S to another solution S' located in the neighborhood $N(S)$ of S [37]. TS moves from a solution to its best admissible neighbor, even if this causes the objective function to deteriorate. To avoid cycling, solutions that have been recently explored are declared *forbidden or tabu* for a number of iterations. The tabu status of a solution is overridden when certain criteria (*aspiration criteria*) are satisfied. Sometimes, *intensification* and *diversification* strategies are used to improve the search. In the first case, the search is accentuated in the promising regions of the feasible domain. In the second case, an attempt is made to consider solutions in a broad area of the search space. Tabu Search algorithms for the VRP are presented in [7,9,20,30,63,70,71,77,85,89,90].
- **Genetic Algorithms (GAs)** are search procedures based on the mechanics of natural selection and natural genetics. The first GA was developed by John H. Holland in the 1960s to allow computers to evolve solutions to difficult search and combinatorial prob-

lems, such as function optimization and machine learning [44]. Genetic algorithms offer a particularly attractive approach for problems like vehicle routing problem since they are generally quite effective for rapid global search of large, non-linear and poorly understood spaces. Moreover, genetic algorithms are very effective in solving large-scale problems. Genetic algorithms [38,72] mimic the evolution process in nature. GAs are based on an imitation of the biological process in which new and better populations among different species are developed during evolution. Thus, unlike most standard heuristics, GAs use information about a population of solutions, called individuals, when they search for better solutions. A GA is a stochastic iterative procedure that maintains the population size constant in each iteration, called a generation. Their basic operation is the mating of two solutions in order to form a new solution. To form a new population, a binary operator called crossover, and a unary operator, called mutation, are applied [65,66]. Crossover takes two individuals, called parents, and produces two new individuals, called offsprings, by swapping parts of the parents. Genetic Algorithms for the VRP are presented in [4,5,8,11,45,56,53,60,64].

- **Greedy Randomized Adaptive Search Procedure – GRASP** [73] is an iterative two phase search method which has gained considerable popularity in combinatorial optimization. Each iteration consists of two phases, a construction phase and a local search procedure. In the construction phase, a randomized greedy function is used to build up an initial solution. This randomized technique provides a feasible solution within each iteration. This solution is then exposed for improvement attempts in the local search phase. The final result is simply the best solution found over all iterations. Greedy Randomized Adaptive Search Procedure algorithms for the VRP are presented in [17,42,55].
- The use of **Artificial Neural Networks** to find good solutions to combinatorial optimization problems has recently caught some attention. A neural network consists of a network [76] of elementary nodes (neurons) that are linked through weighted connections. The nodes represent computational units, which are capable of performing a simple computation, consisting of a summation of the weighted inputs, followed by the addition of a constant called the threshold or bias, and the application of a nonlinear response (activation) function. The result of the computation of a unit constitutes its output. This output is used as an input for the nodes to which it is linked through an outgoing connection. The overall task of the network is to achieve a certain network configuration, for instance a required input–output relation, by means of the collective computation of the nodes. This process is often called *self-organization*. Neural Networks algorithm for the VRP are presented in [61,83].
- The **Ant Colony Optimization (ACO)** metaheuristic is a relatively new technique for solving combinatorial optimization problems (COPs). Based strongly on the Ant System (AS) metaheuristic developed by Dorigo, Maniezzo and Colorni [24], ant colony optimization is derived from the foraging behaviour of real ants in nature. The main idea of ACO is to model the problem as the search for a minimum cost path in a graph. Artificial ants walk through this graph, looking for good paths. Each ant has a rather simple behavior so that it will typically only find rather poor-quality paths on its own. Better paths are found as the emergent result of the global cooperation among ants in the colony. An ACO algorithm consists of a number of cycles (iterations) of solution construction. During each iteration a number of ants (which is a parameter) construct complete solutions using heuristic information and the collected experiences of previous groups of ants. These collected experiences are represented by a digital analogue of trail pheromone which is deposited on the constituent elements of a solution. Small quantities are deposited during the construction phase while larger amounts are deposited at the end of each iteration in proportion to solution quality. Pheromone can be deposited on the components and/or the connections used in a solution depending on the problem. Ant Colony Optimization algorithms for the VRP are presented in [10,15,16,23,57,67,68,69].
- **Path Relinking** This approach generates new solutions by exploring trajectories that connect high-quality solutions – by starting from one of these solutions, called the *starting solution* and generating a path in the neighborhood space that leads

towards the other solution, called the *target solution* [36]. Two new metaheuristic algorithms using the path relinking strategy as a part first of Tabu Search Metaheuristic is proposed in [43] and second as a part of a Particle Swarm Optimization Metaheuristic is proposed in [52].

- **Guided Local Search** (**GLS**), originally proposed by Voudouris and Chang [86,87], is a general optimization technique suitable for a wide range of combinatorial optimization problems. The main focus is on the exploitation of problem and search-related information to effectively guide local search heuristics in the vast search spaces of NP-hard optimization problems. This is achieved by augmenting the objective function of the problem to be minimized with a set of penalty terms which are dynamically manipulated during the search process to steer the heuristic to be guided. GLS augments the cost function of the problem to include a set of penalty terms and passes this, instead of the original one, for minimization by the local search procedure. Local search is confined by the penalty terms and focuses attention on promising regions of the search space. Iterative calls are made to local search. Each time local search gets caught in a local minimum, the penalties are modified and local search is called again to minimize the modification cost function. Guided Local Search algorithms for the VRP are presented in [58,59].
- **Particle Swarm Optimization (PSO)** is a population-based swarm intelligence algorithm. It was originally proposed by Kennedy and Eberhart as a simulation of the social behavior of social organisms such as bird flocking and fish schooling [46]. PSO uses the physical movements of the individuals in the swarm and has a flexible and well-balanced mechanism to enhance and adapt to the global and local exploration abilities. The first algorithm for the solution of the Vehicle Routing Problem was proposed by [52].
- One of the most interesting developments that have occurred in the area of TS in recent years is the concept of **Adaptive Memory** developed by Rochat and Taillard [74]. It is, mostly, used in TS, but its applicability is not limited to this type of metaheuristic. An adaptive memory is a pool of good solutions that is dynamically updated throughout the search process. Periodically, some elements of these solutions are extracted from the pool and combined differently to produce new good solutions. Very interesting and efficient algorithms based on the concept of Adaptive Memory have been proposed [74,78,79,80].
- **Variable Neighborhood Search (VNS)** is a metaheuristic for solving combinatorial optimization problems whose basic idea is systematic change of neighborhood within a local search [41]. Variable Neighborhood Search algorithms for the VRP are presented in [48].

References

1. Aarts E, Korst J (1989) Simulated Annealing and Boltzmann Machines – A stochastic Approach to Combinatorial Optimization and Neural Computing. Wiley, Chichester
2. Aarts E, Korst J, Van Laarhoven P (1997) Simulated Annealing. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, Chichester, pp 91–120
3. Aarts E, Ten Eikelder HMM (2002) Simulated Annealing. In: Pardalos PM, Resende MGC (eds) Handbook of Applied Optimization. Oxford University Press, Oxford, pp 209–221
4. Alba E, Dorronsoro B (2004) Solving the Vehicle Routing Problem by Using Cellular Genetic Algorithms, Conference on Evolutionary Computation in Combinatorial Optimization, EvoCOP'04. LNCS, vol 3004, 11–20, Portugal. Springer, Berlin
5. Alba E, Dorronsoro B (2006) Computing Nine New Best-So-Far Solutions for Capacitated VRP with a Cellular Genetic Algorithm. Inform Process Lett 98(6):225–230
6. Altinkemer K, Gavish B (1991) Parallel Savings Based Heuristics for the Delivery Problem. Oper Res 39(3): 456–469
7. Augerat P, Belenguer JM, Benavent E, Corberan A, Naddef D (1998) Separating Capacity Constraints in the CVRP Using Tabu Search. Eur J Oper Res 106(2–3):546–557
8. Baker BM, Ayechew MA (2003) A Genetic Algorithm for the Vehicle Routing Problem. Comput Oper Res 30(5): 787–800
9. Barbarosoglu G, Ozgur D (1999) A Tabu Search Agorithm for the Vehicle Routing Problem. Comput Oper Res 26:255–270
10. Bell JE, McMullen PR (2004) Ant Colony Optimization Techniques for the Vehicle Routing Problem. Adv Eng Inform 18(1):41–48
11. Berger J, Mohamed B (2003) A Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem. In: Proceedings of the Genetic and Evolutionary Computation Conference, Chicago, pp 646–656
12. Bodin L, Golden B (1981) Classification in Vehicle Routing and Scheduling. Networks 11:97–108

13. Bodin L, Golden B, Assad A, Ball M (1983) The State of the Art in the Routing and Scheduling of Vehicles and Crews. Comput Oper Res 10:63–212
14. Breedam AV (2001) Comparing Descent Heuristics and Metaheuristics for the Vehicle Routing Problem. Comput Oper Res 28(4):289–315
15. Bullnheimer B, Hartl RF, Strauss C (1997) Applying the Ant System to the Vehicle Routing Problem. Paper presented at 2nd International Conference on Metaheuristics, Sophia-Antipolis, France
16. Bullnheimer B, Hartl RF, Strauss C (1999) An Improved Ant System Algorithm for the Vehicle Routing Problem. Ann Oper Res 89:319–328
17. Chaovalitwongse W, Kim D, Pardalos PM (2003) GRASP with a new local search scheme for Vehicle Routing Problems with Time Windows. J Combin Optim 7(2):179–207
18. Christofides N, Mingozzi A, Toth P (1979) The Vehicle Routing Problem. In: Christofides N, Mingozzi A, Toth P, Sandi C (eds) Combinatorial Optimization. Wiley, Chichester
19. Clarke G, Wright J (1964) Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Oper Res 12:568–581
20. Cordeau JF, Gendreau M, Laporte G, Potvin JY, Semet F (2002) A Guide to Vehicle Routing Heuristics. J Oper Res Soc 53:512–522
21. Dantzig GB, Ramser RH (1959) The Truck Dispatching Problem. Manag Sci 6:80–91
22. Desrochers M, Verhoog TW (1989) A Matching Based Savings Algorithm for the Vehicle Routing Problem. Les Cahiers du GERAD G-89–04, Ecole des Hautes Etudes Commerciales de Montreal
23. Doerner K, Gronalt M, Hartl R, Reimman M, Strauss C, Stummer M (2002) Savings Ants for the Vehicle Routing Problem. In: Cagnoni S (ed) EvoWorkshops02. LNCS, vol 2279. Springer, Berlin, pp 11–20
24. Dorigo M, Stutzle T (2004) Ant Colony Optimization, A Bradford Book. MIT Press, London
25. Dueck G (1993) New Optimization Heuristics: The Great Deluge Algorithm and the Record-To-Record Travel. J Comput Phys 104:86–92
26. Dueck G, Scheurer T (1990) Threshold Accepting: A General Purpose Optimization Algorithm. J Comput Phys 90: 161–175
27. Fisher ML (1995) Vehicle routing. In: Ball MO, Magnanti TL, Momma CL, Nemhauser GL (eds) Network Routing. Handbooks in Operations Research and Management Science. North Holland, Amsterdam 8:1–33
28. Fisher ML, Jaikumar R (1981) A Generalized Assignment Heuristic for Vehicle Routing. Networks 11:109–124
29. Foster BA, Ryan DM (1976) An Integer Programming Approach to the Vehicle Scheduling Problem. Oper Res 27:367–384
30. Gendreau M, Hertz A, Laporte G (1994) A Tabu Search Heuristic for the Vehicle Routing Problem. Manag Sci 40:1276–1290
31. Gendreau M, Laporte G, Potvin J-Y (1997) Vehicle Routing: Modern Heuristics. In: Aarts EHL, Lenstra JK (eds) Local search in Combinatorial Optimization. Wiley, Chichester, pp 311–336
32. Gendreau M, Laporte G, Potvin JY (2002) Metaheuristics for the Capacitated VRP. In: Toth P, Vigo D (eds) The Vehicle Routing Problem, Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, pp 129–154
33. Gillett BE, Miller LR (1974) A Heuristic Algorithm for the Vehicle Dispatch Problem. Oper Res 22:240–349
34. Glover F (1989) Tabu Search I. ORSA J Comput 1(3): 190–206
35. Glover F (1990) Tabu Search II. ORSA J Comput 2(1):4–32
36. Glover F, Laguna M, Marti R (2003) Scatter Search and Path Relinking: Advances and Applications. In: Glover F, Kochenberger GA (eds) Handbook of Metaheuristics. Kluwer, Boston, pp 1–36
37. Glover F, Laguna M, Taillard E, de Werra D (eds) (1993) Tabu Search. Baltzer, Basel
38. Goldberg DE (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley, Reading
39. Golden BL, Assad AA (1988) Vehicle Routing: Methods and Studies. North Holland, Amsterdam
40. Golden BL, Wassil E, Kelly J, Chao IM (1998) The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithm, Problem Sets and Computational Results. In: Crainic TG, Laporte G (eds) Fleet Management and Logistics. Kluwer, Boston, pp 33–56
41. Hansen P, Mladenovic N (2001) Variable Neighborhood Search: Principles and Applications. Eur J Oper Res 130: 449–467
42. Hjorring C (1995) The Vehicle Routing Problem and Local Search Metaheuristics, PhD thesis, Department of Engineering Science, University of Auckland
43. Ho SC, Gendreau M (2006) Path Relinking for the Vehicle Routing Problem. J Heuristics 12:55–72
44. Holland JH (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor
45. Jaszkiewicz A, Kominek P (2003) Genetic Local Search with Distance Preserving Recombination Operator for a Vehicle Routing Problem. Eur J Oper Res 151:352–364
46. Kennedy J, Eberhart R (1995) Particle Swarm Optimization. In: Proceedings of (1995) IEEE International Conference on Neural Networks 4:1942–1948
47. Kirkpatrick S, Gelatt CD, Vecchi MP (1982) Optimization by Simulated Annealing. Science 220:671–680
48. Kytojoki J, Nuortio T, Braysy O, Gendreau M (2007) An Efficient Variable Neighborhood Search Heuristic for Very Large Scale Vehicle Routing Problems. Comput Oper Res 34(9):2743–2757
49. Laporte G, Gendreau M, Potvin J-Y, Semet F (2000) Classical and Modern Heuristics for the Vehicle Routing Problem. Int Trans Oper Res 7:285–300
50. Laporte G, Semet F (2002) Classical Heuristics for the Capacitated VRP. In: Toth P, Vigo D (eds) The Vehicle Routing

Problem, Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, pp 109–128
51. Li F, Golden B, Wasil E (2005) Very Large-Scale Vehicle Routing: New Test Problems, Algorithms and Results. Comput Oper Res 32(5):1165–1179
52. Marinakis Y, Marinaki M, Dounias G (2007) Nature Inspired Network Approaches in Management: Network Analysis and Optimization for the Vehicle Routing Problem Using NI-Techniques (submitted in AI Commun)
53. Marinakis Y, Marinaki M, Migdalas A (2006) A Hybrid Genetic – GRASP – ENS Algorithm for the Vehicle Routing Problem (submitted in Asia Pac J Oper Res)
54. Marinakis Y, Migdalas A (2002) Heuristic Solutions of Vehicle Routing Problems in Supply Chain Management. In: Pardalos PM, Migdalas A, Burkard R (eds) Combinatorial and Global Optimization. World Scientific, New Jersey, pp 205–236
55. Marinakis Y, Migdalas A, Pardalos PM (2006) Multiple Phase Neighborhood Search GRASP for the Vehicle Routing Problem (submitted in Comput Manag Sci)
56. Marinakis Y, Migdalas A, Pardalos PM (2007) A New Bilevel Formulation for the Vehicle Routing Problem and a Solution Method Using a Genetic Algorithm. J Global Optim 38:555–580
57. Mazzeo S, Loiseau I (2004) An Ant Colony Algorithm for the Capacitated Vehicle Routing. Electron Notes Discret Math 18:181–186
58. Mester D, Braysy O (2005) Active Guided Evolution Strategies for the Large Scale Vehicle Routing Problems with Time Windows. Comput Oper Res 32:1593–1614
59. Mester D, Braysy O (2007) Active-Guided Evolution Strategies for Large-Scale Capacitated Vehicle Routing Problems. Comput Oper Res 34(10):2964–2975
60. Mester D, Braysy O, Dullaert W (2007) A Multi-Parametric Evolution Strategies Algorithm for Vehicle Routing Problems. Expert Syst Appl 32(2):508–517
61. Modares A, Somhom S, Enkawa T (1999) A Self-Organizing Neural Network Approach for Multiple Traveling Salesman and Vehicle Routing Problems. Int Trans Oper Res 6(6):591–606
62. Mole RH, Jameson SR (1976) A Sequential Route-Building Algorithm Employing a Generalized Savings Criterion. Oper Res Q 27:503–511
63. Osman IH (1993) Metastrategy Simulated Annealing and Tabu Search Algorithms for Combinatorial Optimization Problems. Ann Oper Res 41:421–451
64. Prins C (2004) A Simple and Effective Evolutionary Algorithm for the Vehicle Routing Problem. Comput Oper Res 31:1985–2002
65. Reeves CR (1995) Genetic Algorithms. In: Reeves CR (ed) Modern Heuristic Techniques for Combinatorial Problems. McGraw-Hill, London, pp 151–196
66. Reeves CR (2003) Genetic Algorithms. In: Glover F, Kochenberger GA (eds) Handbooks of Metaheuristics. Kluwer, Dordrecht, pp 55–82
67. Reimann M, Doerner K, Hartl RF (2003) Analyzing a Unified Ant System for the VRP and Some of Its Variants. In: Cagnoni S et al. (eds) EvoWorkshops 2003. LNCS, vol 2611: 300–310. Springer, Berlin
68. Reimann M, Doerner K, Hartl RF (2004) D-Ants: Savings Based Ants Divide and Conquer the Vehicle Routing Problem. Comput Oper Res 31(4):563–591
69. Reimann M, Stummer M, Doerner K (2002) A Savings Based Ant System for the Vehicle Routing Problem. In: Proceedings of the Genetic and Evolutionary Computation Conference, New York, pp 1317–1326
70. Rego C (1998) A Subpath Ejection Method for the Vehicle Routing Problem. Manag Sci 44:1447–1459
71. Rego C (2001) Node-Ejection Chains for the Vehicle Routing Problem: Sequential and Parallel Algorithms. Parallel Comput 27(3):201–222
72. Rego C, Glover F (2002) Local Search and Metaheuristics. In: Gutin G, Punnen A (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 309–367
73. Resende MGC, Ribeiro CC (2003) Greedy Randomized Adaptive Search Procedures. In: Glover F, Kochenberger GA (eds) Handbook of Metaheuristics. Kluwer, Boston, pp 219–249
74. Rochat Y, Taillard ED (1995) Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. J Heuristics 1:147–167
75. Shi Y, Eberhart R (1998) A Modified Particle Swarm Optimizer. In: Proceedings of (1998) IEEE World Congress on Computational Intelligence, pp 69–73
76. Sodererg B, Peterson C (1997) Artificial Neural Networks. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, Chichester, pp 173–214
77. Taillard ED (1993) Parallel Iterative Search Methods for Vehicle Routing Problems. Networks 23:661–672
78. Taillard ED, Gambardella LM, Gendreau M, Potvin JY (2001) Adaptive Memory Programming: A Unified View of Metaheuristics. Eur J Oper Res 135(1):1–16
79. Tarantilis CD (2005) Solving the Vehicle Routing Problem with Adaptive Memory Programming Methodology. Comput Oper Res 32(9):2309–2327
80. Tarantilis CD, Kiranoudis CT (2002) BoneRoute: An Adaptive Memory-Based Method for Effective Fleet Management. Ann Oper Res 115(1):227–241
81. Tarantilis CD, Kiranoudis CT, Vassiliadis VS (2002) A Backtracking Adaptive Threshold Accepting Metaheuristic Method for the Vehicle Routing Problem. Syst Anal Modeling Simul (SAMS) 42(5):631–644
82. Tarantilis CD, Kiranoudis CT, Vassiliadis VS (2002) A List Based Threshold Accepting Algorithm for the Capacitated Vehicle Routing Problem. Int J Comput Math 79(5): 537–553
83. Torki A, Somhon S, Enkawa T (1997) A Competitive Neural Network Algorithm for Solving Vehicle Routing Problem. Comput Indust Eng 33(3–4):473–476

84. Toth P, Vigo D (2002) The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia
85. Toth P, Vigo D (2003) The Granular Tabu Search (and its Application to the Vehicle Routing Problem). INFORMS J Comput 15(4):333–348
86. Voudouris C, Tsang E (1999) Guided Local Search and its Application to the Travelling Salesman Problem. Eur J Oper Res 113:469–499
87. Voudouris C, Tsang E (2003) Guided Local Search. In: Glover F, Kochenberger GA (eds) Handbooks of Metaheuristics. Kluwer, Dordrecht, pp 185–218
88. Wark P, Holt J (1994) A Repeated Matching Heuristic for the Vehicle Routing Problem. J Oper Res Soc 45:1156–1167
89. Willard JAG (1989) Vehicle Routing Using r-Optimal Tabu Search. Master thesis, The Management School, Imperial College, London
90. Xu J, Kelly JP (1996) A New Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem. Transp Sci 30:379–393

Metaheuristics

Stefan Voss
Institute of Information Systems (Wirtschaftsinformatik), University of Hamburg, Hamburg, Germany

MSC2000: 68T20, 90C59, 90C27, 68T99

Article Outline

Keywords and Phrases

Heuristics; Metaheuristics; Greedy randomized adaptive search procedure; Pilot method; Variable neighborhood search; Simulated annealing; Tabu search; Genetic algorithm; Evolutionary algorithm; Scatter search; Hybridization; Optimization software library; POPMUSIC; Adaptive memory programming; Pool template

Introduction

Many decision problems in various areas such as business, engineering, economics, and science, including those in manufacturing, location, routing, and scheduling, may be formulated as optimization problems. Owing to the complexity of many of these optimization problems, particularly those of large sizes encountered in most practical settings, exact algorithms often perform very poorly, in some cases taking days or more to find moderately decent, let alone optimal, solutions even to fairly small instances. As a result, heuristic algorithms are conspicuously preferable in practical applications.

As an extension of simple heuristics, a large number of local search approaches have been developed to improve given feasible solutions. The main drawback of these approaches is their inability to continue the search upon becoming trapped in local optima. This leads to consideration of techniques for guiding known heuristics to overcome local optimality. Following this theme metaheuristics have become a most important class of approaches for solving optimization problems. They support managers in decision-making with robust tools that provide high-quality solutions to important applications in reasonable time horizons.

We describe metaheuristics mainly from an operations research perspective. Earlier survey papers on metaheuristics include those of Blum and Roli [14] and Voß [95]. Here we occasionally rely on the latter. The general concepts have not become obsolete, and many changes are mainly based upon an update to most recent references. A handbook on metaheuristics is available describing a great variety of concepts by various authors in a comprehensive manner [44].

Definitions

The basic concept of heuristic search as an aid to problem solving was first introduced in [76]. A *heuristic* is a technique (consisting of a rule or a set of rules) which seeks (and hopefully finds) *good* solutions at a reasonable computational cost. A heuristic is *approximate* in the sense that it provides (hopefully) a good solution for relatively little effort, but it does not guarantee optimality.

Heuristics provide simple means of indicating which among several alternatives seems to be best. That is, "Heuristics are criteria, methods, or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal. They represent compromises between two requirements: the need to make such criteria simple and, at the same time, the desire to see them discriminate correctly between good and bad choices. A heuristic may be a *rule of thumb* that is used to guide one's action" [73].

Greedy heuristics are simple iterative approaches available for any kind of (e. g., combinatorial) optimization problem. A good characterization is their *myopic* behavior. A greedy heuristic starts with a given feasible or infeasible solution. In each iteration there are a number of alternative choices (*moves*) that can be made to transform the solution. From these alternatives which consist in fixing (or changing) one or more variables, a *greedy choice* is made, i. e., the best alternative according to a given measure is chosen until no such transformations are possible any longer.

Usually, a greedy *construction heuristic* starts with an incomplete solution and completes it stepwise. Savings and dual algorithms follow the same iterative scheme: dual heuristics change an infeasible low-cost solution until reaching feasibility; savings algorithms start with a high-cost solution and realize the highest savings as long as possible. Moreover, in all three cases, once an element has been chosen this decision is (usually) not reversed throughout the algorithm, it is kept.

As each alternative has to be measured, in general we may define some sort of *heuristic measure* (providing, e. g., some priority values or some ranking information) which is iteratively followed until a complete solution is built. Usually this heuristic measure is applied in a greedy fashion.

For heuristics we usually have the distinction between finding initial feasible solutions and improving them. In that sense we first discuss local search before characterizing metaheuristics.

Local Search

The basic principle of local search is to successively alter solutions *locally*. Related transformations are defined by neighborhoods which for a given solution include all solutions that can be reached by one move. That is, neighborhood search usually is assumed to correspond to the process of iteratively moving from one solution to another one by performing some sort of operation. More formally, each solution of a problem has an associated set of neighbors called its *neighborhood*, i. e., solutions that can be obtained by a single operation called transformation or move. Most common ideas for transformations are, e. g., to add or drop some problem-specific individual components. Other options are to exchange two components simultaneously, or to swap them. Furthermore, components may be shifted from a certain position into other positions. All components involved within a specific move are called its elements or attributes.

Moves must be evaluated by some *heuristic measure* to guide the search. Often one uses the implied change of the objective function value, which may provide reasonable information about the (local) advantage of moves. Following a greedy strategy, *steepest descent* (SD) corresponds to selecting and performing in each iteration the best move until the search stops at a local optimum. Obviously, savings algorithms correspond to SD.

As the solution quality of local optima may be unsatisfactory, we need mechanisms which guide the search to overcome local optimality. A simple strategy called iterated local search is to iterate/restart the local search process after a local optimum has been obtained, which requires some perturbation scheme to generate a new initial solution (e. g., performing some random moves). Of course, more structured ways to overcome local optimality may be advantageous.

A general survey of local search can be found in [1] and the references from [2]. A simple template is provided in [90].

Starting in the 1970s (see Lin and Kernighan [66]), a variable way of handling neighborhoods is still a topic within local search. Consider an arbitrary neighborhood structure N, which defines for any solution s a set of neighbor solutions $N_1(s)$ as a neighborhood of depth $d = 1$. In a straightforward way, a neighborhood $N_{d+1}(s)$ of depth $d + 1$ is defined as the set $N_d(s) \cup \{s' | \exists s'' \in N_d(s) : s' \in N_1(s'')\}$. In general, a large d might be unreasonable, as the neighborhood size may grow exponentially. However, depths of two or three may be appropriate. Furthermore, temporarily increasing the neighborhood depth has been found to be a reasonable mechanism to overcome *basins of attraction*, e. g., when a large number of neighbors with equal quality exist.

Large-scale neighborhoods have become an important topic (see, e. g., [5] for a survey), especially when efficient ways are at hand for exploring them. Related research can also be found under various names; see, e. g., [75] for the idea of ejection chains.

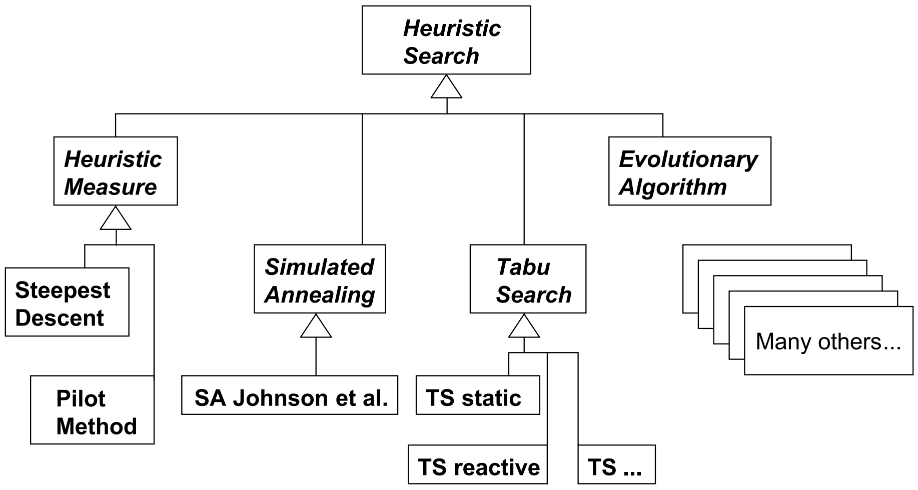
Stochastic local search is pretty much all we know about local search but is enhanced by randomizing choices. That is, a stochastic local search algorithm is a local search algorithm making use of randomized choices in generating or selecting candidate solutions for given instances of optimization problems. Randomness may be used for search initialization as well as the computation of search steps. A comprehensive treatment of stochastic local search is given in [58].

Metaheuristics

The formal definition of metaheuristics is based on a variety of definitions from different authors based on [39]. Basically, a metaheuristic is a top-level strategy that guides an underlying heuristic solving a given problem. In that sense we distinguish between a *guiding process* and an *application process*. The guiding process decides upon possible (local) moves and forwards its decision to the application process, which then executes the move chosen. In addition, it provides information for the guiding process (depending on the requirements of the respective metaheuristic) like the recomputed set of possible moves.

According to [43], "metaheuristics in their modern forms are based on a variety of interpretations of what constitutes *intelligent search*", where the term "intelligent search" has been made prominent by Pearl [73] (regarding heuristics in an artificial intelligence context; see also [92] regarding an operations research context). In that sense we may also consider the following definition: "A metaheuristic is an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search spaces using learning strategies to structure information in order to find efficiently near-optimal solutions" [72].

To summarize, the following definition seems to be most appropriate: "A metaheuristic is an iterative



Metaheuristics, Figure 1
Simplified metaheuristics inheritance tree

master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method. The family of metaheuristics includes, but is not limited to, adaptive memory procedures, tabu search, ant systems, greedy randomized adaptive search, variable neighborhood search, evolutionary methods, genetic algorithms, scatter search, neural networks, simulated annealing, and their hybrids" (p. ix in [97]).

We describe the ingredients and basic concepts of various metaheuristic strategies like *tabu search* (TS), *simulated annealing* (SA), and *scatter search*. This is based on a simplified view of a possible inheritance tree for heuristic search methods, illustrating the relationships between some of the most important methods discussed below, as shown in Fig. 1.

We also emphasize advances including the important incorporation of exact methods into intelligent search. Furthermore, general frames are sketched that may subsume various approaches within the metaheuristics field.

Metaheuristic Methods

We survey the basic concepts of some of the most important metaheuristics. We shall see that adaptive processes originating from different settings such as psychology ("learning"), biology ("evolution"), physics ("annealing"), and neurology ("nerve impulses") have served as interesting starting points.

Simple Local Search Based Metaheuristics

To improve the efficiency of greedy heuristics, one may apply generic strategies to be used alone or in combination with each other, namely, changing the definition of alternative choices, look ahead evaluation, candidate lists, and randomized selection criteria bound up with *repetition*, as well as combinations with local search or other methods.

Greedy Randomized Adaptive Search Omitting a greedy choice criterion for a random strategy, one can run the algorithm several times and obtain a large number of different solutions. A combination of best and random choice seems to be appropriate: We define a *candidate list* as a list consisting of a number of (best, i. e., first best, second best, third best, etc.) alternatives. Out of this list one alternative is chosen randomly. The length of the candidate list is given either as an absolute value, a percentage of all feasible alternatives, or implicitly by defining an allowed quality gap (to the best alternative), which also may be an absolute value or a percentage.

Replicating a search procedure to determine a local optimum multiple times with different starting points has been given the acronym GRASP and investigated with respect to different applications. A comprehensive survey of GRASP and its applications is given in [32]. It should be noted that GRASP goes back to older approaches [52], which is frequently overlooked in many applications. The different initial solutions or starting points are found by a greedy procedure incorporating a probabilistic component. That is, given a candidate list to choose from, GRASP randomly chooses one of the best candidates from this list in a greedy fashion, but not necessarily the best possible choice.

The underlying principle is to investigate many good starting points through the greedy procedure and thereby to increase the possibility of finding a good local optimum on at least one replication. The method is said to be adaptive as the greedy function takes into account previous decisions when performing the next choice.

The Pilot Method Building on a simple greedy algorithm such as, e. g., a construction heuristic, the *pilot method* [29,30] is a metaheuristic not necessarily based on a local search in combination with an improvement procedure. It primarily *looks ahead* for each possible local choice (by computing a so-called "pilot" solution), memorizing the best result, and performing the respective move. (Very similar ideas have been investigated under the name *rollout method* [13].) One may apply this strategy by successively performing a greedy heuristic for all possible local steps (i. e., starting with all incomplete solutions resulting from adding some not yet included element at some position to the current incomplete solution). The look ahead mechanism of the pilot method is related to increased neighborhood depths as the pilot method exploits the evaluation of neighbors at larger depths to guide the neighbor selection at depth one.

In most applications, it is reasonable to restrict the pilot process to some *evaluation depth*. That is, the method is performed up to an incomplete solution (e. g., partial assignment) based on this evaluation depth and is then completed by continuing with a conventional heuristic. For a recent study applying the pilot method to several combinatorial optimization problems obtaining very good results see [96]. Additional applications can be found, e. g., in [18,68].

Variable Neighborhood Search The basic idea of variable neighborhood search (VNS) is to change the neighborhood during the search in a systematic way. VNS usually explores increasingly distant neighborhoods of a given solution, and jumps from this solution to a new one if and only if an improvement has been made. In this way often favorable characteristics of incumbent solutions, e. g., that many variables are already at their appropriate value, will be kept and used to obtain promising neighboring solutions.

Moreover, a local search routine is applied repeatedly to get from these neighboring solutions to local optima. This routine may also use several neighborhoods. Therefore, to construct different neighborhood structures and to perform a systematic search, one needs to have a way for finding the distance between any two solutions, i. e., one needs to supply the solution space with some metric (or quasi-metric) and then induce neighborhoods from it. For an excellent treatment of various aspects of VNS see [51].

Simulated Annealing

Simulated annealing (SA) extends basic local search by allowing moves to inferior solutions [26,64]. A basic SA algorithm may be described as follows: Successively, a candidate move is randomly selected; this move is accepted if it leads to a solution with an improved objective function value compared to the current solution, otherwise, the move is accepted with a probability depending on the deterioration Δ of the objective function value. The acceptance probability is computed as $e^{-\Delta/T}$, using a temperature T as a control parameter. Usually, T is reduced over time for diversification at an earlier stage of the search and to intensify later.

Various authors have described a robust concretization of this general SA approach [60,62]. An interesting variant of SA is to strategically reheat the process, i. e., to perform a nonmonotonic acceptance function.

Threshold accepting [28] is a modification (or simplification) of SA accepting every move that leads to a new solution which is “not much worse” (i. e., deteriorates not more than a certain threshold which reduces with temperature) than the older one.

Tabu Search

The basic paradigm of *tabu search* (TS) is to use information about the search history to guide local search approaches to overcome local optimality (see [43] for a survey on TS). In general, this is done by a dynamic transformation of the local neighborhood. Based on some sort of memory, certain moves may be forbidden; we say they are set tabu. As for SA, the search may lead to performing deteriorating moves when no improving moves exist or when all improving moves of the current neighborhood are set tabu. At each iteration a best admissible neighbor may be selected. A neighbor, or a corresponding move, is called admissible if it is not tabu or if an aspiration criterion is fulfilled. An aspiration criterion is a rule to eventually override a possibly unreasonable tabu status of a move. For example, a move that leads to a neighbor with a better objective function value than encountered so far should be considered as admissible.

We briefly describe some TS methods that differ especially in the way in which tabu criteria are defined, taking into consideration the information about the search history (performed moves, traversed solutions).

The most commonly used TS method is based on a *recency-based* memory that stores moves, or attributes characterizing respective moves, of the recent past (*static TS*). The basic idea of such approaches is to prohibit an appropriately defined inversion of performed moves for a given period. For example, one may store the solution attributes that have been created by a performed move in a tabu list. To obtain the current tabu status of a move to a neighbor, one may check whether (or how many of) the solution attributes that would be destroyed by this move are contained in the tabu list.

Strict TS embodies the idea of preventing cycling to formerly traversed solutions. The goal is to provide necessity and sufficiency with respect to the idea of not revisiting any solution. Accordingly, a move is classified as tabu if and only if it leads to a neighbor that has already been visited during the previous search. There are two primary mechanisms to accomplish the tabu criterion: First, we may exploit logical interdependencies between the sequence of moves performed throughout the search process, as realized by, e. g., the reverse elimination method and the cancellation sequence method [40,94]. Second, we may store information about all solutions visited so far. This may be carried out either exactly or, for reasons of efficiency, approximately (e. g., by using hash codes).

Reactive TS aims at the automatic adaptation of the tabu list length of static TS [12]. The idea is to increase the tabu list length when the tabu memory indicates that the search is revisiting formerly traversed solutions. A possible specification can be described as follows: Starting with a tabu list length l of 1 it is increased every time a solution has been repeated. If there has been no repetition for some iterations, we decrease it appropriately. To accomplish the detection of a repetition of a solution, one may apply a trajectory-based memory using hash codes as for strict TS.

For reactive TS [12], it is appropriate to include means for diversifying moves whenever the tabu memory indicates that we are trapped in a certain region of the search space. As a trigger mechanism one may use, e. g., the combination of at least two solutions each having been traversed three times. A very simple escape strategy is to perform a number of random moves (depending on the average of the number of iterations between solution repetitions); more advanced strategies may take into account some long-term memory information (like the frequencies of appearance of specific solution attributes in the search history).

Of course there are a great variety of additional ingredients that may make TS work successfully, e. g., restricting the number of neighbor solutions to be evaluated (using candidate list strategies).

Evolutionary Algorithms

Evolutionary algorithms comprise a great variety of different concepts and paradigms, including genetic algorithms (GAs) [45,56], evolutionary strategies [55,83], evolutionary programs [36], scatter search [38,41], and memetic algorithms [71]. For surveys and references on evolutionary algorithms see also [9,37,69,78].

GAs are a class of adaptive search procedures based on principles derived from the dynamics of natural population genetics. One of the most crucial ideas for a successful implementation of a GA is the representation of an underlying problem by a suitable scheme. A GA starts, e. g., with a randomly created initial population of artificial creatures (strings), a set of solutions. These strings in whole and in part are the base set for all subsequent populations. They are copied and information is exchanged between the strings in order to find new solutions of the underlying problem. The mechanisms of a simple GA essentially consist of copying strings and exchanging partial strings. A simple GA uses three operators which are named according to the corresponding biological mechanisms: reproduction, crossover, and mutation. Performing an operator may depend on a *fitness function* or its value (fitness), respectively. As some sort of heuristic measure, this function defines a means of measurement for the profit or the quality of the coded solution for the underlying problem and often depends on the objective function of the given problem.

GAs are closely related to *evolutionary strategies*. Whereas the mutation operator in a GA serves to protect the search from premature loss of information, evolution strategies may incorporate some sort of local search procedure (such as SD) with self-adapting parameters involved in the procedure. On a simplified scale many algorithms may be coined evolutionary once they are reduced to the following frame [54]:

1. Generate an initial population of individuals.
2. While no stopping condition is met do.
3. Co-operation.
4. Self-adaptation.

Self-adaptation refers to the fact that individuals (solutions) evolve independently while co-operation refers to an information exchange among individuals.

Scatter search ideas established a link between early ideas from various sides – evolutionary strategies, TS, and GAs. As an evolutionary approach, scatter search originated from strategies for creating composite decision rules and surrogate constraints [38]. Scatter search is designed to operate on a set of points, called reference points, that constitute good solutions obtained from previous solution efforts. The approach systematically generates linear combinations of the reference points to create new points, each of which is mapped into an associated point that yields integer values for discrete variables. Scatter search contrasts with other evolutionary procedures, such as GAs, by providing unifying principles for joining solutions based on generalized path constructions in Euclidean space and by utilizing strategic designs where other approaches resort to randomization. For a very comprehensive treatment of scatter search see [65].

Swarm Intelligence

Swarm intelligence is a relatively novel discipline interested in the study of self-organizing processes in nature and human artifacts [15,63]. While researchers in ethnology and animal behavior have proposed many models to explain various aspects of social insect behavior such as self-organization and shape formation, algorithms inspired by these models have been proposed to solve optimization problems. Successful examples are the so-called *ant system* or *ant colony* paradigm, the bee system, and swarm robotics, where the focus is on applying swarm intelligence techniques to the control of large groups of cooperating autonomous robots.

The *ant system* is a dynamic optimization process reflecting the natural interaction between ants searching for food [23]. The ants' ways are influenced by two different kinds of search criteria. The first one is the local visibility of food, i. e., the attractiveness of food in each ant's neighborhood. Additionally, each ant's way through its food space is affected by the other ants' trails as indicators for possibly good directions. The intensity of trails itself is time-dependent: With time passing, parts of the trails are diminishing, while the intensity may increase by new and fresh trails. With the quantities of these trails changing dynamically, an autocatalytic optimization process is started forcing the ants' search into most promising regions. This process of interactive learning can easily be modeled for most kinds of optimization problems by using simultaneously and interactively processed search trajectories.

A comprehensive treatment of the ant system paradigm can be found in [24]. To achieve enhanced performance of the ant system it is useful to hybridize it at least with a local search component.

Miscellaneous

Target analysis may be viewed as a general learning approach. Given a problem we first explore a set of sample instances and an extensive effort is made to obtain a solution which is optimal or close to optimality. The best solutions obtained provide *targets* to be sought within the next part of the approach. For instance, a TS algorithm may resolve the problems with the aim of finding what are the right choices to come to the already known solution (or as close to it as possible). This may give some information on how to choose parameters for other problem instances.

A different method in this context is *path relinking* (PR), which provides a useful means of intensification and diversification. Here new solutions are generated by exploring search trajectories that combine elite solutions, i. e., solutions that have proven to be better than others throughout the search. For references on target analysis and PR see [43].

Recalling local search based on *data perturbation* the *noising method* may be related to the following approach too. Given an initial feasible solution, the method performs some data perturbation [87] in order to change the values taken by the objective function of a respective problem to be solved. On the perturbed data a local search may be performed (e. g., following a SD approach). The amount of data perturbation (the noise added) is successively reduced until it reaches zero. The noising method is applied, e. g., in [19] for the clique partitioning problem.

The key issue in designing *parallel algorithms* is to decompose the execution of the various ingredients of a procedure into processes executable by parallel processors. In contrast to ant systems or GAs, metaheuristics like TS or SA, at first glance, have an intrinsic sequential nature owing to the idea of performing the neighborhood search from one solution to the next. However, some effort has been undertaken to define templates for parallel local search [20,90,91,93]. A comprehensive treatment with successful applications is provided in [6]. The discussion of parallel metaheuristics has also led to interesting hybrids such as the combination of a population of individual processes, agents, in a cooperative and competitive nature (see, e. g., the discussion of *memetic algorithms* in [71]) with TS.

Neural networks may be considered as metaheuristics, although we have not considered them here; see [85] for a comprehensive survey of these techniques for combinatorial optimization. In contrast, one may use metaheuristics to speed up the learning process regarding artificial neural networks; see [7] for a comprehensive consideration.

Furthermore, recent efforts on problems with multiple objectives and corresponding metaheuristic approaches can be found in [61]. See [82] for some ideas regarding GAs and fuzzy multiobjective optimization.

General Frames

An important avenue of metaheuristics research refers to general frames (to explain the behavior and the relationship between various methods) as well as the development of software systems incorporating metaheuristics (eventually in combination with other methods). Besides other aspects, this takes into consideration that in metaheuristics it has very often been appropriate to incorporate a certain means of diversification vs. intensification to lead the search into new regions of the search space. This requires a meaningful mechanism to detect situations when the search might be trapped in a certain area of the solution space. Therefore, within intelligent search the exploration of memory plays a most important role.

Adaptive Memory Programming

Adaptive memory programming (AMP) coins a general approach (or even thinking) within heuristic search focusing on exploiting a collection of memory components [42,89]. An AMP process iteratively constructs (new) solutions based on the exploitation of some memory, especially when combined with learning mechanisms supporting the collection and use of the memory. Based on the idea of initializing the memory and then iteratively generating new solutions (utilizing the given memory) while updating the memory based on the search, we may subsume various of the above-described metaheuristics as AMP approaches. This also includes exploiting provisional solutions that are improved by a local search approach.

The performance as well as the efficiency of a heuristic scheme strongly depends on its ability to use AMP techniques providing flexible and variable strategies for types of problems (or special instances of a given problem type) where standard methods fail. Such AMP techniques could be, e. g., dynamic handling of operational restrictions, dynamic move selection formulas, and flexible function evaluations.

Consider, as an example, adaptive memory within TS concepts. Realizing AMP principles depends on which specific TS application is used. For example, the reverse elimination method observes logical interdependencies between moves and infers corresponding tabu restrictions, and therefore makes fuller use of AMP than simple static approaches do.

To discuss the use of AMP in intelligent agent systems, one may use the simple model of ant systems as an illustrative starting point. Ant systems are based on combining local search criteria with information derived from the trails. This follows the AMP requirement for using flexible (dynamic) move selection rules (formulas). However, the basic ant system exhibits some structural inefficiencies when viewed from the perspective of general intelligent agent systems, as no distinction is made between successful and less successful agents, no time-dependent distinction is made, and there is no explicit handling of restrictions providing protection against cycling and duplication. Furthermore, there are possible conflicts between the information held in the adaptive memory (*diverging trails*).

A Pool Template

In [48] a *pool template* (PT) is proposed as can be seen in Fig. 2. The following notation is used. A pool of $p \geq 1$ solutions is denoted by P. Its input and output transfer is managed by two functions which are called *IF* and *OF*, respectively. S is a set of solutions with cardinality $s \geq 1$. A solution combination method (procedure *SCM*) constructs a solution from a given set S, and *IM* is an improvement method.

Depending on the method used, in step 1 a pool is either completely (or partially) built by a (randomized) diversification generator or filled with a single solution which has been provided, e. g., by a simple greedy approach. Note that a crucial parameter that deserves careful elaboration is the cardinality p of the pool. The main loop, executed until a termination criterion holds, consists of steps 2–5. Step 2 is the call of the output

```
1. Initialize $P$ by an external procedure
WHILE termination=FALSE DO BEGIN
    2. $S := OF(P)$
    3. IF $s > 1$ THEN $S' := SCM(S)$
       ELSE $S' := S$
    4. $S'' := IM(S')$
    5. $P := IF(S'')$
END
6. Apply a post-optimizing procedure to $P$
```

Metaheuristics, Figure 2
Pool template

function which selects a set of solutions, S, from the pool. Depending on the kind of method represented in the PT, these solutions may be assembled (step 3) to a working solution S' which is the starting point for the improvement phase of step 4. The outcome of the improvement phase, S'', is then evaluated by means of the input function, which possibly feeds the new solution into the pool. Note that a post-optimization procedure in step 6 is for facultative use. It may be a straightforward greedy improvement procedure if used for single-solution heuristics or a pool method on its own. As an example we quote a *sequential* pool method, the TS with PR in [11]. Here a PR phase is added *after* the pool has been initialized by a TS. A *parallel* pool method on the other hand uses a pool of solutions *while* it is constructed by the guiding process (e.g., a GA or scatter search).

Several heuristic and metaheuristic paradigms, whether they are obviously pool-oriented or not, can be summarized under the common PT frame. We provide the following examples:

a) Local search/SD: PT with $p = s = 1$.
b) SA: $p = 2, s = 1$ incorporating its probabilistic acceptance criterion in IM. (It should be noted that $p = 2$ and $s = 1$ seems to be unusual at first glance. For SA we always have a current solution in the pool for which one or more neighbors are evaluated and eventually a neighbor is found which replaces the current solution. Furthermore, at all iterations throughout the search the so far best solution is stored too (even if no real interaction between those two stored solutions takes place). The same is also valid for a simple TS. As for local search the current solution corresponds to the best solution of the specific search, we have $p = 1$.)
c) Standard TS: $p = 2, s = 1$ incorporating adaptive memory in IM.
d) GAs: $p > 1$ and $s > 1$ with population mechanism (crossover, reproduction, and mutation) in SCM of step 3 and without the use of step 4.
e) Scatter search: $p > 1$ and $s > 1$ with subset generation in OF of step 2, linear combination of elite solutions by means of SCM in step 3, e.g., a TS for procedure IM and a reference set update method in IF of step 5.
f) PR (as a parallel pool method): $p > 1$ and $s = 2$ with a PR neighborhood in SCM. Facultative use of step 4.

Partial Optimization Metaheuristic Under Special Intensification Conditions

A natural way to solve large optimization problems is to decompose them into independent subproblems that are solved with an appropriate procedure. However, such approaches may lead to solutions of moderate quality since the subproblems might have been created in a somewhat arbitrary fashion. Of course, it is not easy to find an appropriate way to decompose a problem a priori. The basic idea of POPMUSIC conditions is to locally optimize subparts of a solution, a posteriori, once a solution to the problem is available. These local optimizations are repeated until a local optimum is found. Therefore, POPMUSIC may be viewed as a local search working with a special, large neighborhood. While the acronym POPMUSIC was given by Taillard and Voß [88] other metaheuristics may be incorporated into the same framework too [84].

For large optimization problems, it is often possible to see the solutions as composed of parts (or chunks [102], cf. the term "vocabulary building"). Considering the vehicle routing problem, a part may be a tour (or even a customer). Suppose that a solution can be represented as a set of parts. Moreover, some parts are more in relation with some other parts, so a corresponding heuristic measure can be defined between two parts. The central idea of POPMUSIC is to select a so-called *seed part* and a set P of parts that are mostly related to the seed part to form a subproblem.

Then it is possible to state a local search optimization frame that consists of trying to improve all subproblems that can be defined, until the solution does not contain a subproblem that can be improved. In the POPMUSIC frame of [88], *P* corresponds precisely to seed parts that have been used to define subproblems that have been unsuccessfully optimized. Once *P* contains all the parts of the complete solution, all subproblems have been examined without success and the process stops.

Basically, the technique is a gradient method that starts from a given initial solution and stops in a local optimum relative to a large neighborhood structure. To summarize, both POPMUSIC as well as AMP may serve as a general frame encompassing various other approaches.

Hybrids with Exact Methods

Often a new idea or a new paradigm in metaheuristics is claimed to be *the* idea by the inventor, while others see it as useless in the first instance. However, once it has been hybridized, things begin to *fly*. Especially in population-based metaheuristics, many researchers have followed this trend. That is, we now see many hybrid approaches where the successful ingredients of various metaheuristics have been combined. The term "hybridization", however, goes further, as it also refers to combining metaheuristics with exact methods.

Traditionally, the structure of neighborhoods is determined by local transformations or moves. This usually refers to relatively small homogeneous neighborhoods. Different types of moves have been used in the construction of very large and diverse neighborhoods. In contrast, as a hybrid one may deploy neighborhoods that are *method-based*. By this we mean that the basic structure of a neighborhood is determined by the needs and requirements of a given (say, exact) optimization method used to search the neighborhood. That is, given an incumbent solution one may define the neighborhood so that an exact method can be efficiently used rather than defining a neighborhood and trying to find an appropriate method to explore it. This approach was called *corridor method* by Sniedovich and Voß [86] as it literally defines a neighborhood as a sufficiently sized corridor around a given solution so that a given exact method behaves well. Iteratively the corridor is moved through the search space for exploration.

Constraint programming (CP) is a paradigm for representing and solving a wide variety of problems expressed by means of variables, their domains, and constraints on the variables. Usually CP models are solved using depth-first search and branch and bound. Naturally, these concepts can be complemented by local search concepts and metaheuristics. This idea is followed by several authors; see [21] for TS and guided local search hybrids. Commonalities with the POPMUSIC approach can be deduced from [74].

Of course, the treatment of this topic is by no means complete and various ideas have been developed. One idea is to transform a greedy heuristic into a search algorithm by branching only in a few (i. e., limited number) cases when the choice criterion of the heuristic observes some borderline case or where the choice is least compelling, respectively. This approach may be called *limited discrepancy search* [17,53].

Independent from the CP concept, one may investigate hybrids of branch and bound and metaheuristics, e. g., for deciding upon branching variables or search paths to be followed within a branch and bound tree (see [103] for an application of reactive TS). Here we may also use the term "cooperative solver." Somewhat related is the local branching concept for solving mixed integer programs (MIP), which seeks to explore neighborhoods defined through (invalid) linear cuts. The neighborhoods are searched by means of a general purpose MIP solver [35].

Correspondingly, one of the current research issues refers to exploiting mathematical programming (MP) techniques in a (meta)heuristic framework or, correspondingly, granting to MP approaches the cross-problem robustness and computation time effectiveness which characterize metaheuristics. Discriminating landmark is some form of exploitation of a MP formulation, e. g., by means of MIP. In this respect various efforts have been made towards developing strategies for making a heuristic sequence of roundings to obtain feasible solutions for problems represented by means of appropriate MIP [3,34].

Optimization Software Libraries

Besides some well-known approaches for reusable software in the field of exact optimization (e. g., CPLEX or ABACUS; see http://www.ilog.com and http://www.

informatik.uni-koeln.de/abacus) some ready-to-use and well-documented component libraries in the field of local search based heuristics and metaheuristics have been developed; see especially the contributions in [98].

The most successful approaches documented in the literature are the heuristic optimization framework HOTFRAME of [33] and EASYLOCAL++ of [22]. HOTFRAME, as an example, is implemented in C++, which provides adaptable components incorporating different metaheuristics and an architectural description of the collaboration among these components and problem-specific complements. Typical application-specific concepts are treated as objects or classes: problems, solutions, neighbors, solution attributes, and move attributes. On the other hand, metaheuristic concepts such as the different methods described above and their building blocks such as tabu criteria or diversification strategies are also treated as objects. HOTFRAME uses genericity as the primary mechanism to make these objects adaptable. That is, common behavior of metaheuristics is factored out and grouped in generic classes, applying static type variation. Metaheuristics template classes are parameterized by aspects such as solution spaces and neighborhood structures.

Applications

Applications of metaheuristics are almost uncountable and appear in various journals (e. g., *Journal of Heuristics*), books, and technical reports every day. A helpful source for a subset of successful applications may be special issues of journals or compilations such as [25, 77,79,97], just to mention some.

Specialized conferences like the Metaheuristics International Conference are devoted to the topic [25, 59,72,80,81,97] and even more general conferences reveal that metaheuristics have become part of necessary prerequisites for successfully solving optimization problems [46]. Moreover, ready-to-use systems such as class libraries and frameworks have been developed, although they are usually restricted to application by the *knowledgeable* user.

Specialized applications also reveal research needs, e. g., in dynamic environments. One example refers to the application of metaheuristics for online optimization [49].

Conclusions

Over the last few decades metaheuristics have become a substantial part of the optimization stockroom with various applications in science and, even more important, in practice. Metaheuristics have been considered in textbooks, e. g., in operations research, and a wealth of monographs [27,43,70,92] are available. Most important in our view are general frames. AMP, an intelligent interplay of intensification and diversification (such as ideas from POPMUSIC), and the connection to powerful exact algorithms as subroutines for handable subproblems are avenues to be followed.

From a theoretical point of view, the use of most metaheuristics has not yet been fully justified. While convergence results regarding solution quality exist for most metaheuristics, once appropriate probabilistic assumptions are made [8,31,50] these turn out not to be very helpful in practice as usually a disproportionate computation time is required to achieve these results (usually convergence is achieved for a computation time tending to infinity, with a few exceptions, e. g., for the reverse elimination method within TS or the pilot method where optimality can be achieved with a finite, but exponential number of steps in the worst case). Furthermore, we have to admit that theoretically one may argue that none of the metaheuristics described are *on average* better than any other; there is *no free lunch* [101]. Basically this leaves the choice of a best possible heuristic or related ingredients to the ingenuity of the user/researcher. Some researchers related the term "hyperheuristics" to the question of which (heuristic) method among a given set of methods to choose for a given problem [16].

Moreover, despite the widespread success of various metaheuristics, researchers occasionally still have a poor understanding of many key theoretical aspects of these algorithms, including models of the high-level run-time dynamics and identification of search space features that influence problem difficulty [99]. Moreover, fitness landscape evaluations are considered to be in their infancy too.

From an empirical standpoint it would be most interesting to know which algorithms perform best under various criteria for different classes of problems. Unfortunately, this theme is out of reach as long as we do not have any well-accepted standards regarding the testing and comparison of different methods.

While most papers on metaheuristics claim to provide "high-quality" results based on some sort of measure, we still believe that there is a great deal of room for improvement in testing existing as well as new approaches from an empirical point of view [10,57,67]. In a dynamic research process numerical results provide the basis for systematically developing efficient algorithms. The essential conclusions of finished research and development processes should always be substantiated (i. e., empirically and, if necessary, statistically proven) by numerical results based on an appropriate empirical test cycle. Furthermore, even when excellent numerical results are obtained, it may still be possible to compare with a simple random restart procedure and obtain better results in some cases [47]. However, this comparison is usually neglected.

Usually the ways of preparing, performing, and presenting experiments and their results are significantly different. The failing of a generally accepted standard for testing and reporting on the testing, or at least a corresponding guideline for designing experiments, unfortunately implies the following observation: Some results can be used only in a restricted way, e. g., because relevant data are missing, wrong environmental settings are used, or simply results are glossed over. In the worst case nonsufficiently prepared experiments provide results that are unfit for further use, i. e., any generalized conclusion is out of reach. Future algorithm research needs to provide effective methods for analyzing the performance of, e. g., heuristics in a more scientifically founded way (see [4,100] for some steps into this direction).

A final aspect that deserves special consideration is to investigate the use of information within different metaheuristics. While the AMP frame provides a very good entry into this area, this still provides an interesting opportunity to link artificial intelligence with operations research concepts.

References

1. Aarts EHL, Lenstra JK (eds) (1997) Local Search in Combinatorial Optimization. Wiley, Chichester
2. Aarts EHL, Verhoeven M (1997) Local search. In: Dell'Amico M, Maffioli F, Martello S (eds) Annotated Bibliographies in Combinatorial Optimization. Wiley, Chichester, pp 163–180
3. Achterberg T, Berthold T (2007) Improving the feasibility pump. Discret Optim 4:77–86
4. Adenso-Diaz B, Laguna M (2006) Fine-tuning of algorithms using fractional experimental designs and local search. Oper Res 54:99–114
5. Ahuja RK, Ergun O, Orlin JB, Punnen AB (2002) A survey of very large-scale neighborhood search techniques. Discret Appl Math 123:75–102
6. Alba E (ed) (2005) Parallel Metaheuristics. Wiley, Hoboken
7. Alba E, Marti R (eds) (2006) Metaheuristic Procedures for Training Neural Networks. Springer, New York
8. Althöfer I, Koschnick KU (1991) On the convergence of 'threshold accepting'. Appl Math Optim 24:183–195
9. Bäck T, Fogel DB, Michalewicz Z (eds) (1997) Handbook of Evolutionary Computation. Institute of Physics Publishing, Bristol
10. Barr RS, Golden BL, Kelly JP, Resende MGC, Stewart WR (1995) Designing and reporting on computational experiments with heuristic methods. J Heuristics 1:9–32
11. Bastos MB, Ribeiro CC (2002) Reactive tabu search with path relinking for the Steiner problem in graphs. In: Ribeiro CC, Hansen P (eds) Essays and Surveys in Metaheuristics. Kluwer, Boston, pp 39–58
12. Battiti R, Tecchiolli G (1994) The reactive tabu search. ORSA J Comput 6:126–140
13. Bertsekas DP, Tsitsiklis JN, Wu C (1997) Rollout algorithms for combinatorial optimization. J Heuristics 3:245–262
14. Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: Overview conceptual comparison. ACM Comput Surv 35:268–308
15. Bonabeau E, Dorigo M, Theraulaz G (eds) (1999) Swarm Intelligence – From Natural to Artificial Systems. Oxford University Press, New York
16. Burke EK, Kendall G, Newall J, Hart E, Ross P, Schulenburg S (2003) Hyper-heuristics: An emerging direction in modern search technology. In: Glover FW, Kochenberger GA (eds) Handbook of Metaheuristics. Kluwer, Boston, pp 457–474
17. Caseau Y, Laburthe F, Silverstein G (1999) A metaheuristic factory for vehicle routing problems. Lect Notes Comput Sci 1713:144–158
18. Cerulli R, Fink A, Gentili M, Voß S (2006) Extensions of the minimum labelling spanning tree problem. J Telecommun Inf Technol 4/2006:39–45
19. Charon I, Hudry O (1993) The noising method: A new method for combinatorial optimization. Oper Res Lett 14:133–137
20. Crainic TG, Toulouse M, Gendreau M (1997) Toward a taxonomy of parallel tabu search heuristics. INFORMS J Comput 9:61–72
21. de Backer B, Furnon V, Shaw P, Kilby P, Prosser P (2000) Solving vehicle routing problems using constraint programming and metaheuristics. J Heuristics 6:501–523

22. Di Gaspero L, Schaerf A (2003) EASYLOCAL++: An object-oriented framework for the flexible design of local-search algorithms. Softw Pr Experience 33:733–765
23. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: Optimization by a colony of cooperating agents. IEEE Trans Syst, Man Cybern B 26:29–41
24. Dorigo M, Stützle T (2004) Ant Colony Optimization. MIT Press, Cambridge
25. Dörner KF, Gendreau M, Greistorfer P, Gutjahr WJ, Hartl RF, Reimann M (eds) (2007) Metaheuristics: Progress in Complex Systems Optimization. Springer, New York
26. Dowsland KA (1993) Simulated annealing. In: Reeves C (ed) Modern Heuristic Techniques for Combinatorial Problems. Halsted, Blackwell, pp 20–69
27. Dreo J, Petrowski A, Siarry P, Taillard E (2006) Metaheuristics for Hard Optimization. Springer, Berlin
28. Dueck G, Scheuer T (1990) Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. J Comput Phys 90:161–175
29. Duin CW, Voß S (1994) Steiner tree heuristics – a survey. In: Dyckhoff H, Derigs U, Salomon M, Tijms HC (eds) Operations Research Proceedings. Springer, Berlin, pp 485–496
30. Duin CW, Voß S (1999) The pilot method: A strategy for heuristic repetition with application to the Steiner problem in graphs. Netw 34:181–191
31. Faigle U, Kern W (1992) Some convergence results for probabilistic tabu search. ORSA J Comput 4:32–37
32. Festa P, Resende MGC (2004) An annotated bibliography of GRASP. Technical report, AT&T Labs Research, Florham Park
33. Fink A, Voß S (2002) HotFrame: A heuristic optimization framework. In: Voß S, Woodruff DL (eds) Optimization Software Class Libraries. Kluwer, Boston, pp 81–154
34. Fischetti M, Glover F, Lodi A (2005) The feasibility pump. Math Program A 104:91–104
35. Fischetti M, Lodi A (2003) Local branching. Math Program B 98:23–47
36. Fogel DB (1993) On the philosophical differences between evolutionary algorithms and genetic algorithms. In: Fogel DB, Atmar W (eds) Proceedings of the Second Annual Conference on Evolutionary Programming, Evolutionary Programming Society, La Jolla, pp 23–29
37. Fogel DB (1995) Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, New York
38. Glover F (1977) Heuristics for integer programming using surrogate constraints. Decis Sci 8:156–166
39. Glover F (1986) Future paths for integer programming links to artificial intelligence. Comput Oper Res 13:533–549
40. Glover F (1990) Tabu search – Part II. ORSA J Comput 2:4–32
41. Glover F (1995) Scatter search and star-paths: beyond the genetic metaphor. OR Spektrum 17:125–137
42. Glover F (1997) Tabu search and adaptive memory programming – Advances, applications challenges. In: Barr RS, Helgason RV, Kennington JL (eds) Interfaces in computer science and operations research: Advances in metaheuristics, optimization and stochastic modeling technologies. Kluwer, Boston, pp 1–75
43. Glover F, Laguna M (1997) Tabu Search. Kluwer, Boston
44. Glover FW, Kochenberger GA (eds) (2003) Handbook of Metaheuristics. Kluwer, Boston
45. Goldberg DE (1989) Genetic Algorithms in Search, Optimization, Machine Learning. Addison-Wesley, Reading
46. Golden BL, Raghavan S, Wasil EA (eds) (2005) The Next Wave in Computing, Optimization, Decision Technologies. Kluwer, Boston
47. Gomes AM, Oliveira JF (2006) Solving irregular strip packing problems by hybridising simulated annealing and linear programming. Eur J Oper Res 171:811–829
48. Greistorfer P, Voß S (2005) Controlled pool maintenance for meta-heuristics. In: Rego C, Alidaee B (eds) Metaheuristic optimization via memory evolution. Kluwer, Boston, pp 387–424
49. Gutenschwager K, Niklaus C, Voß S (2004) Dispatching of an electric monorail system: Applying meta-heuristics to an online pickup and delivery problem. Transp Sci 38:434–446
50. Hajek B (1988) Cooling schedules for optimal annealing. Math Oper Res 13:311–329
51. Hansen P, Mladenović N (1999) An introduction to variable neighborhood search. In: Voß S, Martello S, Osman IH, Roucairol C (eds) Meta-heuristics: Advances and trends in local search paradigms for optimization. Kluwer, Boston, pp 433–458
52. Hart JP, Shogan AW (1987) Semi-greedy heuristics: An empirical study. Oper Res Lett 6:107–114
53. Harvey W, Ginsberg M (1995) Limited discrepancy search. In: Proceedings of the 14th IJCAI. Morgan Kaufmann, San Mateo, pp 607–615
54. Hertz A, Kobler D (2000) A framework for the description of evolutionary algorithms. Eur J Oper Res 126:1–12
55. Hoffmeister F, Bäck T (1991) Genetic algorithms and evolution strategies: Similarities and differences. Lect Notes Comput Sci 496:455–469
56. Holland JH (1975) Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor
57. Hooker JN (1995) Testing heuristics: We have it all wrong. J Heuristics 1:33–42
58. Hoos HH, Stützle T (2005) Stochastic Local Search – Foundations and Applications. Elsevier, Amsterdam
59. Ibaraki T, Nonobe K, Yagiura M (eds) (2005) Metaheuristics: Progress as Real Problem Solvers. Springer, New York
60. Ingber L (1996) Adaptive simulated annealing (ASA): Lessons learned. Control Cybern 25:33–54
61. Jaszkiewicz A (2004) A comparative study of multiple-objective metaheuristics on the bi-objective set covering

problem and the pareto memetic algorithm. Ann Oper Res 131:215–235
62. Johnson DS, Aragon CR, McGeoch LA, Schevon C (1989) Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. Oper Res 37:865–892
63. Kennedy J, Eberhart RC (2001) Swarm Intelligence. Elsevier, Amsterdam
64. Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
65. Laguna M, Martí R (2003) Scatter Search. Kluwer, Boston
66. Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling-salesman problem. Oper Res 21:498–516
67. McGeoch C (1996) Toward an experimental method for algorithm simulation. INFORMS J Comput 8:1–15
68. Meloni C, Pacciarelli D, Pranzo M (2004) A rollout metaheuristic for job shop scheduling problems. Ann Oper Res 131:215–235
69. Michalewicz Z (1999) Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Berlin
70. Michalewicz Z, Fogel DB (2004) How to Solve It: Modern Heuristics, 2nd edn. Springer, Berlin
71. Moscato P (1993) An introduction to population approaches for optimization and hierarchical objective functions: A discussion on the role of tabu search. Ann Oper Res 41:85–121
72. Osman IH, Kelly JP (eds) (1996) Meta-Heuristics: Theory and Applications. Kluwer, Boston
73. Pearl J (1984) Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, Reading
74. Pesant G, Gendreau M (1999) A constraint programming framework for local search methods. J Heuristics 5:255–279
75. Pesch E, Glover F (1997) TSP ejection chains. Discret Appl Math 76:165–182
76. Polya G (1945) How to solve it. Princeton University Press, Princeton
77. Rayward-Smith VJ, Osman IH, Reeves CR, Smith GD (eds) (1996) Modern Heuristic Search Methods. Wiley, Chichester
78. Reeves CR, Rowe JE (2002) Genetic Algorithms: Principles and Perspectives. Kluwer, Boston
79. Rego C, Alidaee B (eds) (2005) Metaheuristic optimization via memory and evolution. Kluwer, Boston
80. Resende MGC, de Sousa JP (eds) (2004) Metaheuristics: Computer Decision-Making. Kluwer, Boston
81. Ribeiro CC, Hansen P (eds) (2002) Essays and Surveys in Metaheuristics. Kluwer, Boston
82. Sakawa M (2001) Genetic algorithms and fuzzy multiobjective optimization. Kluwer, Boston
83. Schwefel HP, Bäck T (1998) Artificial evolution: How and why? In: Quagliarella D, Périaux J, Poloni C, Winter G (eds) Genetic Algorithms and Evolution Strategy in Engineering and Computer Science: Recent Advances and Industrial Applications, Wiley, Chichester, pp 1–19
84. Shaw P (1998) Using constraint programming local search methods to solve vehicle routing problems. Working paper, ILOG SA, Gentilly
85. Smith K (1999) Neural networks for combinatorial optimisation: A review of more than a decade of research. INFORMS J Comput 11:15–34
86. Sniedovich M, Voß S (2006) The corridor method: A dynamic programming inspired metaheuristic. Control Cybern 35:551–578
87. Storer RH, Wu SD, Vaccari R (1995) Problem and heuristic space search strategies for job shop scheduling. ORSA J Comput 7:453–467
88. Taillard E, Voß S (2002) POPMUSIC - partial optimization metaheuristic under special intensification conditions. In: Ribeiro CC, Hansen P (eds) Essays and Surveys in Metaheuristics. Kluwer, Boston, pp 613–629
89. Taillard ÉD, Gambardella LM, Gendreau M, Potvin JY (2001) Adaptive memory programming: A unified view of meta-heuristics. Eur J Oper Res 135:1–16
90. Vaessens RJM, Aarts EHL, Lenstra JK (1998) A local search template. Comput Oper Res 25:969–979
91. Verhoeven MGA, Aarts EHL (1995) Parallel local search techniques. J Heuristics 1:43–65
92. Voß S (1993) Intelligent Search. Manuscript, TU Darmstadt
93. Voß S (1993) Tabu search: applications and prospects. In: Du DZ, Pardalos P (eds) Network Optimization Problems. World Scientific, Singapore, pp 333–353
94. Voß S (1996) Observing logical interdependencies in tabu search: Methods and results. In: Rayward-Smith VJ, Osman IH, Reeves CR, Smith GD (eds) Modern Heuristic Search Methods. Wiley, Chichester, pp 41–59
95. Voß S (2001) Meta-heuristics: The state of the art. Lect Notes Artif Intell 2148:1–23
96. Voß S, Fink A, Duin C (2004) Looking ahead with the pilot method. Ann Oper Res 136:285–302
97. Voß S, Martello S, Osman IH, Roucairol C (eds) (1999) Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer, Boston
98. Voß S, Woodruff DL (eds) (2002) Optimization Software Class Libraries. Kluwer, Boston
99. Watson JP, Whitley LD, Howe AE (2005) Linking search space structure, run-time dynamics, and problem difficulty: A step toward demystifying tabu search. J Artif Intell Res 24:221–261
100. Whitley D, Rana S, Dzubera J, Mathias KE (1996) Evaluating evolutionary algorithms. Artif Intell 85:245–276
101. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82
102. Woodruff DL (1998) Proposals for chunking and tabu search. Eur J Oper Res 106:585–598
103. Woodruff DL (1999) A chunking based selection strategy for integrating meta-heuristics with branch and

bound. In: Voß S, Martello S, Osman IH, Roucairol C (eds) Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer, Boston, pp 499–511

Metropolis, Nicholas Constantine

PANOS M. PARDALOS
Center for Applied Optim. Department Industrial and Systems Engineering, University Florida, Gainesville, USA

MSC2000: 90C05, 90C25

Article Outline

Keywords

Metropolis; Simulated annealing; Monte-Carlo method

Nicholas Constantine Metropolis was born in Chicago on June 11, 1915 and died on October 17, 1999 in Los Alamos. At Los Alamos, Metropolis was the main driving force behind the development of the MANIAC series of electronic computers. He was the first to code a problem for the ENIAC in 1945–1946 (together with S. Frankel), a task which consumed approximately 1,000,000 IBM punched cards.

Metropolis received his PhD in physics from the University of Chicago in 1941. He went to Los Alamos in 1943 as a member of the initial staff of fifty scientists of the Manhattan Project. He spent his entire career at Los Alamos, except for two periods (1946–1948 and 1957–1965), during which he was professor of Physics at the University of Chicago.

Metropolis is best known for the development (joint with S. Ulam and J. von Neumann) of the *Monte-Carlo method*. The Monte-Carlo method provides approximate solutions to a variety of mathematical problems by performing statistical sampling experiments on a computer. However, the real use of Monte-Carlo methods as a research tool stems from work on the atomic bomb during the second world war. This work involved a direct simulation of the probabilistic problems concerned with random neutron diffusion in fissile material. Metropolis and his collaborators, obtained Monte-Carlo estimates for the eigenvalues of Schrodinger equation.

In 1953, Metropolis co-authored the first paper on the technique that came to be known as *simulated annealing* [3,8]. Simulated annealing is a method for solving optimization problems. The name of the algorithm derives from an analogy between the simulation of the annealing of solids. Annealing refers to a process of cooling material slowly until it reaches a stable state.

Metropolis also made several early contributions to the use of computers in the exploration of nonlinear dynamics. In the Sixties and Seventies he collaborated with G.-C. Rota and others on significance arithmetic. Another contribution of Metropolis to numerical analysis is an early paper on the use of *Chebyshev's iterative method* for solving large scale linear systems [1].

References

1. Blair A, Metropolis N, von Neumann J, Taub AH, Tsingou M (1959) A study of a numerical solution to a two-dimensional hydrodynamical problem. Math Tables and Other Aids to Computation 13(67):145–184
2. Harlow F, Metropolis N (1983) Computing and computers: Weapons simulation leads to the computer era. Los Alamos Sci 7:132–141
3. Kirkpatrick S, Gelatt CD, Vecchi MP Jr (1983) Optimization by simulated annealing. Science 220(4598):671–680
4. Metropolis N (1987) The beginning of the Monte Carlo method. Los Alamos Sci 15:125–130
5. Metropolis N (1992) The age of computing: A personal memoir. Daedalus 121(1):87–103
6. Metropolis N, Howlett J, Rota G-C (eds) (1980) A history of computing in the twentieth century. Acad. Press, New York
7. Metropolis N, Nelson EC (oct. 1982) Early computing at Los Alamos. Ann Hist Comput 4(4):348–357
8. Metropolis N, Rosenbluth A, Teller A, Teller E (1953) Equation of state calculation by fast computing machines. J Chem Phys 21:1087–1092

Minimax: Directional Differentiability

VLADIMIR F. DEMYANOV
St. Petersburg State University, St. Petersburg, Russia

MSC2000: 90C30, 65K05

Article Outline

Keywords

Minimax problem; Max-function; Maxmin function; Directional derivative; Higher-order derivatives; Hypodifferentiability; Support function

Minimax is a principle of optimal choice (of some parameters or functions). If applied, this principle requires to find extremal values of some max-type function. Since the operation of taking the pointwise maximum (of a finite or infinite number of functions) generates, in general, a nonsmooth function, it is important to study properties of such a function. Fortunately enough, though a max-function is not differentiable, in many cases it is still directionally differentiable. The directional differentiability provides a tool for formulating necessary (and sometimes sufficient) conditions for a minimum or maximum and for constructing numerical algorithms.

Recall that a function $f:\mathbf{R}^n \to \mathbf{R}$ is called *Hadamard directionally differentiable* (H.d.d.) at a point $x \in \mathbf{R}^n$ if for any $g \in \mathbf{R}^n$ there exists the finite limit

$$f'_{\mathrm{H}}(x,g) = \lim_{[\alpha,g']\to[+0,g]} \frac{f(x+\alpha g') - f(x)}{\alpha}.$$

A function $f\colon \mathbf{R}^n \to \mathbf{R}$ is called *Dini directionally differentiable* (D.d.d.) at a point $x \in \mathbf{R}^n$ if for any $g \in \mathbf{R}^n$ there exists the finite limit

$$f'_{\mathrm{D}}(x,g) = \lim_{\alpha\downarrow 0} \frac{f(x+\alpha g) - f(x)}{\alpha}.$$

If f is H.d.d., then it is D.d.d. as well and $f_{\mathrm{H}}{}'(x,g) = f_D{}'(x,g)$.

Let $\Omega \subset \mathbf{R}^n$ be a convex compact set, $x \in \Omega$. The cone

$$N_x(\Omega) = \{v \in \mathbb{R}^n\colon\ (v,x) = \rho_\Omega(x)\}$$

is called *normal* to Ω at x. Here

$$\rho_\Omega(x) = \max_{y\in\Omega}(v,y)$$

is the *support function* of Ω at x.

A max-function

Let

$$f(x) = \max_{y\in G} \varphi(x,y), \tag{1}$$

where $\varphi\colon S \times G \to \mathbf{R}$ is continuous jointly in x, y on $S \times G$ and continuously differentiable in x there, $S \subset \mathbf{R}^n$ is an open set, G is a compact set of some space. Under the conditions stated, the function f is continuous on S.

Proposition 1 *The function f is H.d.d. at any point $x \in S$ and*

$$f'_H(x,g) = \max_{y\in R(x)} (\varphi'_x(x,y),g) = \max_{v\in\partial f(x)} (v,g), \tag{2}$$

where

$$R(x) = \{y \in G\colon\ f(x) = \varphi(x,y)\},$$

$\varphi_x{}'(x, g)$ is the gradient of φ with respect to x for a fixed y, (a, b) is the scalar product of vectors a and b,

$$\partial f(x) = \operatorname{co}\left\{\varphi'_x(x,g)\colon\ y \in R(x)\right\} \subset \mathbb{R}^n.$$

The set $\partial f(x)$ is called the *subdifferential* of f at x. It is convex and compact. The mapping ∂f is, in general, discontinuous.

Remark 2 It turns out that a convex function can also be represented in the form (1) with φ being *affine* in x. For this special (convex) case the set $\partial f(x)$ is

$$\begin{aligned}&\partial f(x)\\ &= \{v \in \mathbb{R}^n\colon\ f(z) - f(x) \geq (v, z-x),\ \forall z \in S\}.\end{aligned}$$

The discovery of the directional differentiability of max-functions ([1,2,6]) and convex functions [10] was a breakthrough and led to the development of *minimax theory* and convex analysis ([4,9,10]).

A Maximum Function with Dependent Constraints

Let $x \subset \mathbf{R}^n$, $Y \subset \mathbf{R}^m$ be open sets and let

$$f(x) = \max_{y \in a(x)} \varphi(x, g), \tag{3}$$

where $a(x)$ is a multivalued mapping with compact images, φ: $X \times Y \to \mathbf{R}$ is Hadamard differentiable as a function of two variables, i. e. there exists the limit

$$\varphi'_{\mathrm{H}}([x, y], [g, v]) = \lim_{[\alpha, g', v'] \to [+0, g, v]} \frac{1}{\alpha} \cdot \left[\varphi(x + \alpha g', y + \alpha v') - \varphi(x, y)\right].$$

Then φ is continuous and $\varphi_{\mathrm{H}}{}'$ is continuous as a function of direction $[g, v]$.

The function f is called a *maximum function with dependent constraints*. Such functions are of great importance and have widely been studied (see [3,5,7,8]). To illustrate the results let us formulate one of them [5, Thm. I, 6.3].

Proposition 3 *Let a mapping a be closed and bounded, its images be convex and compact, the support function* $a(x, l) = \max_{v \in a(x)} (v, l)$ *be uniformly differentiable with respect to parameter l. Let, further,* $x \in X$ *and a function* φ *be concave in some convex neighborhood of the set* $\{[x, y]: y \in R(x)\}$ *(where* $R(x) = \{y \in a(x): \varphi(x, y) = f(x)\}$*). Then f (see (3)) is H.d.d. and*

$$f'(x, g) = \sup_{y \in R(x)} \min_{[l_1, l_2] \in V(x, y)} [(l_1, g) + a'(x, l_2; g)], \tag{4}$$

where

$$V(x, y) = \left\{l = [l_1, l_2] \in \overline{\partial}\varphi(x, y): \ l_2 \in N_{x,y}\right\},$$

$\overline{\partial}\varphi(x, y)$ *is the superdifferential of* φ *at the point* $[x, y]$*, and* $N_{x,y}$ *is the cone normal to* $a(x)$ *at y.*

Recall that if a function F: $\mathbf{R}^s \to \mathbf{R}$ is concave, $Z \subset \mathbf{R}^s$ is open, $z \in Z$, then the set

$$\overline{\partial}F(z) = \left\{ v \in \mathbb{R}^s : \begin{array}{c} F(z') - F(z) \le (v, z' - z), \\ \forall z' \in Z \end{array} \right\}$$

is called the *superdifferential* of F at $z \in Z$. It is convex and compact.

A Maxmin Function

Let $\varphi(x, y, z)$: $S \times G_1 \times G_2 \to \mathbf{R}$ be continuous jointly in all variables, $S \subset \mathbf{R}^n$ be an open set, $G_1 \subset \mathbf{R}^m$, $G_2 \subset \mathbf{R}^p$ be compact. Put

$$f(x) = \max_{y \in G_1} \min_{z \in G_2} \varphi(x, y, z). \tag{5}$$

The function f is continuous on S.

Let

$$\Phi(x, y) = \min_{z \in G_2} \varphi(x, y, z),$$
$$R(x) = \{y \in G_1 : \ \Phi(x, y) = f(x)\},$$
$$Q(x, y) = \{z \in G_2 : \ \varphi(x, y, z) = \Phi(x, y)\}.$$

Fix $x \in S$, let $D_\varepsilon(\varepsilon > 0)$ be an ε-neighborhood of the set $\{x\} \times R(x) \times \cup_{y \in R(x)} Q(x, y)$. Assume that the derivatives

$$\frac{\partial \varphi}{\partial x}, \ \frac{\partial \varphi}{\partial y}, \ \frac{\partial^2 \varphi}{\partial x^2}, \ \frac{\partial^2 \varphi}{\partial x \partial y}, \ \frac{\partial^2 \varphi}{\partial y^2}$$

exist and are continuous jointly in all variables on $D_\varepsilon(x)$ and that

$$\left(\frac{\partial^2 \varphi(\overline{x}, y, z)}{\partial y^2} v, v\right) \le 0, \qquad \forall [\overline{x}, y, z] \in D_\varepsilon(x), \quad v \in \mathbb{R}^m.$$

Assume also that G_1 is convex. Let $y \in G_1$. Put

$$\gamma(y) = \left\{v = \lambda(y' - y): \ \lambda > 0, \ y' \in G_1\right\},$$
$$\Gamma(y) = \operatorname{cl} \gamma(y).$$

Proposition 4 [3, Thm. 5.2] *Under the above assumptions the function f (see (5)) is Hadamard directionally differentiable and*

$$f'_H(x, g) = \sup_{y \in R(x)} \sup_{y \in \Gamma(y)} \min_{z \in Q(x, y)} \left[\left(\frac{\partial \varphi(x, y, z)}{\partial y}, v\right) + \left(\frac{\partial \varphi(x, y, z)}{\partial x}, g\right)\right].$$

Remark 5 More sophisticated results on the directional differentiability of max- and maxmin functions can be found, e. g., in [8].

Higher-Order Directional Derivatives

The results above are related to the first order directional derivatives. Using these derivatives, it is possible to construct the following first order expansion:

$$f(x+\alpha g) = f(x) + \alpha f'(x,g) + o_{x,g}(\alpha), \tag{6}$$

where f' is either $f_H{}'$ or $f_D{}'$.

In some cases it is possible to get 'higher-order' expansions.

Let

$$f(x) = \max_{i \in I} f_i(x), \tag{7}$$

where $I = 1 : N$, $x = (x_1, \dots, x_n) \in \mathbf{R}^n$, the f_i's are continuous and continuously differentiable up the lth order on an open set $S \subset \mathbf{R}^r$. Fix $x \in S$. Then for sufficiently small $\alpha > 0$

$$f_i(x+\alpha g) = f_i(x) + \sum_{k=1}^{l} \frac{\alpha^k}{k!} f_i^{(k)}(x,g) + o_i(g,\alpha^l), \tag{8}$$

where

$$f_i^{(k)}(x,g) = \sum_{j_1,\dots,j_k=1}^{n} \frac{\partial^k f_i(x)}{\partial x_{j_1} \cdots \partial x_{j_k}} g_{j_1}, \dots, g_{j_k},$$

$$k \in 1,\dots,l, \quad \frac{o_i(g,\alpha^l)}{\alpha^l} \underset{\alpha\downarrow 0}{\to} 0 \tag{9}$$

uniformly with respect to g, $\|g\| = 1$.

Let us use the following notation

$$f_i^0(x,g) = f_i(x),$$
$$\forall i \in I, \quad R_0(x,g) = I,$$
$$R_k(x,g) = \{i \in R_{k-1}(x,g) :$$
$$f_i^{(k-1)}(x,g) = \max_{j \in R_{k-1}(x,g)} f_j^{(k-1)}(x,g)\Big\},$$
$$k \in 1,\dots,l.$$

Clearly

$$R_0(x,g) \supset R_1(x,g) \supset R_2(x,g) \supset \cdots.$$

Note that $R_0(x,g)$ does not depend on x and g, and $R_1(x, g)$ does not depend on g.

Proposition 6 **[3, Thm. 9.1]** *The following expansion holds:*

$$f(x+\alpha g) = f(x) + \sum_{k=1}^{l} \frac{\alpha^k}{k!} f^{(k)}(x,g) + o(g,\alpha^l),$$
$$\forall g \in \mathbb{R}^n, \tag{10}$$

where

$$f^{(k)}(x,g) = \max_{i \in R_k(x,g)} f_i^{(k)}(x,g),$$
$$\frac{o(g,\alpha^l)}{\alpha^l} \underset{\alpha\downarrow 0}{\to} 0 \tag{11}$$

uniformly with respect to g, $\|g\| = 1$.

The value $\partial^k f(x)/\partial g^k = f^{(k)}(x,g)$ is called the kth *derivative* of f at x in a direction g.

Remark 7 The mapping $R_1(x,g)$ is not continuous in x, while the mappings $R_k(x,g)$ ($k \geq 2$) are not continuous in x as well as in g. Therefore the functions $f^{(k)}(x,g)$ in (11) are not continuous in x and (if $k \geq 2$) in g and, as a result, expansion (6) is also not 'stable' in x.

To overcome this difficulty we shall employ another tool.

Hypodifferentiability of a Max Function

Let us again consider the case where f is defined by (7). It follows from (8) that, for $\Delta = (\Delta_1, \dots, \Delta_n) \in \mathbf{R}^n$,

$$f(x+\Delta) = \max_{i\in I}\left[f_i(x) + \sum_{k=1}^{l} \frac{1}{k!} f_i^{(k)}(x,\Delta)\right] + o(\|\Delta\|^k). \tag{12}$$

Let us use the notation (see (9))

$$f_i^{(k)}(x,\Delta) = A_{ik}\Delta^k.$$

The function $f_i^{(k)}(x,\Delta)$ is a *kth order form of coordinates* $\Delta_1, \dots, \Delta_n$; A_{ik} being the set of coefficients of this form. Then (12) can be rewritten as

$$\begin{aligned} f(x+\Delta) &= \max_{i\in I}\left[f_i(x) + \sum_{k=1}^{l} \frac{1}{k!} A_{ik}\Delta^k\right] + o(\|\Delta\|^k) \\ &= f(x) + \max_{A \in d^l f(x)}\left[\sum_{k=1}^{l} \frac{1}{k!} A_k\Delta^k\right] + o(\|\Delta\|^k), \end{aligned} \tag{13}$$

where

$$\begin{aligned} &d^l f(x) = \mathrm{co}\left\{A^{(i)} = (A_{i0}, \dots, A_{il}) \colon\ i \in I\right\},\\ &A_{i0} = f_i(x) - f(x), \quad A = (A_0, \dots, A_l),\\ &A_0 \in \mathbb{R},\ A_1 \in \mathbb{R}^n,\\ &A_2 \in \mathbb{R}^{n\times n}, \dots, A_k \in \overbrace{\mathbb{R}^{n\times\dots\times n}}^{k \text{ times}}. \end{aligned}$$

Here, $\overbrace{\mathbb{R}^{n\times\dots\times n}}^{k \text{ times}}$ is the space of kth order real forms, e. g. $\mathbf{R}^{n\times n}$ is the space of real $(n \times n)$-matrices.

The set $d^l f(x)$ is called the *kth order hypodifferential* of f at x. It is an element of the space $\mathbb{R} \times \mathbb{R}^n \times \dots \times \overbrace{\mathbb{R}^{n\times\dots\times n}}^{l}$. The mapping $d^l f$ is continuous in x.

Remark 8 Expansion (13) can be extended to the case where f is given by (1) and φ is l times continuously differentiable in x.

Max functions represent a special case of the class of *quasidifferentiable functions* (see [5]).

See also

- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Minimax Theorems
- ▶ Nondifferentiable Optimization: Minimax Problems
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Quasigradient Methods in Minimax Problems

References

1. Danskin JM (1967) The theory of max-min and its application to weapons allocation problems. Springer, Berlin
2. Demyanov VF (1966) On minimizing the maximal deviation. Vestn Leningrad Univ 7:21–28
3. Demyanov VF (1974) Minimax: Directional differentiability. Leningrad Univ. Press, Leningrad
4. Demyanov VF, Malozemov VN (1974) Introduction to minimax. Wiley, New York. Second edition: Dover, New York (1990)
5. Demyanov VF, Rubinov AM (1995) Constructive nonsmooth analysis. P. Lang, Frankfurt am Main
6. Girsanov IV (1965) Differentiability of solutions of the mathematical programming problems. Abstracts Conf. Applications of Functional Analysis Methods to Solving Nonlinear Problems, pp 43–45
7. Levitin ES (1994) Perturbation theory in mathematical programming and its applications. Wiley, New York
8. Minchenko LI, Borisenko OF (1992) Differential properties of marginal functions and their applications to optimization problems. Nauka i Techn., Minsk
9. Pschenichny BN (1980) Convex analysis and extremal problems. Nauka, Moscow
10. Rockafellar RT (1970) Convex analysis. Princeton Univ. Press, Princeton

Minimax Game Tree Searching

Claude G. Diderich[1], Marc Gengler[2]
[1] Computer Sci. Department, Swiss Federal Institute Technology-Lausanne, Lausanne, Switzerland
[2] Ecole Sup. d'Ingénieurs de Luminy, Université Méditerrannée, Marseille, France

MSC2000: 49J35, 49K35, 62C20, 91A05, 91A40

Article Outline

Keywords

Algorithms; Games; Minimax; Searching

With the introduction of computers, also started the interest in having machines play games. Programming a computer such that it could play, for example chess, was seen as giving it some kind of intelligence. Starting in the mid fifties, a theory on how to play two player zero sum perfect information games, like chess or go, was developed. This theory is essentially based on traversing a tree called minimax or game tree. An edge in the tree represents a move by either of the players and a node a configuration of the game.

Two major algorithms have emerged to compute the best sequence of moves in such a minimax tree. On one hand, there is the alpha-beta algorithm suggested around 1956 by I. McCarthy and first published in [27]. On the other hand, G.C. Stockman [29] introduced the SSS∗ algorithm. Both methods try to minimize the number of nodes explored in the game tree using special traversal strategies and cut conditions.

Minimax Trees

A *two-player zero-sum perfect-information game*, also called *minimax game*, is a game which involves exactly two players who alternatively make moves. No information is hidden from the adversary. No coins are tossed, that is, the game is completely deterministic, and there is perfect symmetry in the quality of the moves allowed. Go, checker and chess are such minimax games whereas backgammon (the outcome of a die determines the moves available) or card games (cards are hidden from the adversary) are not.

A *minimax tree* or *game tree* is a tree where each node represents a state of the game and each edge a possible move. Nodes are alternatively labeled 'max' and 'min' representing either player's turn. A node having no descendants represents a final outcome of the game. The goal of a game is to find a winning sequence of moves, given that the opponent always plays his best move.

The quality of a node t in the minimax game tree, representing a configuration, is given by its value $e(t)$. The value $e(t)$, also called *minimax value*, is defined recursively as

$$e(t) = \begin{cases} f(t) & \text{if } t \text{ is a leave node,} \\ \max\limits_{s \in \text{sons}(t)} e(s) & \text{if } t \text{ is labeled 'max',} \\ \min\limits_{s \in \text{sons}(t)} e(s) & \text{if } t \text{ is labeled 'min'.} \end{cases}$$

If the considered minimax tree represents a *complete game*, that is, all possible board configurations, the function f may be defined as follows:

$$f(t) = \begin{cases} +1 & \text{if } t \text{ leads to a winning position,} \\ 0 & \text{if } t \text{ leads to a tie position,} \\ -1 & \text{if } t \text{ leads to a losing position;} \end{cases}$$

otherwise $f(t)$ represents an evaluation of the quality of a board position.

The relation between minimax trees and games is detailed in the following table.

Minimax tree notion	Minimax game notion
Minimax tree	All board configurations
Node in the tree	Board configuration
Edge from "max" to "min" node	Move by player "max"
Edge from "min" to "max" node	Move by player "min"
Node value	Quality of board position
Leave node	Outcome of a game
Solution path	Sequence of moves leading the best outcome

Sequential Minimax Game Tree Algorithms

Let t be a node of a minimax tree. Then the function first_son(t) returns the first son node s_1 of t and next_son(s_i, t) returns the $i + 1$th son of node t. The function no_more_sons(s, t) returns true of s is the last son of t. Otherwise it returns false. The ordering of the sons introduced by these functions is arbitrary. In practice it is given by some heuristic function. The function father(t) returns the father node of t, is_leave(t) whether or not t is a leave node and node_type(t) the type of node t.

Minimax Algorithm

The most basic minimax algorithm is called the *minimax algorithm*. It systematically traverses, in a depth first, left to right fashion, the complete minimax tree. All nodes are visited exactly once.

Alpha-Beta Algorithm

The first nontrivial algorithm introduced to compute the minimax value of a game tree was the *alpha-beta algorithm*. According to D. Knuth and R. Moore, McCarthy's comments at the Dartmouth summer research conference on artificial intelligence led to the use of alpha-beta pruning in game playing programs since the late 1950s. The first published discussion of an algorithm for minimax tree pruning appeared in 1958 (see [11, p. 56]). Two early extensive studies of the algorithm may be found in [18] and [27].

The idea behind the alpha-beta algorithm is to traverse the minimax tree in a depth first, left to right fashion. It tries to prune sub-trees that can not influence the minimax value of the tree. The conditions used to prune sub-trees are called *cut conditions*. The idea behind the suggested cut conditions is to associate to each node a lower and an upper bound, called α and β bounds. The bounds of a node are passed to its sons and tightened during the execution of the algorithm. It is easy to see that if the lower bound of a node t of type 'max' is larger than its upper bound then all not visited sons of node t can be pruned, and similar for nodes of type 'min'.

```
FUNCTION AlphaBeta($n, \alpha, \beta$) IS
BEGIN
    IF is_leave($n$) THEN RETURN $f(n)$
    $s \leftarrow$ first_son($n$)
    IF node_type($n$)=max THEN
        LOOP
            $\alpha \leftarrow \max\{\alpha, \text{AlphaBeta}(s, \alpha, \beta)\}$
            IF $\alpha \geq \beta$ THEN RETURN $\beta$
            EXIT LOOP WHEN no_more_sons($s, n$)
            $s \leftarrow$ next_son($s, n$)
        END LOOP
        RETURN $\alpha$
    ELSE
        LOOP
            $\beta \leftarrow \max\{\alpha, \text{AlphaBeta}(s, \alpha, \beta)\}$
            IF $\beta \leq \alpha$ THEN RETURN $\alpha$
            EXIT LOOP WHEN no_more_sons($s, n$)
            $s \leftarrow$next_sons($s, n$)
        END LOOP
        RETURN $\beta$
    END IF
END AlphaBeta
```

Pseudocode for the alpha-beta algorithm

It has been proved in [18] that the alpha-beta algorithm correctly calculates the minimax value of a tree. The above pseudocode describes the alpha-beta algorithm.

The minimax value of a tree T is computed as follows.

$$e\,(\text{root}(T)) \leftarrow \text{AlphaBeta}\,(\text{root}(T), -\infty, +\infty)\,.$$

Optimal State Space Search Algorithm SSS*

It has been introduced by Stockman in 1979, [29]. It originates not in game playing but in systematic pattern recognition. The algorithm was first analyzed and criticized in [26].

The idea behind the SSS* algorithm is to use a tree traversal strategy that is, better than the depth first and left to right strategy found in the alpha-beta algorithm. The criteria used to order the nodes yet to visit is an upper bound of their value. Nodes are stored in non increasing order of their upper bound in a list called 'open'.

The SSS* algorithm first traverses the minimax tree from top to bottom. Nodes whose sons have not yet been visited and which cannot yet be pruned are marked 'live'. Nodes marked 'solved' have already been visited once and have therefore their best upper bound associated.

The operation purge(t, open) removes all nodes from the open list for which the node t is an ancestor. Due to the fact that the nodes in the open list are sorted in nonincreasing order of their associated upper bound, the pruning operation only eliminates nodes that need no further consideration.

The SSS* algorithm is described by the following pseudocode.

```
FUNCTION SSS* IS
BEGIN
    open $\leftarrow \emptyset$
    insert (root, live, $+\infty$, open)
    LOOP
        $(s, t, m) \leftarrow$ remove (open)
        IF $s$ = root AND $t$ = solved THEN RETURN $m$
        $\langle$ Apply the $\Gamma$ operator to node $s$ $\rangle$
    END LOOP
END SSS*
```

Pseudocode for the SSS* algorithm

The operator $\Gamma(s)$ is applied to each node s extracted from the 'open' list.

It is possible to define a dual version of the SSS∗, which may be called *SSS∗ -dual*, in which the computation of upper bounds is replaced by the computation of lower bounds. The SSS∗ -dual algorithm has been suggested in [21].

Stockman has shown that if the SSS∗ algorithm explores a node, then this node is also explored by the alpha-beta algorithm. In fact, the alpha-beta algorithm loses efficiency (in the number of nodes visited) against the SSS∗ algorithm when the value of the minimax tree is found towards the right of the tree. If the SSS∗ algorithm is applied to win-lose trees then it visits exactly the same nodes in the same order as would the alpha-beta algorithm.

```
⟨Apply the $\Gamma$ operator to node $s$⟩ ≡
  IF $t$ = live AND node_type = max
    AND NOT is_leave($t$) THEN
    $s \leftarrow$ first_son($t$)
    LOOP
      insert ($s$, live, $m$, open)
      EXIT LOOP WHEN no_more_sons($s$, $t$)
      $s \leftarrow$ next_son($s$, $t$)
    END LOOP
  END IF
  IF $t$ = live AND node_type = min
    AND NOT is_leave($t$) THEN
    insert(first_son($t$), live, $m$, open)
  END IF
  IF $t$ = live AND is_leave($t$) THEN
    insert($t$, solved, min$\{f(t), m\}$, open)
  END IF
  IF $t$ = solved AND node_type = max
    AND NOT no_more_sons($t$, father($t$)) THEN
    insert(next_son($t$, father($t$)), live, $m$, open)
  END IF
  IF $t$ = solved AND node_type = max
    AND no_more_sons($t$, father($t$)) THEN
    insert(father($t$), solved, $m$, open)
  END IF
  IF $t$ = solved AND node_type = min THEN
    insert(father($t$), solved, $m$, open)
    purge(father($t$), open)
  END IF
```

SCOUT: Minimax Algorithm of Theoretical Interest

In the previous sections, we have described the most common minimax algorithms. While trying to show the optimality of the alpha-beta algorithm, J. Pearl [23] introduced the SCOUT algorithm. His idea was to show that the SCOUT algorithm is dominated by the alpha-beta algorithm and to prove that SCOUT achieves an optimal performance. But counterexamples showed that the alpha-beta algorithm does not dominate the SCOUT algorithm because the conservative testing approach of the SCOUT algorithm may sometimes cut off nodes that would have been explored by the alpha-beta algorithm.

The SCOUT algorithm itself recursively computes the value of the first of its sons. Then it tests to see if the value of the first son is better that the value of the other sons. In case of a negative result, the son that failed the test is completely evaluated by recursively calling SCOUT.

Although the SCOUT algorithm is more of theoretical interest, there are some problem instances where it outperforms all other minimax algorithms. A last advantage of the SCOUT algorithm versus one of its major competitors, the SSS∗ algorithm, is that its storage requirements are similar to those of the alpha-beta algorithm.

GSEARCH: Generalized Game Tree Search Algorithm

In 1986, T. Ibaraki [16] proposed a generalization of the previously known algorithms to compute the minimax value of a game tree. His idea was to use a branch and bound like approach. Nodes of the considered tree which have not yet been evaluated are stored in a list which is ordered according to a given criteria. Different orderings give different traversal strategies. A lower and upper bound is associated to each node. These bounds generalize the α and β values found in the alpha-beta algorithm.

Finally Ibaraki showed how the algorithm GS}CH is related to other minimax algorithms like alpha-beta or SSS∗, and proved that his algorithm always surpasses the alpha-beta algorithm.

SSS-2: Recursive State Space Search Algorithm

The SSS-2 algorithm has been proposed by W. Pijls and A. de Bruin [24]. It is based on the idea of computing an upper bound for the root node and then repeatedly transforming this upper bound into a tighter one. They have shown that the SSS-2 algorithm exactly expands the same nodes as those to which the SSS∗ algorithm applies the Γ operator.

Some Variations On The Subject

Computing the minimax value of a game tree may be seen as aspiring the solution value from a leave node through the whole tree up to the root node. While moving closer to the root node, more and more useless subtrees will be eliminated, as we have already stated for the alpha-beta algorithm. The better the α and β bounds, the more subtrees may be pruned. If, for instance, one knows that the minimax value will, with high probability, be found in the subset $]a, b[$, then it may be worth calling the alpha-beta algorithm as

$$e \leftarrow \text{AlphaBeta}\,(\text{root}(T), a, b)$$

If, indeed, the minimax value $e(\text{root}(T))$ belongs to the set $]a, b[$, then the algorithm will correctly return that value. If the minimax value does not belong to the set $]a, b[$, then the value returned will be either a or b, depending on whether the minimax value belongs to $]-\infty, a]$ or $[b, +\infty[$. We then say that the alpha-beta algorithm *failed low*, respectively *high*. In the case where the algorithm failed low, the call

$$e \leftarrow \text{AlphaBeta}\,(\text{root}(T), -\infty, a + 1)$$

will return the correct value. But it would also be possible to reiterate this procedure on a subset $]a_1, a + 1[$.

The technique of limiting the interval in which the solution may be found is called *aspiration search*. If the minimax value belongs to the specified interval, then a much larger number of cut conditions are verified and the tree actually traversed is much smaller than the one traversed by the alpha-beta algorithm without initial alpha and beta bounds.

Furthermore it is interesting to note that aspiration search is at the bases of a technique called *iterative deepening* which is used in many game playing programs.

I. Althöfer [5] suggested an *incremental negamax algorithm* which uses estimates of all nodes in the minimax tree, rather than only those of the leave nodes, to determine the value of the root node. This algorithm is useful when dealing with erroneous leave evaluation functions. Under the assumption of independently occurring and sufficiently small errors, the proposed algorithm is shown to have exponentially reduced error probabilities with respect to the depth of the tree.

R.L. Rivest [25] proposed an algorithm for searching minimax trees based on the idea of approximating the min and the max operators by generalized mean value operators. The approximation is used to guide the selection of the next leave node to expand, since the approximation allows to select efficiently that leave node upon whose value the minimax value most highly depends. B.W. Ballard [6] proposed a similar algorithm where the value of some nodes (the chance node as he calls them) is a, possibly weighted, average of the values of its sons. In fact he considers one additional type of nodes called chance nodes.

Conspiracy numbers have been introduced by D.A. McAllester in [22] as a measurement of the accuracy of the minimax value of an incomplete tree. They measure the number of leave nodes whose value must change in order to change the minimax value of the root node by a given amount.

Parallel Minimax Tree Algorithms

Parallelizing the minimax algorithm is trivial over uniform trees. Even on irregular trees, the parallelization remains easy. The only additional problem arises from the fact that the size of the subtrees to explore may now vary. Different processors will be attributed problems of varying computational volume. All what is needed then to achieve excellent speedups, is a load-balancing scheme, that is, a mechanism by means of which processors may, during run-time, exchange problems so as to keep all processors busy all the time.

The parallelization of the alpha-beta and the SSS∗ algorithms are much more interesting than the more theoretical minimax algorithm. There exist basically two approaches or techniques to parallelize the alpha-beta algorithm. In the first approach, which has been one of the first techniques used, all processors explore the entire tree but using different search-intervals. This approach is at the basic of the algorithm called *parallel aspiration search* by G. Baudet [7]. The second one consists in exploring simultaneously different parts of the minimax tree.

A Simple Way to Parallelize the Exploration of Minimax Trees

Exploring a minimax tree in parallel can very simply be obtained by generating the sons of the root node, and their sons and so on up to the point where one has as many son nodes waiting to be explored as there are processors. At this point, each processor explores the subtree rooted at one of these nodes, using any given sequential minimax algorithm. When all processors have completed their exploration, the solution for the entire tree is computed by using the partial results obtained from each of the processors.

In practice the sons of a node may be ordered in such a way that any son has a probability of yielding the locally optimal path that is no smaller than the corresponding probabilities for its right neighbors. The probability to find the optimum in the subtree rooted at a given son then always decreases when traversing the sons in a left to right order. Such ordering information is generally available in game-playing programs, the ordering function being a heuristic function based on the knowledge of the game to be played.

A Mandatory Work First Algorithm

R. Hewett and G. Krishnamurthy [15] proposed an algorithm that achieves an efficiency of roughly 50% for an number of processors in the range of 2 to 25. All the nodes that still need to be explored are maintained in a list called 'open' list. This list is ordered with respect to how the nodes have been reached. More precisely, the algorithm maintains two lists called 'open' and 'closed', and a tree called 'cut'. The 'open' list contains all the nodes yet to be explored, the 'closed' list contains the expanded nodes not yet pruned and the 'cut' tree contains the pruned nodes. The 'open' list initially contains only the root node. All processors fetch nodes from the 'open' list and process them if they cannot be discarded, that is, they do not have any of their ancestors in the 'cut' tree. Leave nodes are evaluated and their result is returned to the parent which may update its value and check for possible pruning by traversing the 'cut' tree up to the root node applying the usual alpha and beta cutoffs. If the node selected is not a leave node, it is expanded and its sons are inserted into the 'open' list and itself into the 'closed' list.

S.G. Akl et al. [1,2] proposed an algorithm that uses the same approach for exploring the minimax tree. Their priority function is computed as

$$p(n_i) = p(\text{father}(n_i)) - (b_{n_i} + 1 - i) \cdot 10^{(h-f-1)},$$

where n_i is the ith son of node father(n_i), b_{n_i} the branching of node father(n_i), h the search depth (the maximal depth of the minimax tree) and f the depth of node father(n_i) in the minimax tree.

K. Almquist et al. [3] also developed an algorithm based on the idea of having two categories of unexplored nodes which are ordered according to a given priority function. Furthermore they add to this concept parallel aspiration search as well as a novel scheduling algorithm.

In the same direction, V.-D. Cung and C. Roucairol [9] have proposed a shared memory parallel minimax algorithm which distinguishes between critical and non critical nodes. In their algorithm one processor is assigned to each node.

In the algorithm by I.R. Steinberg and M. Solomon [28], which is also a mandatory work first type algorithm, the list containing the speculative work or non critical nodes is dynamically ordered.

Aspiration Search

The parallel algorithm called *aspiration search* has been introduced by Baudet in 1978 [7]. In this algorithm the search interval $]-\infty, +\infty[$ used by the sequential alpha-beta algorithm is divided into a certain number of subintervals that cover the entire range $]-\infty, +\infty[$. Now, every processor explores the entire minimax tree using one subinterval, different processors being assigned different intervals. Any processor searching an interval $]a_i, a_{i+1}]$ may either fail low or high. The principle is the same as in the sequential version of the algorithm. Exactly one processor will neither fail low, nor fail high. The value computed by this processor is the value of the minimax tree to explore.

The implementation of the aspiration search algorithm is really simple. Furthermore, there is no information exchange needed between processors. If the nodes in the to explore minimax tree are ordered in such a way that the alpha-beta algorithm has to explore the whole tree, then the speedup obtained by using the aspiration search algorithm is maximal. But, when the aspiration search algorithm is applied to randomly generated trees then Baudet has shown that the speedup is limited to about six and is independent of the number of processors used.

Tree-Splitting Algorithm

Among the early parallel minimax algorithms is the *tree-splitting algorithm* by R.A. Finkel and J.P. Fishburn [14]. This algorithm is based on the idea to look at the available processors as a tree of processors. Each processor, except for the ones representing leaves in the processor tree, have a fixed number p_b of son or slave processors. During the execution of the algorithm a non leave processor associated with a node n in the minimax tree spawns the exploration of the sons s_i of n to its p_b slaves. As soon as one slave returns the next unexplored son s_j is spawned to that slave or the current value is returned to the father processor if the cut condition is satisfied. If all the sons of a node have been spawned to its slaves, the father processor waits for the results of all its slaves. Leave processors simply compute the value of their associated node using the sequential alpha-beta algorithm.

An important advantage of the tree-splitting algorithm over other more elaborated algorithms is that it may be simply implemented as well on a shared memory parallel machine as on a distributed memories parallel machine.

The tree-splitting algorithm has been implemented and its execution has been simulated. On a 27 processor simulated machine, in which each processor has tree slave sons associated, the average speedup was 5.31 for trees of depth eight and a branching of three.

PVSPLIT: Principal Variation Splitting Algorithm

It has been proposed by T.A. Marsland and M.S. Campbell [19] and is by far the most often implemented algorithm, especially in chess playing programs. The algorithm is based on the structure of the sequential alpha-beta algorithm. The idea is to first explore in a sequential fashion a path from the root node to its leftmost leave. This path is called the *principal variation path*. The traversal is done to obtain alpha and beta bounds. If the minimax tree to explore is of type best first, then the explored principal variation path represents the solution path. In a second phase, for each level of the minimax tree all the yet to be visited sons are explored in parallel by using the bounds computed during the principal variation path computation and the traversal of the lower levels of the minimax tree.

The PVSPLIT algorithm is completely described by the following pseudocode using the negamax notation.

The PVSPLIT algorithm has been implemented in [20] on a network of Sun workstations. An acceleration of 3.06 has been measured on 4 processors when traversing minimax trees representing real chess games. The main problem of the PVSPLIT algorithm is that, during the second phase, the subtrees explored in parallel are not necessarily of the same size.

The PVSPLIT algorithm is most efficient when the iterative deepening technique is used, because with each iteration is is increasingly likely that the first move tried, that is, the one on the principal variation path, is the best one.

```
FUNCTION PVSplit($b, \alpha, \beta$) IS
BEGIN
  IF is_leave($n$) THEN RETURN $f(n)$
  $s \leftarrow$ first_son($n$)
  $\alpha \leftarrow -$PVSplit($s, -\beta, -\alpha$)
  IF $\alpha \geq \beta$ THEN RETURN $\alpha$
  FOR $s' \in$ sons($n$) $- \{s\}$ LOOP IN PARALLEL
    ⟨wait until a slave node is idle⟩
    $v_i \leftarrow -$TreeSplit($s', -\beta, -\alpha$)
    IF $v_i > \alpha$ THEN
      $\alpha \leftarrow v_i$
      ⟨Update the bounds according to $\alpha$ on all
      slaves⟩
    END IF
    IF $\alpha > \beta$ THEN
      ⟨Terminate all slave processors⟩
      RETURN $\alpha$
    END IF
  END LOOP
  RETURN $\alpha$
END PVSplit
```

Pseudocode for the PVSPLIT algorithm

Synchronized Distributed State Space Search

A completely different approach to parallelizing the SSS∗ algorithm has been taken by C.G. Diderich and

M. Gengler [10]. The algorithm proposed is called synchronized distributed state space search (SDSSS). It is an alternation of computation and synchronization phases. The algorithm has been designed for a distributed memory multiprocessor machine. Each processor manages its own local 'open' list of unvisited nodes.

The synchronization phase may be subdivided in three major parts. First, the processors exchange information about which nodes can be removed from the local 'open' lists. This corresponds to each processor sending the nodes for which the 'purge' operation may be applied by all the other processors. Next, all the processors agree on the globally lowest upper bound m^* for which nodes exist in some of the 'open' lists. Finally all the nodes having the same upper bound m^* are evenly distributed among all the processors. This operation concludes the synchronization phase.

The computation phase of the SDSSS algorithm may be described by the following pseudocode.

```
⟨Computation phase⟩ ≡
    WHILE ⟨there exists a node in the open list
    having an upper bound of $m^*$⟩
        LOOP
        $(s, t, m^*) \leftarrow$ remove(open)
        IF $s$ = root AND $t$ = solved THEN
            BROADCAST 'the solution has been
            found'
            RETURN $m^*$
        END IF
        ⟨Apply the $\Gamma$ operator to node $s$⟩
    END LOOP
```

Pseudocode for the computation phase of the SDSSS algorithm

Experiments executing the SDSSS algorithm on an Intel iPSC/2 parallel machine have been conducted. Speedups of up to 11.4 have been measured for 32 processors.

Distributed Game Tree Search Algorithm

R. Feldman [12] parallelized the alpha-beta algorithm for massively parallel distributed memory machines. Different subtrees are searched in parallel by different processors. The allocation of processors to trees is done by imposing certain conditions on the nodes which are be selectable. They introduce the concept of *younger brother waits*. This concept essentially says that in the case of a subtree rooted at s_1, where s_1 is the first son node of a node n, is not yet evaluated, then the other sons $s_2, \ldots, s_b$ of node n are not selectable. Younger brothers may only be considered after their elder brothers, which has as a consequence that the value of the elder brothers may be used to give a tight search window to the younger brothers.

This concept is nevertheless not sufficient to achieve the same good search window as the alpha-beta algorithm achieves. Indeed when node s_1 is computed, then the younger brothers may all be explored in parallel using the value of node s_1. Thus the node s_2 has the same search window as it would have in the sequential alpha-beta algorithm, but this is not true anymore for s_i, where $i \geq 3$. Indeed if nodes s_2 and s_3 are processed in parallel, they only know the value of node s_1, while in the sequential alpha-beta algorithm, the node s_3 would have known the value of both s_1 and s_2. This fact forces the parallel algorithm to provide an information dissemination protocol.

In case the nodes s_2 and s_3 are evaluated on processors P and P', and processor P finishes its work before P', producing a better value than node s_1 did, then processor P will inform processor P' of this value, allowing it to continue with better information on the rest of its subtree or to terminate its work if the new value allows P' to conclude that its computation becomes useless. The load distribution is realized by means of a dynamic load balancing scheme, where idle processors ask other processors for work.

Speedups as high as 100 have been obtained on a 256 processor machines. In [13], a speedup of 344 on a 1024 transputer network interconnected as a grid and a speedup of 142 on a 256 processor transputer de Bruijn interconnected network have been shown.

Parallel Minimax Algorithm with Linear Speedup

In 1988, Althöfer [4] proved that it is possible, to develop a parallel minimax algorithm which achieves linear speedup in the average case. With the assumption that all minimax trees are binary win-loss trees, he exhibited such a parallel minimax algorithm.

M. Böhm and E. Speckenmeyer [8] also suggested an algorithm which uses the same basic ideas as Althöffer. Their algorithm is more general in the sense that it needs only to know the distribution of the leave values and is independent of the branching of the tree explored.

In 1989, R.M. Karp and Y. Zhang [17] proved that it is possible to obtain linear speedup on every instance of a random uniform minimax tree if the number of processors is close to the height of the tree.

See also

► Bottleneck Steiner Tree Problems
► Directed Tree Networks
► Shortest Path Tree Algorithms

References

1. Akl SG, Barnard DT, Doran RJ (1979) Searching game trees in parallel. In: Proc. 3rd Biennial Conf. Canad. Soc. Computation Studies of Intelligence, pp 224–231
2. Akl SG, Barnard DT, Doran RJ (1982) Design, analysis, and implementation of a parallel tree search algorithm. IEEE Trans Pattern Anal Machine Intell PAMI-4(2):192–203
3. Almquist K, McKenzie N, Sloan K (1988) An inquiry into parallel algorithms for searching game trees. Techn. Report Univ. Washington, Seattle, WA 12(3)
4. Althöfer I (1988) On the complexity of searching game trees and other recursion trees. J Algorithms 9:538–567
5. Althöfer I (1990) An incremental negamax algorithm. Artif Intell 43:57–65
6. Ballard BW (1983) The $*$ -minimax search procedure for trees containing chance nodes. Artif Intell 21:327–350
7. Baudet GM (1978) The design and analysis of algorithms for asynchronous multiprocessors. PhD Thesis Carnegie-Mellon Univ. Pittsburgh, PA, CMU-CS-78-116
8. Böhm M, Speckenmeyer E (1989) A dynamic processor tree for solving game trees in parallel. Proc. SOR'89
9. Cung V-D, Roucairol C (1991) Parallel minimax tree searching. Res Report INRIA, vol 1549
10. Diderich CG (1992) Evaluation des performances de l'algorithme SSS$*$ avec phases de synchronisation sur une machine parallèle à mémoires distribuées. Techn. Report Computer Sci. Dept. Swiss Federal Inst. Techn. Lausanne, Switzerland, LiTH-99 (In French.)
11. Feigenbaum EA, Feldman J (1963) Computers and thought. McGraw-Hill, New York
12. Feldmann R, Monien B, Mysliwietz P, Vornberger O (1989) Distributed game tree search. ICCA J 12(2):65–73
13. Feldmann R, Mysliwietz P, Monien B (1994) Game tree search on a massively parallel system. In: van den Herik HJ, Herschberg IS, Uiterwijk JWHM (eds) Advances in Computer Chess, vol 7. Univ. Limburg, Maastricht, pp 203–218
14. Finkel RA, Fishburn JP (1982) Parallelism in alpha-beta search. Artif Intell 19:89–106
15. Hewett R, Krishnamurthy G (1992) Consistent linear speedup in parallel alpha-beta search. Proc. ICCI'92, Computing and Information. IEEE Computer Soc Press, New York, pp 237–240
16. Ibaraki T (1986) Generalization of alpha-beta and {SSS*} search procedures. Artif Intell 29:73–117
17. Karp RM, Zhang Y (1989) On parallel evaluation of game trees. In: ACM Annual Symp. Parallel Algorithms and Architectures (SPAA'89). ACM, New York, pp 409–420
18. Knuth DE, Moore RW (1975) An analysis of alpha-beta pruning. Artif Intell, 6(4):293–326
19. Marsland TA, Campbell MS (1982) Parallel search of strongly ordered game trees. ACM Computing Surveys 14(4):533–551
20. Marsland TA, Popowich F (1985) Parallel game-tree search. IEEE Trans Pattern Anal Machine Intell PAMI-7(4):442–452
21. Marsland TA, Reinefeld A, Schaeffer J (1987) Low overhead alternatives to SSS*. Artif Intell 31:185–199
22. McAllester DA (1988) Conspiracy numbers for min-max searching. Artif Intell 35:287–310
23. Pearl J (1980) Asymptotical properties of minimax trees and game searching procedures. Artif Intell 14(2):113–138
24. Pijls W, de Bruin A (Aug. 1990) Another view of the SSS$*$ algorithm. In: Proc. Internat. Symp. (SIGAL'90)
25. Rivest RL (1987) Game tree searching by min/max approximation. Artif Intell 34(1):77–96
26. Roizen I, Pearl J (1983) A minimax algorithm better than alpha-beta? Yes and no. Artif Intell 21:199–230
27. Slagle JH, Dixon JK (Apr. 1969) Experiments with some programs that search game trees. J ACM 16(2):189–207
28. Steinberg IR, Solomon M (1990) Searching game trees in parallel. Proc. IEEE Internat. Conf. Parallel Processing, III, III–9–III–17
29. Stockman GC (1979) A minimax algorithm better than alpha-beta? Artif Intell 12(2):179–196

Minimax Theorems

STEPHEN SIMONS
Department Math., University California, Santa Barbara, USA

MSC2000: 46A22, 49J35, 49J40, 54D05, 54H25, 55M20, 91A05

Article Outline

Keywords

Minimax theorem; Fixed point theorem; Hahn–Banach theorem; Connectedness

We suppose that X and Y are nonempty sets and $f\colon X \times Y \to \mathbf{R}$. A *minimax theorem* is a theorem that asserts that, under certain conditions,

$$\inf_Y \sup_X f = \sup_X \inf_Y f\,,$$

that is to say,

$$\inf_{y \in Y} \sup_{x \in X} f(x, y) = \sup_{x \in X} \inf_{y \in Y} f(x, y).$$

The purpose of this article is to give the reader the flavor of the different kind of minimax theorems, and of the techniques that have been used to prove them. This is a very large area, and it would be impossible to touch on all the work that has been done in it in the space that we have at our disposal. The choice that we have made is to give the historical roots of the subject, and then go directly to the most recent results. The reader who is interested in a more complete narrative can refer to the 1974 survey article [35] by E.B. Yanovskaya, the 1981 survey article [8] by A. Irle and the 1995 survey article [31] by S. Simons.

Von Neumann's Results

In his investigation of *games of strategy*, J. von Neumann realized that, even though a two-person zero-sum game did not necessarily have a solution in *pure* strategies, it did have to have one in *mixed* strategies. Here is a statement of that seminal result ([19], translated into English in [21]):

Theorem 1 *(1928) Let A be an $m \times n$ matrix, and X and Y be the sets of nonnegative row and column vectors with unit sum. Then*

$$\min_{y \in Y} \max_{x \in X} xAy = \max_{x \in X} \min_{y \in Y} xAy.$$

Despite the fact that the statement of this result is quite elementary, the proof was quite sophisticated, and depended on an extremely ingenious induction argument. Nine years later, in [20], von Neumann showed that the bilinear character of Theorem 1 was not needed when he extended it as follows, using *Brouwer's fixed point theorem*:

Theorem 2 *(1937) Let X and Y be nonempty compact, convex subsets of Euclidean spaces, and $f\colon X \times Y \to \mathbf{R}$ be jointly continuous. Suppose that f is quasiconcave on X and quasiconvex on Y (see below). Then*

$$\min_Y \max_X f = \max_X \min_Y f\,.$$

When we say that f is *quasiconcave* on X, we mean that
- for all $y \in Y$ and $\lambda \in \mathbf{R}$, $GT(\lambda, y)$ is convex,

and when we say that f is *quasiconvex* on Y, we mean that
- for all $x \in X$ and $\lambda \in \mathbf{R}$, $LE(x, \lambda)$ is convex.

Here, $GT(\lambda, y)$ and $LE(x, \lambda)$ are 'level sets' associated with the function f. Specifically,

$$GT(\lambda, y) := \{x \in X\colon\ f(x, y) > \lambda\}$$

and

$$LE(x, \lambda) := \{y \in Y\colon\ f(x, y) \le \lambda\}\,.$$

In 1941, S. Kakutani [10] analyzed von Neumann's proof and, as a result, discovered the fixed point theorem that bears his name.

Infinite-Dimensional Results for Convex Sets

The first infinite-dimensional minimax theorem was proved in 1952 by K. Fan ([1]), who generalized Theorem 2 to the case when X and Y are compact, convex subsets of infinite-dimensional locally convex spaces, and the quasiconcave and quasiconvex conditions are somewhat relaxed. The result in this general line that has the simplest statement is that of M. Sion, who proved the following ([33]):

Theorem 3 *(1958) Let X be a convex subset of a linear topological space, Y be a compact convex subset of a linear topological space, and $f: X \times Y \to \boldsymbol{R}$ be upper semicontinuous on X and lower semicontinuous on Y. Suppose that f is quasiconcave on X and quasiconvex on Y. Then*

$$\min_Y \sup_X f = \sup_X \min_Y f .$$

When we say that f is 'upper semicontinuous on X' and 'lower semicontinuous on Y' we mean that, for all $y \in Y$, the map $x \longmapsto f(x, y)$ is *upper semicontinuous* and, for all $x \in X$, the map $y \longmapsto f(x, y)$ is *lower semicontinuous*. The importance of Sion's weakening of continuity to semicontinuity was that it indicated that many kinds of minimax problems have equivalent formulations in terms of subsets of X and Y, and led to Fan's 1972 work ([4]) on sets with convex sections and minimax inequalities, which has since found many applications in economic theory. Like Theorem 2, all these result relied ultimately on Brouwer's fixed point theorem (or the related *Knaster–Kuratowski–Mazurkiewicz lemma* (KKM lemma) on closed subsets of a finite-dimensional simplex).

Functional-Analytic Minimax Theorems

The first person to take minimax theorems out of the context of convex subsets of vector spaces, and their proofs (other than that of the matrix case discussed in Theorem 1) out of the context of fixed point theorems was Fan in 1953 ([2]). We present here a generalization of Fan's result due to H. König ([15]). König's proof depended on the Mazur–Orlicz version of the Hahn–Banach theorem (see Theorem 5 below).

Theorem 4 *(1968) Let X be a nonempty set and Y be a nonempty compact topological space. Let $f: X \times Y \to \boldsymbol{R}$ be lower semicontinuous on Y. Suppose that:*

- *for all $x_1, x_2 \in X$, there exists $x_3 \in X$ such that*

$$f(x_3, \cdot) \geq \frac{f(x_1, \cdot) + f(x_2, \cdot)}{2} \quad \text{on } Y;$$

- *for all $y_1, y_2 \in Y$, there exists $y_3 \in Y$ such that*

$$f(\cdot, y_3) \leq \frac{f(\cdot, y_1) + f(\cdot, y_2)}{2} \quad \text{on } X.$$

Then

$$\min_Y \sup_X f = \sup_X \min_Y f .$$

We give here the statement of the *Mazur–Orlicz version of the Hahn–Banach theorem*, since it is a very useful result and it not as well-known as it deserves to be.

Theorem 5 (*Mazur–Orlicz theorem*) *Let S be a sublinear functional on a real vector space E, and C be a nonempty convex subset of E. Then there exists a linear functional L on E such that $L \leq S$ on E and $\inf_C L = \inf_C S$.*

See [16,22] and [23] for applications of the Mazur–Orlicz theorem and the related 'sandwich theorem' to measure theory, Hardy algebra theory and the theory of flows in infinite networks.

The kind of minimax theorem discussed in this section (where X is not topologized) has turned out to be extremely useful in functional analysis, in particular in *convex analysis* and also in the theory of *monotone operators on a Banach space*. (See [32] for more details of these kinds of applications.)

Minimax Theorems that Depend on Connectedness

It was believed for some time that proofs of minimax theorems required either the fixed point machinery of algebraic topology, or the functional-analytic machinery of convexity. However, in 1959, W.-T. Wu proved the first minimax theorem in which the conditions of convexity were totally replaced by conditions related to *connectedness*. This line of research was continued by H. Tuy, L.L. Stachó, M.A. Geraghty with B.-L. Lin, and J. Kindler with R. Trost, whose results were all subsumed by a family of general topological minimax theorem established by König in [17]. Here is a typical result from [17]. In order to simplify the statements of this and some of our later results, we shall write $f_* := \sup_X \inf_Y f$. f_* is the 'lower value' of f. If $\lambda \in \mathbf{R}$, $V \subset Y$ and $W \subset X$, we write $GT(\lambda, V) := \bigcap_{y \in V} GT(\lambda, y)$ and $LE(W, \lambda) := \bigcap_{x \in W} LE(x, \lambda)$.

Theorem 6 (1992) *Let X be a connected topological space, Y be a compact topological space, and $f: X \times Y \to \boldsymbol{R}$ be upper semicontinuous on X and lower semicontinuous on Y. Let Λ be a nonempty subset of (f_*, ∞)*

such that inf $\Lambda = f_*$ and suppose that, for all $\lambda \in \Lambda$, for all nonempty subsets V of Y, and for all nonempty finite subsets W of X,

$$GT(\lambda, V) \quad \textit{is connected in } X,$$

and

$$LE(W, \lambda) \quad \textit{is connected in } Y.$$

Then

$$\min_Y \sup_X f = \sup_X \min_Y f.$$

Mixed Minimax Theorems

In [34], F. Terkelsen proved the first *mixed minimax theorem*. We describe Terkelsen's result as 'mixed' since one of the conditions in it is taken from Theorem 4, and the other from Theorem 6:

Theorem 7 (1972) *Let X be a nonempty set and Y be a nonempty compact topological space. Let $f: X \times Y \to$ **R** be lower semicontinuous on Y. Suppose that,*

- *for all $x_1, x_2 \in X$ there exists $x_3 \in X$ such that*

$$f(x_3, \cdot) \geq \frac{f(x_1, \cdot) + f(x_2, \cdot)}{2} \quad \textit{on } Y.$$

*Suppose also that, for all $\lambda \in$ **R** and, for all nonempty finite subsets W of X,*

$$LE(W, \lambda) \textit{ is connected in } Y.$$

Then

$$\min_Y \sup_X f = \sup_X \min_Y f.$$

A Metaminimax Theorem

It was believed for some time that Brouwer's fixed point theorem or the Knaster–Kuratowski–Mazurkiewicz lemma was required to order to prove Sion's theorem, Theorem 3. However, in 1966, M.A. Ghouila-Houri ([7]) proved Theorem 3 using a simple combinatorial property of convex sets in finite-dimensional space. This was probably the first indication of the breakdown of the classification of minimax theorems as either of 'topological' or 'functional-analytic' type. Further indication of this breakdown was provided by Terkelsen's result, Theorem 7, and the subsequent 1982 results of I. Joó and Stachó ([9]), the 1985 and 1986 results of Geraghty and Lin ([5] and [6]), and the 1989 results of H. Komiya ([18]).

Kindler ([11]) was the first to realize (in 1990) that some abstract concept akin to connectedness might be involved in minimax theorems, even when the topological condition of connectedness was not explicitly assumed. This idea was pursued by Simons with the introduction in 1992 of the concept of *pseudoconnectedness*, which we will now describe. We say that sets H_0 and H_1 are *joined* by a set H if

$$H \subset H_0 \cup H_1, \quad H \cap H_0 \neq \emptyset$$

and

$$H \cap H_1 \neq \emptyset.$$

We say that a family $\mathcal{H}$ of sets is *pseudoconnected* if

$$\begin{aligned} &H_0, H_1, H \in \mathcal{H} \quad \text{and} \quad H_0 \text{ and } H_1 \text{ joined by } H \\ &\Downarrow \\ &H_0 \cap H_1 \neq \emptyset. \end{aligned}$$

Any family of closed connected subsets of a topological space is pseudoconnected. So also is any family of open connected subsets. However, pseudoconnectedness can be defined in the absence of any topological structure and, as we shall see in Theorem 8, is closely related to minimax theorems. Theorem 8 is the improvement of the result of [29] due to König (see [30]). We shall say that a subset W of X is *good* if

- W is finite; and
- for all $x \in X$, $LE(x, f_*) \cap LE(W, f_*) \neq \emptyset$.

Theorem 8 (1995) *Let Y be a topological space, and Λ be a nonempty subset of **R** such that inf $\Lambda = f_*$. Suppose that, for all $\lambda \in \Lambda$ and for all good subsets W of X,*

- *for all $x \in X$, $LE(x, \lambda)$ is closed and compact; $\{LE(x, \lambda) \cap LE(W, \lambda)\}_{x \in X}$ is pseudoconnected; and*
- *for all $x_0, x_1 \in X$, there exists $x \in X$ such that $LE(x_0, \lambda)$ and $LE(x_1, \lambda)$ are joined by $LE(x, \lambda) \cap LE(W, \lambda)$.*

Then

$$\min_Y \sup_X f = \sup_X \min_Y f.$$

Theorem 8 is proved by induction on the cardinality of the good subsets of W. Given the obvious topological motivation behind the concept of pseudoconnectedness, it is hardly surprising that Theorem 8 implies Theorem 6. What is more unexpected is that Theorem 8 implies Theorems 4 and 7 also. We prefer to describe Theorem 8 as a *metaminimax theorem* rather than a *minimax theorem*, since it is frequently harder to prove that the conditions of Theorem 8 are satisfied in any particular case that it is to prove Theorem 8 itself. So Theorem 8 is really a device for obtaining minimax theorems rather than a minimax theorem in its own right.

More recent work by Kindler ([12,13] and [14]) on abstract intersection theorems has been at the interface between minimax theory and abstract set theory.

Minimax Theorems and Weak Compactness

There are close connections between minimax theorems and *weak compactness*. The following 'converse minimax theorem' was proved by Simons in [25]; this result also shows that there are limitations on the extent to which one can totally remove the assumption of compactness from minimax theorems.

Theorem 9 (1971) *Suppose that X is a nonempty bounded, convex, complete subset of a locally convex space E with dual space E^*, and*

$$\inf_{y\in Y}\sup_{x\in X}\langle x, y\rangle = \sup_{x\in X}\inf_{y\in Y}\langle x, y\rangle$$

whenever Y is a nonempty convex, equicontinuous, subset of E^*. Then X is weakly compact.

No compactness is assumed in the following, much harder, result (see [26]):

Theorem 10 (1972) *If X is a nonempty bounded, convex subset of a locally convex space E such that every element of the dual space E^* attains its supremum on X, and Y is any nonempty convex equicontinuous subset of E^*, then*

$$\inf_{y\in Y}\sup_{x\in X}\langle x, y\rangle = \sup_{x\in X}\inf_{y\in Y}\langle x, y\rangle .$$

If one now combines the results of Theorems 9 and 10, one can obtain a proof of the 'sup theorem' of R.C. James, one of the most beautiful results in functional analysis:

Theorem 11 (*James sup theorem*) *If C is a nonempty bounded closed convex subset of E, then C is $w(E, E^*)$-compact if and only if, for all $x^* \in E^*$, there exists $x \in C$ such that $\langle x, x^*\rangle = \max_C x^*$.*

James's theorem is not easy - the standard proof can be found in the paper [24] by J.D. Pryce.

See [31] for more details of the connections between minimax theorems and weak compactness.

Minimax Inequalities for Two or More Functions

Motivated by *Nash equilibrium* and the theory of *noncooperative games*, Fan generalized Theorem 2 to the case of more than one function. In particular, he proved in [3] the following two-function minimax inequality (since the compactness of X is not needed, this result can in fact be strengthened to include Sion's theorem, Theorem 3, by taking $g = f$):

Theorem 12 (1964) *Let X and Y be nonempty compact, convex subsets of topological vector spaces and f, g: $X \times Y \to \boldsymbol{R}$. Suppose that f is lower semicontinuous on Y and quasiconcave on X, g is upper semicontinuous on X and quasiconvex on Y, and*

$$f \le g \quad \text{on } X \times Y .$$

Then

$$\min_Y \sup_X f \le \sup_X \inf_Y g .$$

Fan (unpublished) and Simons (see [27]) generalized König's theorem, Theorem 4, with the following *two-function minimax inequality*:

Theorem 13 (1981) *Let X be a nonempty set, Y be a compact topological space and f, g: $X \times Y \to \boldsymbol{R}$. Suppose that f is lower semicontinuous on Y, and*

- *for all $y_1, y_2 \in Y$ there exists $y_3 \in Y$ such that*

$$f(\cdot, y_3) \le \frac{f(\cdot, y_1) + f(\cdot, y_2)}{2} \quad \text{on } X;$$

- *for all $x_1, x_2 \in X$ there exists $x_3 \in X$ such that*
$$g(x_3, \cdot) \geq \frac{g(x_1, \cdot) + g(x_2, \cdot)}{2} \quad \text{on } Y;$$
and
- *$f \leq g$ on $X \times Y$.*

Then
$$\min_Y \sup_X f \leq \sup_X \inf_Y g.$$

Theorems 12 and 13 both unify the theory of minimax theorems and the theory of *variational inequalities*. The curious feature about these two results is that they have 'opposite geometric pictures'. This question is discussed in [27] and [28]. The relationship between Theorem 12 and Brouwer's fixed point theorem is quite interesting. As we have already pointed out, Sion's theorem, Theorem 3, can be proved in an elementary fashion without recourse to fixed point related concepts. On the other hand, Theorem 12 can, in fact, be used to prove *Tychonoff's fixed point theorem*, which is itself a generalization of Brouwer's fixed point theorem. (See [3] for more details of this.)

A number of authors have proved minimax inequalities for more than two functions. See [31] for more details of these results.

Coincidence Theorems

A *coincidence theorem* is a theorem that asserts that if $S\colon X \to 2^Y$ and $T\colon Y \to 2^X$ have nonempty values and satisfy certain other conditions, then there exist $x_0 \in X$ and $y_0 \in Y$ such that $y_0 \in Sx_0$ and $x_0 \in Ty_0$. The connection with minimax theorems is as follows: Suppose that $\inf_Y \sup_X f \neq \sup_X \inf_Y f$. Then there exists $\lambda \in \mathbf{R}$ such that
$$\sup_X \inf_Y f < \lambda < \inf_Y \sup_X f.$$
Hence,

- for all $x \in X$ there exists $y \in Y$ such that $f(x, y) < \lambda$; and
- for all $y \in Y$ there exists $x \in X$ such that $f(x, y) > \lambda$.

Define $S\colon X \to 2^Y$ and $T\colon Y \to 2^X$ by
$$Sx := \{y \in Y\colon\ f(x, y) < \lambda\} \neq \emptyset$$
and
$$Tx := \{x \in X\colon\ f(x, y) > \lambda\} \neq \emptyset.$$

If S and T were to satisfy a coincidence theorem, then we would have $x_0 \in X$ and $y_0 \in Y$ such that
$$f(x_0, y_0) < \lambda \quad \text{and} \quad f(x_0, y_0) > \lambda,$$
which is clearly impossible. Thus this coincidence theorem would imply that
$$\inf_Y \sup_X f = \sup_X \inf_Y f.$$

The coincidence theorems known in algebraic topology consequently give rise to corresponding minimax theorems. There is a very extensive literature about coincidence theorems. See [31] for more details about this.

See also

- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Minimax: Directional Differentiability
- ▶ Nondifferentiable Optimization: Minimax Problems
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Quasigradient Methods in Minimax Problems

References

1. Fan K (1952) Fixed-point and minimax theorems in locally convex topological linear spaces. Proc Nat Acad Sci USA 38:121–126
2. Fan K (1953) Minimax theorems. Proc Nat Acad Sci USA 39:42–47
3. Fan K (1964) Sur un théorème minimax. CR Acad Sci Paris 259:3925–3928
4. Fan K (1972) A minimax inequality and its applications. In: Shisha O (ed) Inequalities, vol III. Acad. Press, New York, pp 103–113
5. Geraghty MA, Lin B-L (1985) Minimax theorems without linear structure. Linear Multilinear Algebra 17:171–180
6. Geraghty MA, Lin B-L (1986) Minimax theorems without convexity. Contemp Math 52:102–108
7. Ghouila-Houri MA (1966) Le théorème minimax de Sion. In: Theory of games. English Univ. Press, London, pp 123–129
8. Irle A (1981) Minimax theorems in convex situations. In: Moeschlin O, Pallaschke D (eds) Game Theory and Mathematical Economics. North-Holland, Amsterdam, pp 321–331
9. Joó I, Stachó LL (1982) A note on Ky Fan's minimax theorem. Acta Math Acad Sci Hung 39:401–407
10. Kakutani S (1941) A generalization of Brouwer's fixed-point theorem. Duke Math J 8:457–459

11. Kindler J (1990) On a minimax theorem of Terkelsen's. Arch Math 55:573–583
12. Kindler J (1993) Intersection theorems and minimax theorems based on connectedness. J Math Anal Appl 178:529–546
13. Kindler J (1994) Intersecting sets in midset spaces. I. Arch Math 62:49–57
14. Kindler J (1994) Intersecting sets in midset spaces. II. Arch Math 62:168–176
15. König H (1968) Über das Von Neumannsche Minimax-Theorem. Arch Math 19:482–487
16. König H (1970) On certain applications of the Hahn-Banach and minimax theorems. Arch Math 21:583–591
17. König H (1992) A general minimax theorem based on connectedness. Arch Math 59:55–64
18. Komiya H (1989) On minimax theorems. Bull Inst Math Acad Sinica 17:171–178
19. Neumann Jvon (1928) Zur Theorie der Gesellschaftspiele. MATH-A 100:295–320
20. Neumann Jvon (1937) Ueber ein ökonomisches Gleichungssystem und eine Verallgemeinerung des Brouwerschen Fixpunktsatzes. Ergebn Math Kolloq Wien 8:73–83
21. Neumann Jvon (1959) On the theory of games of strategy. In: Tucker AW, Luce RD (eds) Contributions to the Theory of Games, vol 4, Princeton Univ. Press, Princeton, pp 13–42
22. Neumann M (1989) Some unexpected applications of the sandwich theorem. In: Proc. Conf. Optimization and Convex Analysis, Univ. Mississippi
23. Neumann M (1991) Generalized convexity and the Mazur–Orlicz theorem. In: Proc. Orlicz Memorial Conf., Univ. Mississippi
24. Pryce JD (1966) Weak compactness in locally convex spaces. Proc Amer Math Soc 17:148–155
25. Simons S (1970/1) Critères de faible compacité en termes du théorème de minimax. Sém. Choquet 23:8
26. Simons S (1972) Maximinimax: minimax, and antiminimax theorems and a result of R.C. James. Pacific J Math 40:709–718
27. Simons S (1981) Minimax and variational inequalities: Are they or fixed point or Hahn–Banach type? In: Moeschlin O, Pallaschke D (eds) Game Theory and Mathematical Economics. North-Holland, Amsterdam, pp 379–388
28. Simons S (1986) Two-function minimax theorems and variational inequalities for functions on compact and noncompact sets with some comments on fixed-points theorems. Proc Symp Pure Math 45:377–392
29. Simons S (1994) A flexible minimax theorem. Acta Math Hungarica 63:119–132
30. Simons S (1995) Addendum to: A flexible minimax theorem. Acta Math Hungarica 69:359–360
31. Simons S (1995) Minimax theorems and their proofs. In: Du DZ, Pardalos PM (eds) Minimax and Applications. Kluwer, Dordrecht, pp 1–23
32. Simons S (1998) Minimax and monotonicity. Lecture Notes Math, vol 1693. Springer, Berlin
33. Sion M (1958) On general minimax theorems. Pacific J Math 8:171–176
34. Terkelsen F (1972) Some minimax theorems. Math Scand 31:405–413
35. Yanovskaya EB (1974) Infinite zero-sum two-person games. J Soviet Math 2:520–541

Minimum Concave Transportation Problems
MCTP

Bruce W. Lamar
Economic and Decision Analysis Center, The MITRE Corp., Bedford, USA

MSC2000: 90C26, 90C35, 90B06, 90B10

Article Outline

Keywords

Flows in networks; Global optimization; Nonconvex programming; Fixed charge transportation problem

The minimum concave transportation problem MCTP concerns the least cost method of carrying flow on a bipartite network in which the marginal cost for an arc is a nonincreasing function of the flow on that arc. A *bipartite network* contains source nodes and sink nodes, but no transshipment (i. e., intermediate) nodes. The MCTP can be formulated as

$$\min \sum_{(i,j)\in A} \phi_{ij}(x_{ij}) \tag{1}$$

subject to:

$$\sum_{j\in N} x_{ij} = s_i, \quad \forall i \in M, \tag{2}$$

$$\sum_{i\in M} x_{ij} = d_j, \quad \forall j \in N, \tag{3}$$

$$x_{ij} \geq 0, \quad \forall (i,j) \in A, \tag{4}$$

where M is the set of source nodes; N is the set of sink nodes; s_i is the supply at source node i, d_j is the demand at sink node j; $A = \{(i, j) : i \in M, j \in N\}$ is the (directed) arc set; x_{ij} is the flow carried on arc (i, j); and $\phi_{ij}(x_{ij})$ is the concave cost function for arc (i, j). Objective function (1) minimizes total costs; constraints (2) balance flow at the source nodes; and constraints (3) balance flow at the sink nodes. If $\sum_{i \in M} s_i$ is less (greater) than $\sum_{j \in N} d_j$, then a dummy source (sink) node can be added to set M (N).

MCTPs arise naturally in distribution problems involving shipments sent directly from supply points to demand points in which the transportation costs exhibit economies of scale [21]. However, the MCTP is not limited to this class of problems. Specifically, any network flow problem with arc cost functions that are not concave can be converted to a network flow problem on an expanded network whose arc cost functions are all concave [16]. Then, the expanded network can be converted to a bipartite network by replacing each transshipment node with a source node and a sink node. Arc flow capacities can be removed by adding additional source nodes, one for each capacitated arc [19,23].

The fixed charge transportation problem FCTP is a type of MCTP in which the cost function $\phi_{ij}(x_{ij})$ for each arc $(i, j) \in A$ is of the form

$$\phi_{ij}(x_{ij}) = \begin{cases} 0 & \text{if } x_{ij} = 0, \\ f_{ij} + g_{ij} \cdot x_{ij} & \text{if } x_{ij} > 0, \end{cases} \tag{5}$$

where f_{ij} and g_{ij} are coefficients with $f_{ij} \geq 0$. FCTPs are commonly used to model network flow problems involving setup costs [9]. Furthermore, a variety of combinatorial problems can be converted to FCTPs. For instance, consider the 0–1 knapsack problem KP. The KP is formulated as

$$\max \sum_{k=1}^{n} c_k \cdot y_k \tag{6}$$

subject to:

$$\sum_{k=1}^{n} a_k \cdot y_k \leq b, \tag{7}$$

$$y_k \in \{0, 1\}, \quad \text{for } k = 1, \dots, n, \tag{8}$$

with $a_k \geq 0$ and $c_k \geq 0$ for $k = 1, \dots, n$. The KP can be converted to a FCTP with two source nodes and $n + 1$ sink nodes. Define $a_{n+1} = b$ and $c_{n+1} = 0$. Then, the network is specified as $M = \{1, 2\}$, $N = \{1, \dots, n + 1\}$, $s_1 = b$, $s_2 = \sum_{k=1}^{n} a_k$, and $d_j = a_j$ for $j = 1, \dots, n + 1$; and the cost function is of the form of (5) where, for each arc $(i, j) \in A$, the coefficients f_{ij} and g_{ij} are given by

$$f_{ij} = \begin{cases} \sum_{k=1}^{n} c_k & \text{if } j = 1, \dots, n, \\ 0 & \text{if } j = n + 1, \end{cases} \tag{9}$$

$$g_{ij} = \begin{cases} -\frac{c_j}{a_j} & \text{if } i = 1, \\ 0 & \text{if } i = 2. \end{cases} \tag{10}$$

For $j = 1, \dots, n$ sink node j has two incoming arcs, exactly one of which will have nonzero flow in the optimal solution to the FCTP. If $x^*_{1j} > 0$ in the FCTP, then $y^*_j = 1$ in the KP. If $x^*_{2j} > 0$ in the FCTP, then $y^*_j = 0$ in the KP.

One consequence of this result is that *any* integer programming problem with integer coefficients can (in principle) be formulated and solved as a FCTP by first converting the integer program to a KP [10].

Exact solution methods for the MCTP are predominately branch and bound enumeration procedures [2,3,4,6,8,11,12,15]. Binary partitioning is used for the FCTP; and interval partitioning is used for the MCTP with arbitrary concave arc cost functions. Finite convergence of the method was shown by R.M. Soland [22]. The convex envelope of the cost function $\phi_{ij}(x_{ij})$ is an affine function. Hence, a subproblem in the branch and bound procedure can be solved efficiently as a linear transportation problem (LTP) [1]. Fathoming techniques (such as 'up and down penalties' and 'capacity improvement') based on post-optimality analysis of the LTP facilitate the branch and bound procedure for the MCTP [2,3,18,20]. The LTP is also used in approximate solution methods for the MCTP which rely on successive linearizations of the concave cost function, $\phi_{ij}(x_{ij})$ [5,13,14].

Test problems for the MCTP are given in [7,8,12,17,20].

See also

- ► Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ► Concave Programming
- ► Motzkin Transposition Theorem

► Multi-index Transportation Problems
► Stochastic Transportation and Location Problems

References

1. Balinski ML (1961) Fixed-cost transportation problems. Naval Res Logist 8:41–54
2. Barr RS, Glover F, Klingman D (1981) A new optimization method for large scale fixed charge transportation problems. Oper Res 29:448–463
3. Bell GB, Lamar BW (1997) Solution methods for nonconvex network problems. In: Pardalos PM Hearn DW, Hager WW (eds) Network Optimization. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 32–50
4. Cabot AV, Erenguc SS (1984) Some branch-and-bound procedures for fixed-cost transportation problems. Naval Res Logist 31:145–154
5. Diaby M (1991) Successive linear approximation procedure for generalized fixed-charge transportation problems. J Oper Res Soc 42:991–1001
6. Florian M, Robillard P (1971) An implicit enumeration algorithm for the concave cost network flow problem. Managem Sci 18:184–193
7. Floudas CA, Pardalos PM (1990) A collection of test problems for constrained global optimization Algorithms. Lecture Notes Computer Sci, vol 455. Springer, Berlin
8. Gray P (1971) Exact solution of the fixed-charge transportation problem. Oper Res 19:1529–1538
9. Guisewite GM, Pardalos PM (1990) Minimum concave-cost network flow problems: Applications, complexity, and algorithms. Ann Oper Res 25:75–100
10. Kendall KE, Zoints S (1977) Solving integer programming problems by aggregating constraints. Oper Res 25:346–351
11. Kennington J (1976) The fixed-charge transportation problem: A computational study with a branch-and-bound code. AIIE Trans 8:241–247
12. Kennington J, Unger VE (1976) A new branch-and-bound algorithm for the fixed charge transportation problem. Managem Sci 22:1116–1126
13. Khang DB, Fujiwara O (1991) Approximate solutions of capacitated fixed-charge minimum cost network flow problems. Networks 21:689–704
14. Kim D, Pardalos PM (1999) A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. Oper Res Lett 24:195–203
15. Lamar BW (1993) An improved branch and bound algorithm for minimum concave cost network flow problems. J Global Optim 3:261–287
16. Lamar BW (1993) A method for solving network flow problems with general nonlinear arc costs. In: Du D-Z and Pardalos PM (eds) Network Optimization Problems: Algorithms, Applications, and Complexity. World Sci., Singapore, pp 147–167
17. Lamar BW, Wallace CA (1996) A comparison of conditional penalties for the fixed charge transportation problem. Techn. Report Dept. Management Univ. Canterbury
18. Lamar BW, Wallace CA (1997) Revised-modified penalties for fixed charge transportation problems. Managem Sci 43:1431–1436
19. Lawler EL (1976) Combinatorial optimization: Networks and matroids. Holt, Rinehart and Winston, New York
20. Palekar US, Karwan MH, Zionts S (1990) A branch-and-bound method for the fixed charge transportation problem. Managem Sci 36:1092–1105
21. Rech P, Barton LG (1970) A non-convex transportation algorithm. In: Beale EML (ed) Applications of Mathematical Programming Techniques. English Univ. Press, London
22. Soland RM (1974) Optimal facility location with concave costs. Oper Res 22:373–382
23. Wagner HM (1959) On a class of capacitated transportation problems. Managem Sci 5:304–318

Minimum Cost Flow Problem

RAVINDRA K. AHUJA[1], THOMAS L. MAGNANTI[2], JAMES B. ORLIN[3]

[1] Department Industrial and Systems Engineering, University Florida, Gainesville, USA
[2] Sloan School of Management and Department Electrical Engineering and Computer Sci., Massachusetts Institute Technol., Cambridge, USA
[3] Sloan School of Management, Massachusetts Institute Technol., Cambridge, USA

MSC2000: 90C35

Article Outline

Keywords

Network; Minimum cost flow problem;
Cycle-canceling algorithm; Successive shortest path algorithm; Network simplex algorithm

The *minimum cost flow problem* seeks a least cost shipment of a commodity through a network to satisfy demands at certain nodes by available supplies at other nodes. This problem has many, varied applications: the distribution of a product from manufacturing plants to warehouses, or from warehouses to retailers; the flow of raw material and intermediate goods through various machining stations in a production line; the routing of automobiles through an urban street network; and the routing of calls through the telephone system. The minimum cost flow problem also has many less direct applications. In this article, we briefly introduce the theory, algorithms and applications of the minimum cost flow problem. [1] contains much additional material on this topic.

Let $G = (N, A)$ be a *directed network* defined by a set N of n *nodes* and a set A of m *directed arcs*. Each arc $(i, j) \in A$ has an associated *cost* c_{ij} that denotes the cost per unit flow on that arc. We assume that the flow cost varies linearly with the amount of flow. Each arc $(i, j) \in A$ has an associated *capacity* u_{ij} denoting the maximum amount that can flow on this arc, and a lower bound l_{ij} that denotes the minimum amount that must flow on the arc. We assume that the capacity and flow lower bound for each arc (i, j) are integers. We associate with each node $i \in N$ an integer $b(i)$ representing its supply/demand. If $b(i) > 0$, node i is a *supply node*; if $b(i) < 0$, then node i is a *demand node* with a demand of $-b(i)$; and if $b(i) = 0$, then node i is a *transshipment node*. We assume that $\sum_{i \in N} b(i) = 0$. The decision variables x_{ij} are arc flows defined for each arc $(i, j) \in A$.

The minimum cost flow problem is an optimization model formulated as follows:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1}$$

subject to

$$\sum_{\{j:\ (i,j)\in A\}} x_{ij} - \sum_{\{j:\ (j,i)\in A\}} x_{ji} = b(i), \quad \text{for all } i \in N, \tag{2}$$

$$l_{ij} \le x_{ij} \le u_{ij}, \quad \text{for all } (i, j) \in A. \tag{3}$$

We refer to the constraints (2) as the *mass balance constraints*. For a fixed node i, the first term in the constraint (2) represents the total outflow of node i and the second term represents the total inflow of node i. The mass balance constraints state that outflow minus inflow must equal the supply/demand of each node. The flow must also satisfy the lower bound and capacity constraints (3), which we refer to as *flow bound constraints*.

This article is organized as follows. To help in understanding the applicability of the minimum cost flow problem, we begin in Section 2 by describing several applications. In Section 3, we present preliminary material needed in the subsequent sections. We next discuss algorithms for the minimum cost flow problem, describing the cycle-canceling algorithm in Section 4 and the successive shortest path algorithm in Section 5. The cycle-canceling algorithm identifies negative cost cycles in the network and augments flows along them. The successive shortest path algorithm augments flow along shortest cost augmenting paths from the supply nodes to the demand nodes. In Section 6, we describe the network simplex algorithm.

Applications

Minimum cost flow problems arise in almost all industries, including agriculture, communications, defense, education, energy, health care, manufacturing, medicine, retailing, and transportation. Indeed, minimum cost flow problems are pervasive in practice. In this section, by considering a few selected applications that arise in distribution systems planning, capacity planning, and vehicle routing, we give a passing glimpse of these applications.

Distribution Problems

A large class of network flow problems center around distribution applications. One core model is often described in terms of shipments from plants to warehouses (or, alternatively, from warehouses to retailers). Suppose a firm has p plants with known supplies and q warehouses with known demands. It wishes to identify a flow that satisfies the demands at the warehouses from the available supplies at the plants and that minimizes its shipping costs. This problem is a well-known special case of the minimum cost flow problem, known as the *transportation problem*. We next describe in more detail a slight generalization of this model that also incorporates manufacturing costs at the plants.

A car manufacturer has several manufacturing plants and produces several car models at each plant that it then ships to geographically dispersed retail centers throughout the country. Each retail center requests a specific number of cars of each model. The firm must determine the production plan of each model at each plant and a shipping pattern that satisfies the demand of each retail center while minimizing the overall cost of production and transportation.

We describe this formulation through an example. Figure 1 illustrates a situation with two manufacturing plants, two retailers, and three car models. This model has four types of nodes:

i) *plant nodes*, representing various plants;
ii) *plant/model nodes*, corresponding to each model made at a plant;
iii) *retailer/model nodes*, corresponding to the models required by each retailer; and
iv) *retailer nodes* corresponding to each retailer.

The network contains three types of arcs:

i) production arcs;
ii) transportation arcs; and
iii) demand arcs.

The production arcs connect a plant node to a plant/model node; the cost of this arc is the cost of producing the model at that plant. We might place lower and upper bounds on production arcs to control for the minimum and maximum production of each particular car model at the plants. Transportation arcs connect plant/model nodes to retailer/model nodes; the cost of any such arc is the total cost of shipping one car from the manufacturing plant to the retail center. The transportation arcs might have lower or upper bounds imposed upon their flows to model contractual agreements with shippers or capacities imposed upon any distribution channel. Finally, demand arcs connect retailer/model nodes to the retailer nodes. These arcs have zero costs and positive lower bounds that equal the demand of that model at that retail center.

The production and shipping schedules for the automobile company correspond in a one-to-one fashion with the feasible flows in this network model. Consequently, a minimum cost flow provides an optimal production and shipping schedule.

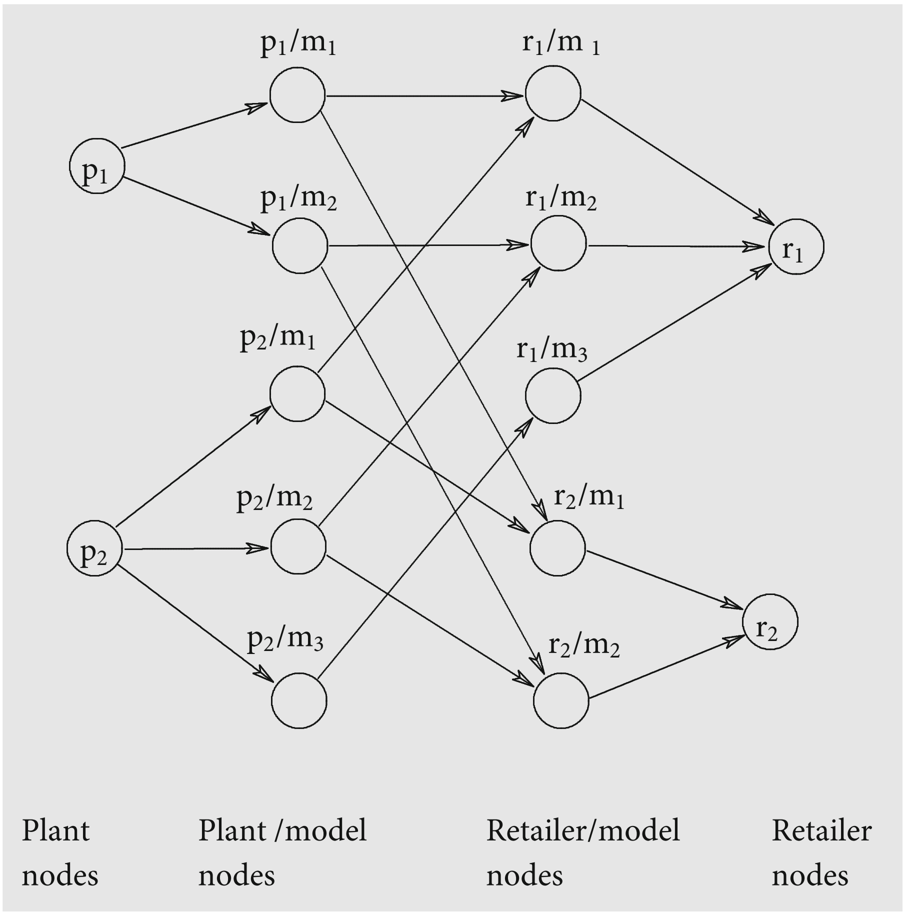
Airplane Hopping Problem

A small commuter airline uses a plane, with a capacity to carry at most p passengers, on a 'hopping flight' as shown in Fig. 2a). The hopping flight visits the cities $1, \ldots, n$, in a fixed sequence. The plane can pick up passengers at any node and drop them off at any other node. Let b_{ij} denote the number of passengers available at node i who want to go to node j, and let f_{ij} denote the fare per passenger from node i to node j. The airline would like to determine the number of passengers that the plane should carry between the various origins to destinations in order to maximize the total fare per trip while never exceeding the plane's capacity.

Figure 2b) shows a minimum cost flow formulation of this hopping plane flight problem. The network contains data for only those arcs with nonzero costs and with finite capacities: any arc listed without an associated cost has a zero cost; any arc listed without an associated capacity has an infinite capacity. Consider, for example, node 1. Three types of passengers are available at node 1: those whose destination is node 2, node 3 or node 4. We represent these three types of passengers in a new derived network by the nodes $1-2$, $1-3$ and $1-4$ with supplies b_{12}, b_{13} and b_{14}. A passenger available at any such node, say $1-3$, could board the plane at its origin node represented by flowing through the arc $(1-3, 1)$ and incurring a cost of $-f_{13}$ units (or profit of f_{13} units). Or, the passenger might never board the plane, which we represent by the flow through the arc $(1-3, 3)$. It is easy to establish a one-to-one correspondence between feasible flows in Fig. 2b) and feasible loading of the plane with passengers. Consequently, a minimum cost flow in Fig. 2b) will prescribe a most profitable loading of the plane.

Directed Chinese Postman Problem

The *directed Chinese postman problem* is a generic routing problem that can be stated as follows. In a directed network $G = (N, A)$ in which each arc (i, j) has an associated cost c_{ij}, we wish to identify a walk of minimum cost that starts at some node (the post office), visits each arc of the network at least once, and returns to the starting point (see the next Section for the def-



Minimum Cost Flow Problem, Figure 1
Formulating the production-distribution problem

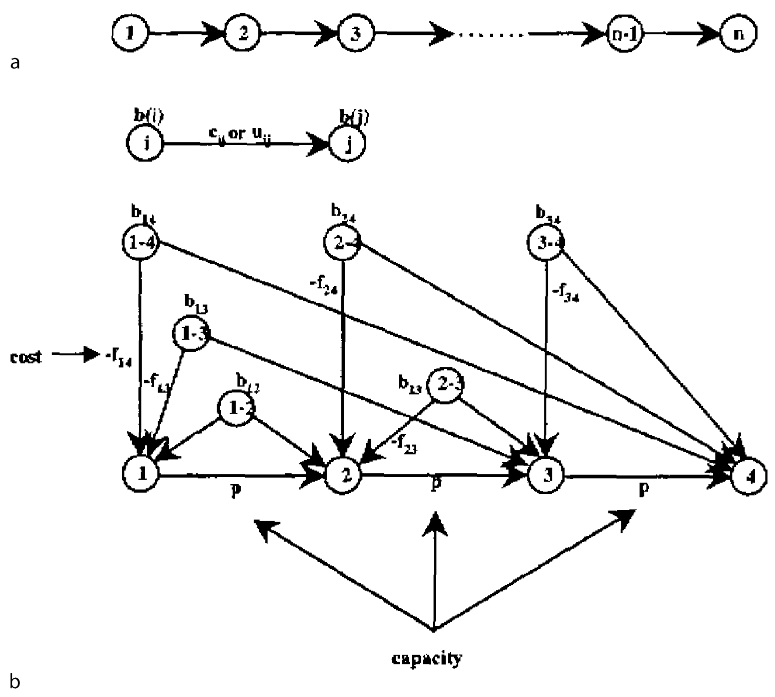
inition of a walk). This problem has become known as the Chinese postman problem because a Chinese mathematician, K. Mei-Ko, first discussed it. The Chinese postman problem arises in other settings as well; for instance, patrolling streets by police, routing street sweepers and household refuse collection vehicles, fuel oil delivery to households, and spraying roads with sand during snowstorms. The directed Chinese postman problem assumes that all arcs are directed, that is, the postal carrier can traverse an arc in only one direction (like one-way streets).

In the directed Chinese postman problem, we are interested in a closed (directed) walk that traverses each arc of the network at least once. The network might not contain any such walk. It is easy to show that a network contains a desired walk if and only if the network is *strongly connected*, that is, every node in the network is reachable from every other node via a directed path. Simple graph search algorithms are able to determine whether the network is strongly connected, and we shall therefore assume that the network is strongly connected.

In an optimal walk, a postal carrier might traverse arcs more than once. The minimum length walk minimizes the sum of lengths of the repeated arcs. Let x_{ij} denote the number of times the postal carrier traverses arc (i, j) in a walk. Any carrier walk must satisfy the following conditions:

$$\sum_{\{j:\ (i,j)\in A\}} x_{ij} - \sum_{\{j:\ (j,i)\in A\}} x_{ji} = 0 \quad \text{for all } i \in N, \tag{4}$$

$$x_{ij} \geq 1 \quad \text{for all } (i, j) \in A. \tag{5}$$



Minimum Cost Flow Problem, Figure 2
Formulation of the hopping plane flight problem as a minimum cost flow problem

The constraints (4) state that the carrier enters a node the same number of times that he or she leaves it. The constraints (5) state that the carrier must visit each arc at least once. Any solution x satisfying the system (4)–(5) defines a carrier's walk. We can construct a walk in the following manner. Given a flow x_{ij}, we replace each arc (i, j) with x_{ij} copies of the arc, each arc carrying a unit flow. In the resulting network, say $G' = (N, A')$, each node has the same number of outgoing arcs as it has the incoming arcs. It is possible to decompose this network into at most $m/2$ arc-disjoint directed cycles (by walking along an arc (i, j) from some node i with $x_{ij} > 0$, leaving an node each time we enter it until we repeat a node). We can connect these cycles together to form a closed walk of the carrier.

The preceding discussion shows that the solution x defined by a feasible walk for the carrier satisfies conditions (4)–(5), and, conversely, every feasible solution of system (4)–(5) defines a walk of the postman. The length of a walk defined by the solution x equals $\sum_{(i,j) \in A} c_{ij} x_{ij}$. This problem is an instance of the minimum cost flow problem.

Preliminaries

In this Section, we discuss some preliminary material required in the following sections.

Assumptions

We consider the minimum cost flow problem subject to the following six assumptions:

1) $l_{ij} = 0$ for each $(i, j) \in A$;
2) all data (cost, supply/demand, and capacity) are integral;
3) all arc costs are nonnegative;
4) for any pair of nodes i and j, the network does not contain both the arcs (i, j) and (j, i);
5) the minimum cost flow problem has a feasible solution; and
6) the network contains a directed path of sufficiently large capacity between every pair of nodes.

It is possible to show that none of these assumptions, except 2), restricts the generality of our development. We impose them just to simply our discussion.

Graph Notation

We use standard graph notation. A directed graph $G = (N, A)$ consists of a set N of nodes and a set A of arcs. A directed arc (i, j) has two *endpoints*, i and j. An arc (i, j) is *incident* to nodes i and j. The arc (i, j) is an *outgoing arc* of node i and an *incoming arc* of node j. A *walk* in a directed graph $G = (N, A)$ is a sequence of nodes and arcs $i_1, a_1, i_2, a_2, \ldots, i_r$ satisfying the property that for all $1 \leq k \leq r - 1$, either $a_k = (i_k, i_{k+1}) \in A$ or $a_k = (i_{k+1}, i_k) \in A$. We sometimes refer to a walk as a *sequence of arcs* (or nodes) without any explicit mention of the nodes (or arcs). A *directed walk* is an oriented version of the walk in the sense that for any two consecutive nodes i_k and i_{k+1} on the walk, $a_k = (i_k, i_{k+1}) \in A$. A *path* is a walk without any repetition of nodes, and a *directed path* is a directed walk without any repetition of nodes. A *cycle* is a path $i_1, i_2, \ldots, i_r$ together with the arc (i_r, i_1) or (i_1, i_r). A directed cycle is a directed path $i_1, i_2, \ldots, i_r$ together with the arc (i_r, i_1). A spanning tree of a directed graph G is a subgraph $G' = (N, A')$ with $A' \subseteq A$ that is connected (that is, contains a path between every pair of nodes) and contains no cycle.

Residual Network

The algorithms described in this article rely on the concept of a *residual network* $G(x)$ corresponding to a flow x. For each arc $(i, j) \in A$, the residual network contains two arcs (i, j) and (j, i). The arc (i, j) has cost c_{ij} and *residual capacity* $r_{ij} = u_{ij} - x_{ij}$, and the arc (j, i) has cost $c_{ji} = - c_{ij}$ and residual capacity $r_{ji} = x_{ij}$. The residual network consists of arcs with positive residual capacity. If $(i, j) \in A$, then sending flow on arc (j, i) in $G(x)$ corresponds to decreasing flow on arc (i, j); for this reason, the cost of arc (j, i) is the negative of the cost of arc (i, j). These conventions show how to determine the residual network $G(x)$ corresponding to any flow x. We can also determine a flow x from the residual network $G(x)$ as follows. If $r_{ij} > 0$, then using the definition of residual capacities and Assumption 4), we set $x_{ij} = u_{ij} - r_{ij}$ if $(i, j) \in A$, and $x_{ji} = r_{ij}$ otherwise. We define the *cost of a directed cycle W* in the residual network $G(x)$ as $\sum_{(i,j) \in W} c_{ij}$.

Order Notation

In our discussion, we will use some well-known notation from the field of complexity theory. We say that an algorithm for a problem $\mathcal{P}$ is an $O(n^3)$ algorithm, or has a *worst-case complexity* of $O(n^3)$, if it is possible to solve any instance of $\mathcal{P}$ using a number of computations that is asymptotically bounded by some constant times the term n^3. We refer to an algorithm as a *polynomial time algorithm* if its worst-case running time is bounded by a polynomial function of the input size parameters, which for a minimum cost flow problem, are n, m, $\log C$ (the number of bits needed to specify the largest arc cost), and $\log U$ (the number of bits needed to specify the largest arc capacity). A *polynomial time algorithm* is either a *strongly polynomial time algorithm* (when the complexity terms involves only n and m, but not $\log C$ or $\log U$), or is a *weakly polynomial time algorithm* (when the complexity terms include $\log C$ or $\log U$ or both). We say that an algorithm is a *pseudopolynomial time algorithm* if its worst-case running time is bounded by a polynomial function of n, m and U. For example, an algorithm with worst-case complexity of $O(nm^2 \log n)$ is a strongly polynomial time algorithm, an algorithm with worst-case complexity $O(nm^2 \log U)$ is a weakly polynomial time algorithm, and an algorithm with worst-case complexity of $O(n^2\, mU)$ is a pseudopolynomial time algorithm.

Cycle-Canceling Algorithm

In this Section, we describe the cycle-canceling algorithm, one of the more popular algorithms for solving the minimum cost flow problem. The algorithm sends flows (called *augmenting flows*) along directed cycles with negative cost (called *negative cycles*). The algorithm rests upon the following negative cycle optimality condition stated as follows.

Theorem 1 (Negative cycle optimality condition) *A feasible solution x^* is an optimal solution of the minimum cost flow problem if and only if the residual network $G(x^*)$ contains no negative cost (directed) cycle.*

It is easy to see the necessity of these conditions. If the residual network $G(x^*)$ contains a negative cycle (that is, a negative cost directed cycle), then by augmenting positive flow along this cycle, we can decrease the cost of the flow. Conversely, it is possible to show that if the residual network $G(x^*)$ does not contain any negative cost cycle, then x^* must be an optimal flow.

The negative cycle optimality condition suggests one simple algorithmic approach for solving the min-

```
BEGIN
  establish a feasible flow $x$ in the network;
  WHILE $G(x)$ contains a negative cycle DO
  BEGIN
    identify a negative cycle $W$;
    $\delta := \min\{r_{ij} : (i,j) \in W\}$;
    augment $\delta$ units of flow in the cycle $W$
    and update $G(x)$;
  END;
END
```

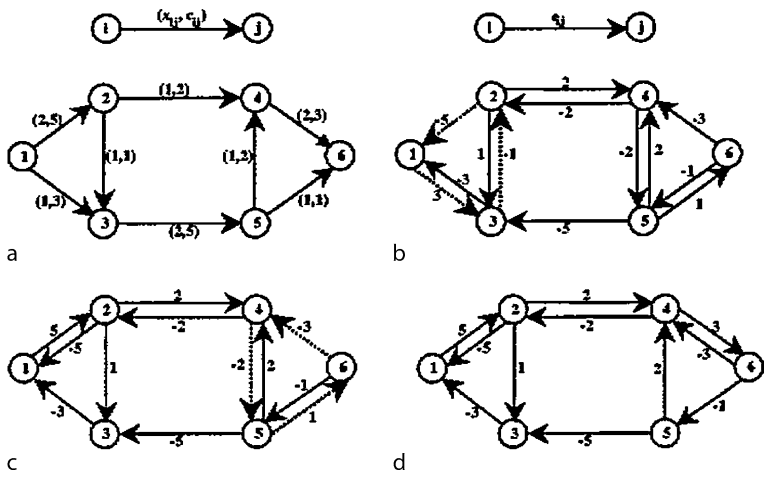
Minimum Cost Flow Problem, Figure 3
Cycle-canceling algorithm

imum cost flow problem, which we call the *cycle-canceling algorithm*. This algorithm maintains a feasible solution and at every iteration improves the objective function value. The algorithm first establishes a feasible flow x in the network by solving a related (and easily solved) problem known as the maximum flow problem. Then it iteratively finds negative cycles in the residual network and augments flows on these cycles. The algorithm terminates when the residual network contains no negative cost directed cycle. Theorem 1 implies that when the algorithm terminates, it has found a minimum cost flow. Figure 3a specifies this generic version of the cycle-canceling algorithm.

The numerical example shown in Fig. 4a) illustrates the cycle-canceling algorithm. This figure shows the arc costs and the starting feasible flow in the network. Each arc in the network has a capacity of 2 units. Figure 4b) shows the residual network corresponding to the initial flow. We do not show the residual capacities of the arcs in Fig. 4b) since they are implicit in the network structure. If the residual network contains both arcs (i, j) and (j, i) for any pair i and j of nodes, then both have residual capacity equal to 1; and if the residual network contains only one arc, then its capacity is 2 (this observation uses the fact that each arc capacity equals 2). The residual network shown in Fig. 4b) contains a negative cycle 1 – 3 – 2 – 1 with cost – 3. By augmenting a unit flow along this cycle, we obtain the residual network shown in Fig. 4c). The residual network shown in Fig. 4c) contains a negative cycle 6 – 4 – 5 – 6 with cost – 4. We augment unit flow along this cycle, producing the residual network shown in Fig. 4d), which contain no negative cycle. Given the optimal residual network, we can determine optimal flow using the method described in the previous Section.

A byproduct of the cycle-canceling algorithm is the following important result.

Theorem 2 (Integrality property) *If all arc capacities and supply/demands of nodes are integer, then the minimum cost flow problem always has an integer minimum cost flow.*



Minimum Cost Flow Problem, Figure 4
Illustration of the cycle-canceling algorithm. a) the original network with flow *x* and arc costs; b) the residual network *G*(*x*); c) the residual network after augmenting a unit of flow along the cycle 2 – 1 – 3 – 2; d) the residual network after augmenting a unit of flow along the cycle 4 – 5 – 6 – 4

This result follows from the fact that for problems with integer arc capacities and integer node supplies/demand, the cycle-canceling algorithm starts with an integer solution (which is provided by the maximum flow algorithm used to obtain the initial feasible flow) and at each iteration augments flow by an integral amount.

What is the worst-case computational requirement (complexity) of the cycle-canceling algorithm? The algorithm must repeatedly identify negative cycles in the residual network. We can identify a negative cycle in the residual network in $O(nm)$ time using a shortest path label-correcting algorithm [1]. How many times must the generic cycle-canceling algorithm perform this computation? For the minimum cost flow problem, mCU is an upper bound on the initial flow cost (since $c_{ij} \leq C$ and $x_{ij} \leq U$ for all $(i, j) \in A$) and $-mCU$ is a lower bound on the optimal flow cost (since $c_{ij} \geq -C$ and $x_{ij} \leq U$ for all $(i, j) \in A$). Any iteration of the cycle-canceling algorithm changes the objective function value by an amount $\sum_{(i,j) \in W} c_{i,j}) \delta$, which is strictly negative. Since we have assumed that the problem has integral data, the algorithm terminates within $O(mCU)$ iterations and runs in $O(nm^2 CU)$ time, which is a pseudopolynomial running time.

The generic version of the cycle-canceling algorithm does not specify the order for selecting negative cycles from the network. Different rules for selecting negative cycles produce different versions of the algorithm, each with different worst-case and theoretical behavior. Two versions of the cycle-canceling algorithm are polynomial time implementations:

i) a version that augments flow in arc-disjoint negative cycles with the maximum improvement [2]; and
ii) a version that augments flow along a negative cycle with minimum mean cost, that is, the average cost per arc in the cycle [4]).

Successive Shortest Path Algorithm

The cycle-canceling algorithm maintains feasibility of the solution at every step and attempts to achieve optimality. In contrast, the successive shortest path algorithm maintains optimality of the solution at every step (that is, the condition that the residual network $G(x)$ contains no negative cost cycle) and strives to attain feasibility. It maintains a solution x, called a pseudoflow (see below), that is nonnegative and satisfies the arcs' flow capacity restrictions, but violates the mass balance constraints of the nodes. At each step, the algorithm selects a node k with excess supply (i. e., supply not yet sent to some demand node), a node l with unfulfilled demand, and sends flow from node k to node l along a shortest path in the residual network. The algorithm terminates when the current solution satisfies all the mass balance constraints.

To be more precise, a *pseudoflow* is a vector x satisfying only the capacity and nonnegativity constraints; it need not satisfy the mass balance constraints. For any pseudoflow x, we define the *imbalance* of node i as

$$e(i) = b(i) + \sum_{\{j,i\} \in A} x_{ji} - \sum_{\{(i,j) \in A\}} x_{ij} \quad \text{for all } i \in N. \quad (6)$$

If $e(i) > 0$ for some node i, then we refer to $e(i)$ as the *excess* of node i; if $e(i) < 0$, then we refer to $-e(i)$ as the node's *deficit*. We refer to a node i with $e(i) = 0$ as *balanced*. Let E and D denote the sets of excess and deficit nodes in the network. Notice that $\sum_{i \in N} e(i) = \sum_{i \in N} b(i) = 0$, which implies that $\sum_{i \in E} e(i) = -\sum_{i \in D} e(i)$. Consequently, if the network contains an excess node, then it must also contain a deficit node. The residual network corresponding to a pseudoflow is defined in the same way that we define the residual network for a flow. The successive shortest path algorithm uses the following result.

Theorem 3 (Shortest augmenting path theorem) *Suppose a pseudoflow (or a flow) x satisfies the optimality conditions and we obtain x′ from x by sending flow along a shortest path from node k to some other node l in the residual network, then x′ also satisfies the optimality conditions.*

To prove this Theorem, we would show that if the residual network $G(x)$ contain no negative cycle, then augmenting flow along any shortest path does not introduce any negative cycle (we will not establish this result in this discussion). Figure 5 gives a formal description of the successive shortest path algorithm.

The numerical example shown in Fig. 6a) illustrates the successive shortest path algorithm. The algorithm starts with $x = 0$, and at this value of flow, the residual network is identical to the starting network. Just as we

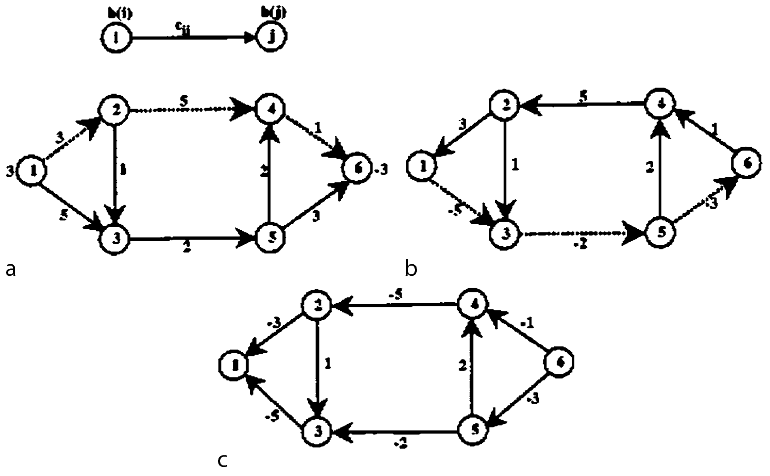
```
BEGIN
    $x := 0$;
    $e(i) = b(i)$ for all $i \in N$;
    initialize the sets $E$ and $D$;
    WHILE $E \neq \emptyset$ DO
    BEGIN
        select a node $k \in E$ and a node $l \in D$;
        identify a shortest path $P$ in $G(x)$ from
        node $k$ to node $l$;
        $\delta := \min[e(s), -e(t), \min\{r_{ij} : (i,j) \in P\}]$;
        augment $\delta$ units of flow along the path $P$ and
        update $x$ and $G(x)$;
    END
END
```

Minimum Cost Flow Problem, Figure 5
Successive shortest path algorithm

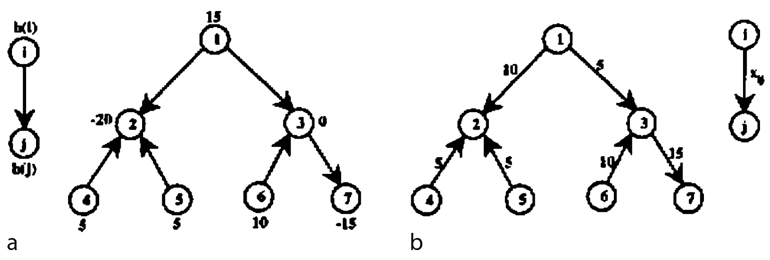
observed in Fig. 4, whenever the residual network contains both the arcs (i, j) and (j, i), the residual capacity of each arc is 1. If the residual network contains only one arc, (i, j) or (j, i), then its residual capacity is 2 units. For this problem, $E = \{1\}$ and $D = \{6\}$. In the residual network shown in Fig. 6a), the shortest path from node 1 to node 6 is 1 – 2 – 4 – 6 with cost equal to 9. The residual capacity of this path equals 2. Augmenting two units of flow along this path produces the residual network shown in Fig. 6b), and the next shortest path from node 1 to node 6 is 1 – 3 – 5 – 6 with cost equal to 10. The residual capacity of this path is 2 and we augment two unit of flow on it. At this point, the sets $E = D = \emptyset$, and the current solution solves the minimum cost flow problem.

To show that the algorithm correctly solves the minimum cost flow problem, we argue as follows. The algorithm starts with a flow $x = 0$ and the residual network $G(x)$ is identical to the original network. Assumption 3) implies that all arc costs are nonnegative. Consequently, the residual network $G(x)$ contains no negative cycle and so the flow vector x satisfies the negative cycle optimality conditions. Since the algorithm augments flow along a shortest path from excess nodes to deficit nodes, Theorem 3 implies that the pseudoflow maintained by the algorithm always satisfies the optimality conditions. Eventually, node excesses and deficits become zero; at this point, the solution maintained by the algorithm is an optimal flow.

What is the worst-case complexity of this algorithm? In each iteration, the algorithm reduces the excess of some node. Consequently, if U is an upper bound on the largest supply of any node, then the algorithm would terminate in at most nU iterations. We can determine a shortest path in $G(x)$ in $O(nm)$ time using a label-correcting shortest path algorithm [1]. Consequently, the running time of the successive shortest path algorithm is $n^2 mU$.



Minimum Cost Flow Problem, Figure 6
Illustration of the successive shortest path algorithm. a) the residual network corresponding to $x = 0$; b) the residual network after augmenting 2 units of flow along the path 1 – 2 – 4 – 6; c) the residual network after augmenting 2 units of flow along the path 1 – 3 – 5 – 6



Minimum Cost Flow Problem, Figure 7
Computing flows for a spanning tree

The successive shortest path algorithm requires pseudopolynomial time to solve the minimum cost flow problem since it is polynomial in n, m and the largest supply U. This algorithm is, however, polynomial time for some special cases of the minimum cost flow problem (such as the assignment problem for which $U = 1$). Researchers have developed weakly polynomial time and strongly polynomial time versions of the successive shortest path algorithm; some notable implementations are due to [3] and [5].

Network Simplex Algorithm

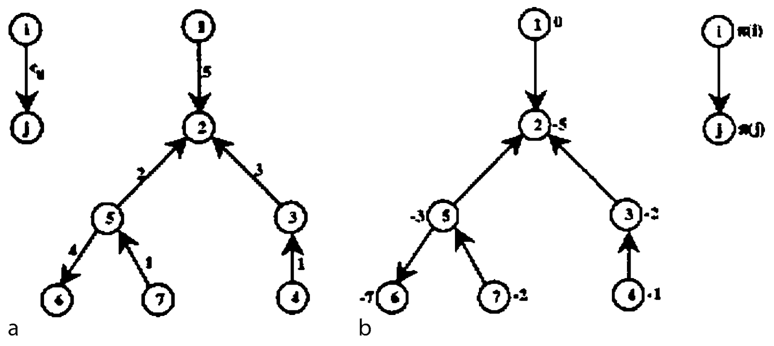
The network simplex algorithm for solving the minimum cost flow problem is an adaptation of the well-known simplex method for general linear programs. Because the minimum cost flow problem is a highly structured linear programming problem, when applied to it, the computations of the simplex method become considerably streamlined. In fact, we need not explicitly maintain the matrix representation (known as the simplex tableau) of the linear program and can perform all of the computations directly on the network. Rather than presenting the network simplex algorithm as a special case of the linear programming simplex method, we will develop it as a special case of the cycle-canceling algorithm described above. The primary advantage of our approach is that it permits the network simplex algorithm to be understood without relying on linear programming theory.

The network simplex algorithm maintains solutions called spanning tree solutions. A *spanning tree solution* partitions the arc set A into three subsets:

i) T, the arcs in the spanning tree;
ii) L, the nontree arcs whose flows are restricted to value zero;
iii) U, the nontree arcs whose flow values are restricted in value to the arcs' flow capacities.

We refer to the triple (T, L, U) as a *spanning tree structure*. Each spanning tree structure (T, L, U) has a unique solution that satisfies the mass balance constraints (2). To determine this solution, we set $x_{ij} = 0$ for all arcs $(i, j) \in L$, $x_{ij} = u_{ij}$ for all arcs $(i, j) \in U$, and then solve the mass balance equations (2) to determine the flow values for arcs in T.

To show that the flows on spanning tree arcs are unique, we use a numerical example. Consider the spanning tree T shown in Fig. 7a). Assume that $U = \varphi$, that is, all nontree arcs are at their lower bounds. Consider the leaf node 4 (a leaf node is a node with exactly one arc incident to it). Node 4 has a supply of 5 units and has only one arc (4, 2) incident to it. Consequently, arc (4, 2) must carry 5 units of flow. So we set $x_{42} = 5$, add 5 units to $b(2)$ (because it receives 5 units of flow sent from node 4), and delete arc (4, 2) from the tree. We now have a tree with one fewer node and next select another leaf node, node 5 with the supply of 5 units and the single arc (5, 2) incident to it. We set $x_{52} = 5$, again add 5 units to $b(2)$, and delete the arc (5, 2) from the tree. Now node 2 becomes a leaf node with modified supply/demand of $b(5) = -10$, implying that node 5 has an unfulfilled demand of 10 units. Node 2 has exactly one incoming arc (1, 2) and to meet the demand of 10 units of node 2, we must send 10 units of flow on this arc. We set $x_{12} = 10$, subtract 10 units from $b(1)$ (since node 1 sends 10 units), and delete the arc (1, 2) from the tree. We repeat this process until we have identified flow on all arcs in the tree. Figure 7b) shows the corresponding flow. Our discussion assumed that U is empty. If U were nonempty, we would first set $x_{ij} = u_{ij}$, add u_{ij} to $b(j)$, and subtract u_{ij} from $b(i)$ for each arc $(i, j) \in U$, and then apply the preceding method.



Minimum Cost Flow Problem, Figure 8
Computing node potentials for a spanning tree

We say a spanning tree structure is *feasible* if its associated spanning tree solution satisfies all of the arcs' flow bounds. We refer to a spanning tree structure as *optimal* if its associated spanning tree solution is an optimal solution of the minimum cost flow problem. We will now derive the optimality conditions for a spanning tree structure (T, L, U).

The network simplex algorithm augments flow along negative cycles. To identify negative cycles quickly, we use the concept of *node potentials*. We define node potentials $\pi(i)$ so that the reduced cost for any arc in the spanning tree T is zero. That is, that is, $c^{\pi}_{ij} = c_{ij} - \pi(i) + \pi(j) = 0$ for each $(i, j) \in T$. With the help of an example, we show how to compute the vector π of node potentials. Consider the spanning tree shown in Fig. 8a) with arc costs as shown. The vector π has n variables and must satisfy $n - 1$ equations, one for each arc in the spanning tree. Therefore, we can assign one potential value arbitrary. We assume that $\pi(1) = 0$. Consider arc (1, 2) incident to node 1. The condition $c^{\pi}_{12} = c_{12} - \pi(1) + \pi(2) = 0$ yields $\pi(2) = -5$. We next consider arcs incident to node 2. Using the condition $c^{\pi}_{52} = c_{52} - \pi(5) + \pi(2) = 0$, we see that $\pi(5) = -3$, and the condition $c^{\pi}_{32} = c_{32} - \pi(3) + \pi(2) = 0$ shows that $\pi(3) = -2$. We repeat this process until we have identified potentials of all nodes in the tree T. Figure 8b) shows the corresponding node potentials.

Consider any nontree arc (k, l). Adding this arc to the tree T creates a unique cycle, which we denote as W_{kl}. We refer to W_{kl} as the *fundamental cycle* induced by the nontree arc (k, l). If $(k, l) \in L$, then we define the orientation of the fundamental cycle as in the direction of (k, l), and if $(k, l) \in U$, then we define the orientation opposite to that of (k, l). In other words, we define the orientation of the cycle in the direction of flow change permitted by the arc (k, l). We let $c(W_{kl})$ denote the change in the cost if we send one unit of flow on the cycle W_{kl} along its orientation. (Notice that because of flow bounds, we might not always be able to send flow along the cycle W_{kl}.) Let $\overline{W}_{kl}$ denote the set of forward arcs in W_{kl} (that is, those with the same orientation as (k, l)), and let $\underline{W}_{kl}$ denote the set of backward arcs in W_{kl} (that is, those with an opposite the orientation to arc (k, l)). Then, if we send one unit of flow along W_{kl}, then the flow on arcs in $\overline{W}_{kl}$ increases by one unit and the flow on arcs in $\underline{W}_{kl}$ decreases by one unit. Therefore,

$$c(W_{kl}) = \sum_{(i,j)\in W_{kl}} c_{ij} - \sum_{(i,j)\in \underline{W}_{kl}} c_{ij} .$$

Let $c^{\pi}(W_{kl})$ denote the change in the reduced costs if we send one unit of flow in the cycle W_{kl} along its orientation, that is,

$$c^{\pi}(W_{kl}) = \sum_{(i,j)\in \overline{W}_{kl}} c^{\pi}_{ij} - \sum_{(i,j)\in \underline{W}_{kl}} c^{\pi}_{ij} .$$

It is easy to show that $c^{\pi}(W_{kl}) = c(W_{kl})$. This result follows from the fact that when we substitute $c^{\pi}_{kl} = c_{ij} - \pi(i) + \pi(j)$ and add the reduced costs around any cycle, then the node potentials $\pi(i)$ cancel one another. Next notice that the manner we defined node potentials ensures that each arc in the fundamental cycle W_{kl} except the arc (k, l) has zero reduced cost. Consequently, if arc $(k, l) \in L$, then

$$c(W_{kl}) = c^{\pi}(W_{kl}) = c^{\pi}_{kl} ,$$

and if arc $(k, l) \in U$, then

$$c(W_{kl}) = c^{\pi}(W_{kl}) = -c^{\pi}_{kl}.$$

This observation and the negative cycle optimality condition (Theorem 1) implies that for a spanning tree solution to be optimal, it must satisfy the following necessary conditions:

$$c^{\pi}_{kl} \geq 0 \quad \text{for every arc } (i, j) \in L, \tag{7}$$

$$c^{\pi}_{kl} \leq 0 \quad \text{for every arc } (i, j) \in U. \tag{8}$$

It is possible to show that these conditions are also sufficient for optimality; that is, if any spanning tree solution satisfies the conditions (7)–(8), then it solves the minimum cost flow problem.

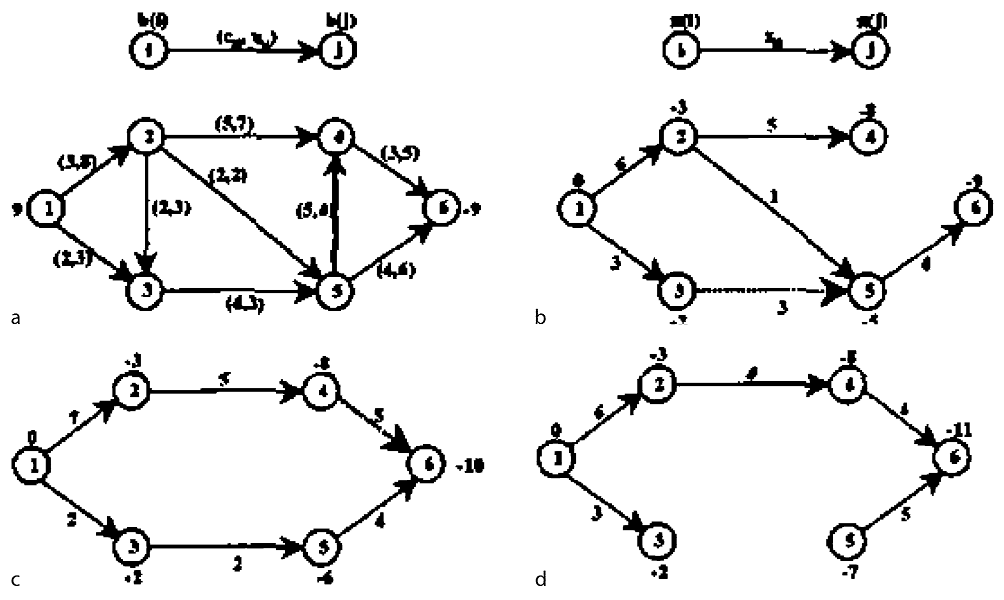
We now have all the necessary ingredients to describe the network simplex algorithm. The algorithm maintains a feasible spanning tree structure at each iteration, which it successively transforms it into an improved spanning tree structure until the solution becomes optimal. The algorithm first obtains an initial spanning tree structure. If an initial spanning tree structure is not easily available, then we could use the following method to construct one: for each node i with $b(i) \geq 0$, we connect node i to node 1 with an (artificial) arc of sufficiently large cost and large capacity; and for each node i with $b(i) < 0$, we connect node 1 to node i with an (artificial) arc of sufficiently large cost and capacity. These arcs define the initial tree T, all arcs in A define the set L, and $U = \emptyset$. Since these artificial arcs have large costs, subsequent iterations will drive the flow on these arcs to zero.

Given a spanning tree structure (T, L, U), we first check whether it satisfies the optimality conditions (7) and (8). If yes, we stop; otherwise, we select an arc $(k, l) \in L$ or $(k, l) \in U$ violating its optimality condition as an *entering arc* to be added to the tree T, obtain the fundamental cycle W_{kl} induced by this arc, and augment the maximum possible flow in the cycle W_{kl} without violating the flow bounds of the tree arcs. At this value of augmentation, the flow on some tree arc, say arc (p, q), reaches its lower or upper bound; we select this arc as an arc to leave the spanning tree T, adding it added to L or U depending upon its flow value. We next add arc (k, l) to T, giving us a new spanning tree structure. We repeat this process until the spanning tree structure satisfies the optimality conditions. Figure 9 specifies the essential steps of the algorithm.

```
BEGIN
    determine an initial feasible tree structure
    $(T, L, U)$;
    let $x$ be the flow and let $\pi$ be the corresponding
    node potentials;
    WHILE (some nontree arc violates its opti-
    mality condition) DO
    BEGIN
        select an entering arc $(k, l)$ violating the opti-
        mality conditions;
        add arc $(k, l)$ to the spanning tree $T$, thus
        forming a unique cycle $W_{kl}$;
        augment the maximum possible flow $\delta$ in the
        cycle $W_{kl}$ and
        identify a leaving arc $(p, q)$ that reaches its
        lower or upper flow bound;
        update the flow $x$, the spanning tree struc-
        ture $(T, L, U)$ and the potentials $\pi$;
    END;
END
```

Minimum Cost Flow Problem, Figure 9
The network simplex algorithm

To illustrate the network simplex algorithm, we use the numerical example shown in Fig. 10a). Figure 10b) shows a feasible spanning tree solution for the problem. For this solution, $T = \{(1, 2), (1, 3), (2, 4), (2, 5), (5, 6)\}$, $L = \{(2, 3), (5, 4)\}$ and $U = \{(3, 5), (4, 6)\}$. We next compute $c^{\pi}_{35} = 1$. We introduce the arc (3, 5) into the tree, creating a cycle. Since (3, 5) is at its upper bound, the orientation of the cycle is opposite to that of (3, 5). The arcs (1, 2) and (2, 5) are forward arcs in the cycle and arcs (3, 5) and (1, 3) are backward arcs. The maximum increase in flow permitted by the arcs (3, 5), (1, 3), (1, 2), and (2, 5) without violating their upper and lower bounds is, respectively, 3, 3, 2, and 1 units. Thus, we augment 1 unit of flow along the cycle. The augmentation increases the flow on arcs (1, 2) and (2, 5) by one unit and decreases the flow on arcs (1, 3) and (3, 5) by one unit. Arc (2, 5) reaches its upper bound and we select it as the leaving arc. We update the spanning tree structure; Fig. 10c) shows the new spanning tree T and the new node potentials. The sets L and U become $L = \{(2, 3), (5, 4)\}$ and $U = \{(2, 5), (4, 6)\}$. In the next iter-



Minimum Cost Flow Problem, Figure 10
Numerical example for the network simplex algorithm

ation, we select arc (4, 6) since this arc violates the arc optimality condition. We augment one unit flow along the cycle 6 – 4 – 2 – 1 – 3 – 5 – 6 and arc (3, 5) leaves the spanning tree. Figure 10d) shows the next spanning tree and the updated node potentials. All nontree arcs satisfy the optimality conditions and the algorithm terminates with an optimal solution of the minimum cost flow problem.

The network simplex algorithm can select any nontree arc that violates its optimality condition as an entering arc. Many different rules, called pivot rules, are possible for choosing the entering arc, and these rules have different empirical and theoretical behavior. [1] describes some popular pivot rules. We call the process of moving from one spanning tree structure to another as a *pivot operation*. By choosing the right data structures for representing the tree T, it is possible to perform a pivot operation in $O(m)$ time.

To determine the number of iterations performed by the network simplex algorithm, we distinguish two cases. We refer to a pivot operation as *nondegenerate* if it augments a positive amount of flow in the cycle W_{kl} (that is, $\delta > 0$), and *degenerate* otherwise (that is, $\delta = 0$). During a degenerate pivot, the cost of the spanning tree solution decreases by $|c^{\pi}_{kl}|\delta$. When combined with the integrality of data assumption (Assumption 2) above), this result yields a pseudopolynomial bound on the number of nondegenerate iterations. However, degenerate pivots do not decrease the cost of flow and so are difficult to bound. There are methods to bound the number of degenerate pivots. Obtaining a polynomial bound on the number of iterations remained an open problem for quite some time; [6] suggested an implementation of the network simplex algorithm that runs in polynomial time. In any event, the empirical performance of the network simplex algorithm is very attractive. Empirically, it is one of the fastest known algorithms for solving the minimum cost flow problem.

See also

- ▶ Auction Algorithms
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Generalized Networks
- ▶ Maximum Flow Problem
- ▶ Multicommodity Flow Problems
- ▶ Network Design Problems

- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Nonoriented Multicommodity Flow Problems
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs, NJ
2. Barahona F, Tardos E (1989) Note of Weintraub's minimum cost circulation algorithm. SIAM J Comput 18:579–583
3. Edmonds J, Karp RM (1972) Theoretical improvements in algorithmic efficiency for network flow problems. J ACM 19:248–264
4. Goldberg AV, Tarjan RE (1988) Finding minimum-cost circulations by canceling negative cycles. Proc. 20th ACM Symposium on the Theory of Computing, pp 388–397. Full paper: J ACM (1989) 36:873–886
5. Orlin JB (1988) A faster strongly polynomial minimum cost flow algorithm. Proc. 20th ACM Symp. Theory of Computing, pp 377–387. Full paper: Oper Res (1989) 41:338–350
6. Orlin JB (1997) A polynomial time primal network simplex algorithm for minimum cost flows. Math Program 78B:109–129

MINLP: Application in Facility Location-allocation

MARIANTHI IERAPETRITOU[1],
CHRISTODOULOS A. FLOUDAS[2]
[1] Department Chemical and Biochemical Engineering, Rutgers University, Piscataway, USA
[2] Department Chemical Engineering, Princeton University, Princeton, USA

MSC2000: 90C26

Article Outline

Keywords

MINLP; Facility location-allocation

The *location-allocation problem* may be stated in the following general way: Given the location or distribution of a set of customers which could be probabilistic and their associated demands for a given product or service, determine the optimal locations for a number of service facilities and the allocation of their products or services to the costumers, so as to minimize total (expected) location and transportation costs. This problem finds a variety of applications involving the location of warehouses, distribution centers, service and production facilities and emergency service facilities. In the last section we are going to consider the development of an offshore oil field as a real-world application of the location-allocation problem. This problem involves the location of the oil platforms and the allocation of the oil wells to platforms.

It was shown in [25] that the joint location-allocation problem is *NP*-hard even with all the demand points located along a straight line. In the next section alternative location-allocation models will be presented based on different objectives and the incorporation of consumer behavior, price elasticity and system dynamics within the location-allocation decision framework.

Location-allocation Models

In developing location-allocation models different objectives alternatives are examined. One possibility is to follow the approach in [5], to minimize the number of centers required to serve the population. This objective is appropriate when the demand is exogenously fixed. A more general objective is to maximize demand by optimally locating the centers as proposed in [10]. The demand maximization requires the incorporation of *price elasticity* representing the dependence of the costumer preference to the distance from the center. The cost of establishing the centers can also be incorporated in the model as proposed in [13]. An alternative objective towards the implementation of costumer preference towards the nearest center is the minimization of an aggregated weighted distance which is called the *median location-allocation* problem.

The simplest type of location-allocation problem is the *Weber problem*, as posed in [9], which involves locating a production center so as to minimize aggregate weighted distance from the different raw material sources. The extension of the Weber problem is the *p-median location-allocation problem*, which involves the optimal location of a set of *p uncapacitated centers* to minimize the total weighted distance between them and *n* demand locations. Here, each source is assumed to have infinite capacity. In continuous space, the *p*-median problem can be formulated as follows:

$$\begin{cases} \min & C = \sum_{i=1}^{n}\sum_{j=1}^{p} O_i \lambda_{ij} c_{ij} \\ \text{s.t.} & \sum_{j=1}^{p} \lambda_{ij} = 1, \quad i = 1,\dots,n, \\ & \lambda_{ij} = 0, 1, \quad i = 1,\dots,n,\ j = 1,\dots,p, \end{cases}$$

where O_i is the quantity demanded at location i whose coordinates are (x_i, y_i); and λ_{ij} is the binary variables that is assigned the value of 1 if demand point i is located to center j and zero otherwise. The above formulation allocate the consumers to their nearest center while ensuring that only one center will serve each customer. This however, can lead to disproportionally sized facilities. In the more realistic situation where the capacities of the facilities are limited to supplies of $s_1, \dots, s_n$ for $i = 1, \dots, n$ facilities then the location-allocation problem takes the following form [24]:

$$\begin{cases} \min & C = \sum_{i=1}^{n}\sum_{j=1}^{p} w_{ij} c_{ij} \\ \text{s.t} & \sum_{j=1}^{p} w_{ij} = s_i, \quad i = 1,\dots,n, \\ & \sum_{i=1}^{n} w_{ij} = d_j, \quad j = 1,\dots,p, \\ & \lambda_{ij} = 0, 1, \quad i = 1,\dots,n,\ j = 1,\dots,p, \end{cases}$$

where w_{ij} is the amount shipped from facility i located at (x_i, y_i) to destination j. In the above formulations the distance (or the generalized transport cost, which is assumed to be proportional to distance) between the demand point i and the supply point j is represented by c_{ij}. The Euclidean metric:

$$c_{ij} = \sqrt{(x_i - \alpha_j)^2 + (y_i - \beta_j)^2}$$

or the rectilinear metric:

$$c_{ij} = \left|x_i - \alpha_j\right| + \left|y_i - \beta_j\right| .$$

The rectilinear metric is appropriate when the transportation is occurring along a grid of city streets (Manhattan norm) or along the aisles of a floor shop [8].

The aforementioned location-allocation models are based on the assumption that the consumers always prefer the nearest center to obtain service. In reality however, as reported in the literature from several empirical studies [11] there exist several services for which consumers choose their service facility center. The travel patterns of the consumers for example can produce a variety of allocations that differ from the nearest center rule. In order to accommodate such behavior a *spatial-interaction model* is incorporated within the uncapacitated *p*-median location-allocation model in the following manner:

$$\begin{cases} \min & \frac{1}{\beta}\sum_j Y_j \sum_i S_{ij}\log(S_{ij} - 1) \\ & \quad + \sum_j \sum_i Y_j S_{ij} c_{ij} \\ \text{s.t.} & \sum_j Y_j S_{ij} = O_i, \quad i = 1,\dots,n, \\ & \sum_j Y_j = p \\ & S_{ij} \le Y_j, \quad i = 1,\dots,n,\ j = 1,\dots,p, \\ & Y_j = 0, 1, \quad j = 1,\dots,p, \end{cases}$$

where the decision variables include Y_j which takes the value of one if the facility is located at J models. and zero otherwise;

$$S_{ij} = A_i O_i Y_j \exp(-\beta c_{ij})$$

that defines the interaction of facility i and consumer j.

$$A_i = \frac{1}{\sum_l Y_l \exp(-\beta c_{il})}, \quad i = 1, \dots, m,$$

that ensures that the sum of all outflows from the origin i add up to the amount of demand at that location; β is either calibrated to match some known interaction data or is defined exogenously. The following relationship holds between the original p-median model and the spatial-interaction model as shown in [17]. The value of the optimal objective function at the solution of the p-median problem is given by:

$$\sum_i \sum_j O_i X_{ij} c_{ij},$$

where X_{ij} allocates demand to the nearest of p available centers. Turning to spatial-interaction model, as the impedance parameter β increases the term:

$$\frac{Y_j \exp(-\beta c_{ij})}{\sum_l Y_l \exp(-\beta c_{il})}$$

of the S_{ij} tends to X_{ij}, where $X_{ij} = 1$ if the travel time from i to j is smaller that the travel time from i to any other facility and zero otherwise. Therefore, the S_{ij} tends to $O_i X_{ij}$ and this model allocates the demand to the nearest facility as the original p-median problem.

All the models mentioned above consider the static location-allocation problem where all the activities take place at one instance. These formulations are sufficient if neither the level nor the location of demand alters over time. An important factor however, in any location-allocation problem is the dynamics of the system involving demand changes over time. Particularly, in the competitive environment, an optimal center location could become undesirable as new competing centers develop. Potential directions include the literature on *decision making under uncertainty*, [12]. A.J. Scott [18] proposed a general framework for the integration of the spatial and discrete temporal dimensions in the location-allocation models. He proposed a modification of the location-allocation so as to minimize an aggregate weighted transport cost over T time periods, during which time the number n_t, level O_{it} and the location (x_{it}, y_{it}) of the demand points change. If the locations were greatly different the center would be likely to relocate at some time and costs of relocation are included in the model. It was assumed that when a center relocates it incurs a fixed cost, α. Based on these ideas the formulation proposed for the uncapacitated location-allocation problem has the following form:

$$\begin{cases} \min & \alpha_1 + \sum_i^{n_1} O_{i1} c_{ij1} \\ & \qquad + \sum_t^T \left(\alpha\lambda_t + \sum_{i=1}^{n_t} O_{it} c_{ijt} \right) \\ \text{s.t.} & \lambda_t = 0, 1, \quad t = 2, \dots, T, \end{cases}$$

where the subscript t refers to different time periods, α_1 is the cost of establishing the center in the first time period. The problem as formulated above is to locate in the first period one center that takes into account future variations. Extending the aspects of this model allows the replacement of a truly dynamic model by a series of static problems as proposed in [3], thus outlining a multilayer approach, where the objective is to sequentially locate each period's facility given the previous period's facility locations in order to minimize the present period cost. This strategy is appropriate whenever the period durations are sufficiently long or under uncertainty regarding future data or decisions. An alternative approach proposed in [24] is a discounted present worth strategy which is appropriate whenever the foregoing conditions do not hold. In this case the facilities are being located one per period and the decisions are made in a rolling horizon framework.

Solution Approaches

For the uncapacitated location-allocation problem using Euclidean metric for the distances between each facility and the different demand points, R.F. Love and H. Juel [15] showed that this problem is equivalent to a concave minimization problem for which they used several heuristic procedures. For the capacitated problems assuming that the costs are proportional to l_p^q using l_p distances where $p \geq 1$ and $q \geq 1$ are integers, M. Avriel [1] developed a geometric programming approach. H.D. Sherali and C.M. Shetty [22] proposed a polar cutting plane algorithm for the case $p = q = 1$. For the case $p = q = 2$, Sherali and C.H. Tuncbilek [23] proposed a branch and bound algorithm (cf. ► MINLP: Branch and bound methods; ► MINLP: Branch and bound global optimization algorithm) that utilizes a specialized tight, linear programming representation to calculate strong upper bounds via a Lagrangian relaxation scheme. They exploit the special structure of the transportation constraints to derive a partitioning scheme. Additional cut-set inequalities are also incorporated to preserve partial solution.

For the uncapacitated location-allocation model using rectilinear distance metric Love and J.G. Morris [16] have developed an exact two-stage algorithm. R.E. Kuenne and R.M. Soland [14], have developed a branch and bound algorithm based on a constructive assignment of customers to sources. The capacitated problem has been addressed in [19,21] and utilize the discrete equivalence of the capacitated location-allocation problem. In particular, [8], and [26] showed that

a) the optimal values of x_i and y_i for each i must satisfy $x_i = \alpha_j$ for some j and $y_i = \beta_j$ for some j, which means that the rectilinear distance location problem always has an optimal solution with the sources located at the grid points of the vertical and horizontal lines drawn through the existing customer locations; and
b) the optimal source locations lie in the convex hull of the existing facility locations.

Based on these ideas and by denoting $k = 1, \dots, K$ the intersection grid points that also belong to the convex hull of the existing facility locations, [21], introduced the decision binary variables z_{ik} that take the value of 1 if source i is located at point k and zero otherwise. This leads to the following discrete location-allocation problem:

$$\begin{cases} \min & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{p}\sum_{k=1}^{K} c_{ijk} w_{ij} z_{ik} \\ \text{s.t.} & \displaystyle\sum_{k=1}^{K} z_{ik} = 1, \quad i = 1, \dots, n, \\ & \displaystyle\sum_{j=1}^{p} w_{ij} = s_i, \quad i = 1, \dots, n, \\ & \displaystyle\sum_{i=1}^{n} w_{ij} = d_j, \quad j = 1, \dots, p, \\ & w_{ij} \geq 0, \quad i = 1, \dots, n, \quad j = 1, \dots, p, \\ & z_{ik} = 0, 1, \quad i = 1, \dots, n, \end{cases}$$

where $c_{ijk} = c_{ij}\ [|\alpha_k - \alpha_j| + |\beta_k - \beta_j|]$. The above model corresponds to a mixed integer bilinear programming problem. See [19] for a related version of this discrete-site location-allocation problem involving one-to-one assignment restriction and fixed charges. See [20] for the solution of the problem as a bilinear programming problem, since the binary variables z can be treated as positive variables because of the problem structure that preserves the binariness of z at optimality. However, in [21] it is proved that it is more useful to exploit the binary nature of z variables for the efficient solution of the above model. Before giving more details of this proposed branch and bound based approach we should mention the heuristic approach proposed in [4], which is very widely used. This so-called *alternating procedure* exploited the fundamental concepts of the location-allocation problem and simply involves allocating demand to centers and relocating centers until some convergence criterion is achieved. For the uncapacitated p-median problem, the alternating procedure involves iterating through the following equations:

$$x_j = \frac{\sum_{i=1}^{n} \frac{O_i \lambda_{ij} x_i}{c_{ij}}}{\sum_{i=1}^{n} \frac{O_i \lambda_{ij}}{c_{ij}}},$$

$$y_j = \frac{\sum_{i=1}^{n} \frac{O_i \lambda_{ij} y_i}{c_{ij}}}{\sum_{i=1}^{n} \frac{O_i \lambda_{ij}}{c_{ij}}},$$

which are derived from differentiating the objective function with respect to x_j and y_j and setting the partial derivatives to zero. The major drawback of this procedure is that it does not guarantee global optimality. This is in fact a concern because the spatial configuration of the local and the global optimum may be very different. As a rule, repeated runs using numerous starting values should be undertaken, although there is no guarantee that the repeatedly found solution would be the global optimum. Note however that the procedure is general to all different models of the location-allocation problem.

Returning to the approach proposed in [21] for the case of rectilinear capacitated location-allocation problem, the following linear reformulation of the problem is used:

$$\begin{cases} \min & \sum_{i=1}^{n}\sum_{j=1}^{p}\sum_{k=1}^{K} c_{ijk}X_{ijk} \\ \text{s.t.} & \sum_{k=1}^{K} X_{ijk} - w_{ij} = 0, \quad \forall(i,j), \\ & \sum_{j=1}^{p} X_{ijk} - s_i z_{ik} = 0, \quad \forall(i,k), \\ & -X_{ijk} + u_{ij} z_{ik} \geq 0, \quad \forall(i,j), \\ & \sum_{k=1}^{K} z_{ik} = 1, \quad \forall i, \\ & \sum_{j=1}^{p} w_{ij} = s_i, \quad \forall i, \\ & \sum_{i=1}^{n} w_{ij} = d_j, \quad \forall j, \\ & w_{ij} \geq 0, \quad \forall(i,j), \\ & z_{ik} = 0,1, \quad \forall i, \\ & X_{ijk} \geq 0, \quad \forall(i,j,k), \end{cases}$$

where $u_{ij} = \min\{s_i, d_j\}$. The above model corresponds to a mixed integer linear programming problem for which a special branch and bound algorithm is applied based on the derivation of tight lower bounds via a suitable Lagrangian dual formulation.

Briefly, for the location-allocation problems that have embedded spatial-interaction equations dual-based exact methods, [17], and heuristic approaches, [2], have been developed.

Application: Development of Offshore Oil Fields

In this section a real world application of the location-allocation problem is presented considering the minimum-cost development of offshore oil fields, [6]. The facilities to be located are the platforms and the demands to be allocated are the oil wells. For the initial information about an oil field, locations are decided upon the production wells which are specified by two map coordinates and a depth coordinate. The drilling is performed directionally from fixed platforms. The cost of drilling depends on the length and angle of the well from the platform. The platform cost depends on the water depth and on the number of wells to be drilled from the platform. Consequently for a large number of wells (25 to 300) an optimization problem that arises is to find the number, size and location of the platforms and the allocation of wells to platforms so as to minimize the sum of platform and drilling costs.

In order to formulate this problem the following indices, parameters and variables are introduced. Let m denote the number of wells and i the index of well, n the number of platforms and j the index for platform, z_{ij} are then the binary variables that represent the allocation of the well i to platform j if it takes the value of 1, otherwise it becomes 0, S_j the capacity of the platform j representing the number of wells drilled from this platform, (a_i, b_i) denote the location coordinates of well i and (x_j, y_j) the location of platform j, $d_{ij} = \sqrt{[(x_j - a_i)^2 + (y_j - b_i)^2]}$ is the horizontal Euclidean distance between well i and platform j, $g(d_{ij})$ denotes the drilling cost function that depends on distance d_{ij}, $P(S_j, x_j, y_j)$ is the platform cost which is a function of platform size S_j and its location. Based on this notation the location-allocation problem can be formulated as follows:

$$\begin{cases} \min & \sum_{i=1}^{m}\sum_{j=1}^{n} z_{ij} g(d_{ij}) + \sum_{j=1}^{n} P(S_j, x_j, y_j) \\ \text{s.t.} & \sum_{j=1}^{n} z_{ij} = 1, \quad \forall(i), \\ & \sum_{i=1}^{m} z_{ij} = S_j, \quad \forall j, \\ & z_{ij} = 0,1, \quad \forall(i,j), \end{cases}$$

where the first set of constraints guarantee that each well is assigned to exactly one platform and the second set guarantee that exactly S_j wells are assigned to each platform. Note that n is fixed in the problem and is usually small in the size of 3 to 5. The nature of the problem depends upon the form of the cost of the drilling function and the platform cost function. The approach taken in [6] is the alternating location-allocation method presented in the previous section. For the specific problem the approach involves the following steps:

a) given fixed platform locations find a minimum cost allocation of wells to platforms;
b) given fixed allocation of wells to platforms find the minimum total cost location for each platform.

The procedure alternates between steps a) and b) until convergence is achieved. The convergence criterion is the following: From the solution of step a) a set of n subproblems are generated for each one of the platforms, the solution of these problems result in the relocation of the platforms. The iterations continue until no changes are possible. As mentioned above, the solution obtained from this algorithmic procedure is locally optimum in the sense that for a given assignment of wells to platforms the solution cannot be improved by changing locations and for given locations, the solution cannot be improved by altering the assignment of wells to platforms. The mathematical formulation of problem a), the *allocation subproblem* is the following:

$$\begin{cases} \min & \sum_{i=1}^{m}\sum_{j=1}^{n} z_{ij} g(d_{ij}) + \sum_{j=1}^{n} P(S_j) \\ \text{s.t.} & \sum_{j=1}^{n} z_{ij} = 1, \quad \forall(i), \\ & \sum_{i=1}^{m} z_{ij} = S_j, \quad \forall j, \\ & z_{ij} = 0, 1, \quad \forall(i, j), \end{cases}$$

note that the platform cost now depends only on S_j since the location of the platforms are known. The solution procedure for this problem depends on the form of the platform cost $P(S_j)$. Five different forms are discussed in [6]:

1) *Single fixed cost with no capacity constraints*: $P(S_j) = a_j$ In this case the total cost for platforms is fixed and the optimal allocation corresponds to the assignment of the wells to the closest platform.
2) *Single fixed cost with capacity constraints*: $P(S_j) = a_j$ and capacity constraints are introduced as inequalities $\sum_{i=1}^{m} z_{ij} \leq S_j$, $\forall j$. In this case the problem corresponds to a linear programming model.
3) *Linear platform cost*: $P(S_j) = a_j + b_j S_j$ By considering the following transformation $c_{ij}' = c_{ij} + b_j$ the problem takes the form of case 1).
4) *Piecewise linear function*. In this case the problem has the structure of 'transshipment problem' which can be solved network flow techniques.
5) *Step function*: $P(S_j) = \sum_{k=1}^{K_j} r_j^k z_{ij}^k$, where K_j are the number of different size platforms available and r_j^k is the cost of kth size of platform j. The problem in this case is a mixed integer linear programming problem.

The mathematical formulation for problem b), the location problem, is the following. Assuming that A_j is the set of indices for the wells assigned to platform j, then $z_{ij} = 1$, for $i \in A_j$, $z_{ij} = 0$ otherwise and the problem for platform j takes the form:

$$\min \sum_{i=1}^{m} \sum_{i \in A_j} g(d_{ij}) + P(x_j, y_j).$$

Note that the platform cost is a function of platform location only since the size is assumed known. Since the drilling cost function is convex, if the platform cost is also convex then the problem corresponds to the minimization of a convex function that can be achieved through a local minimization algorithm. Of course if the platform cost is nonconvex then global optimality cannot be guaranteed and global optimization techniques should be considered, [7].

Finally, M.D. Devine and W.G. Lesso, [6], applied the aforementioned procedure to two test problems one involving 60 wells and 7 platforms and a second one involving 102 wells and 3 platforms. In both cases they reported large economic savings in the field development.

See also

▶ Chemical Process Planning
▶ Combinatorial Optimization Algorithms in Resource Allocation Problems
▶ Competitive Facility Location
▶ Extended Cutting Plane Algorithm
▶ Facility Location with Externalities
▶ Facility Location Problems with Spatial Interaction
▶ Facility Location with Staircase Costs
▶ Generalized Benders Decomposition
▶ Generalized Outer Approximation
▶ Global Optimization in Weber's Problem with Attraction and Repulsion
▶ MINLP: Applications in Blending and Pooling Problems
▶ MINLP: Applications in the Interaction of Design and Control
▶ MINLP: Branch and Bound Global Optimization Algorithm
▶ MINLP: Branch and Bound Methods
▶ MINLP: Design and Scheduling of Batch Processes
▶ MINLP: Generalized Cross Decomposition
▶ MINLP: Global Optimization with αBB
▶ MINLP: Heat Exchanger Network Synthesis

- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming
- ▶ Multifacility and Restricted Location Problems
- ▶ Network Location: Covering Problems
- ▶ Optimizing Facility Location with Rectilinear Distances
- ▶ Production-distribution System Design Problem
- ▶ Resource Allocation for Epidemic Control
- ▶ Single Facility Location: Circle Covering Problem
- ▶ Single Facility Location: Multi-objective Euclidean Distance Location
- ▶ Single Facility Location: Multi-objective Rectilinear Distance Location
- ▶ Stochastic Transportation and Location Problems
- ▶ Voronoi Diagrams in Facility Location
- ▶ Warehouse Location Problem

References

1. Avriel M (1980) A geometric programming approach to the solution of locational problems. J Reg Sci 20:239–246
2. Beaumont JR (1980) Spatial interaction models and the location-allocation problem. J Reg Sci 20:37–50
3. Cavalier TM, Sherali HD (1985) Sequential location-allocation problems on chains and trees with probabilistic link demands. Math Program 32:249–277
4. Cooper L (1964) Heuristic methods for location-allocation problems. SIAM Rev 6:37–53
5. Cristaller W (1966) Central places in southern Germany. Prentice-Hall, Englewood Cliffs, NJ
6. Devine MD, Lesso WG (1972) Models for the minimum cost development of offshore oil fields. Managem Sci 18:378–387
7. Floudas CA (1997) Deterministic global optimization in design, control, and computational chemistry. In: IMA Proc.: Large Scale Optimization with Applications. Part II: Optimal Design and Control 93:129–184
8. Francis RL, White JA (1974) Facility layout and location: An analytical approach. Prentice-Hall, Englewood Cliffs, NJ
9. Friedrich CJ (1929) Alfred Weber's theory of the location of industries. Univ. Chicago Press, Chicago
10. Getis A, Getis J (1966) Cristaller's central place theory. J Geography 65:200–226
11. Hubbard MJ (1978) A review of selected factors conditioning consumer travel behavior. J Consumer Res 5:1–21
12. Ierapetritou MG, Acevedo J, Pistikopoulos EN (1996) An optimization approach for process engineering problems under uncertainty. Comput Chem Eng 20:703–709
13. Koshaka RE (1983) A central-place model as a two-level location-allocation system. Environm Plan 15:5–14
14. Kuenne RE, Soland RM (1972) Exact and approximate solutions to the multisource Weber problem. Math Program 3:193–209
15. Love RF, Juel H (1982) Properties and solution mathods for large location-allocation problems. J Oper Res Soc 33:443–452
16. Love RF, Morris JG (1975) A computational procedure for the exact solution of location-allocation problems with rectangular distances. Naval Res Logist Quart 22:441–453
17. O'Kelly M (1987) Spatial interaction based location-allocation models. In: Ghosh A, Rushton G (eds) Spatial Analysis and Location Allocation Models. v. Nostrand, Princeton, NJ, pp 302–326
18. Scott AJ (1971) Dynamic location-allocation systems: Some basic planning strategies. Environm Plan 3:73–82
19. Sherali AD, Adams WP (1984) A decomposition algorithm for a discrete location-allocation problem. Oper Res 32:878–900
20. Sherali AD, Alameddine AR (1992) A new reformulation-linearization technique for the bilinear programming problems. J Global Optim 2:379–410
21. Sherali AD, Ramachandran S, Kim S (1994) A localization and reformulation discrete programming approach for the rectilinear discrete location-allocation problem. Discrete Appl Math 49:357–378
22. Sherali AD, Shetty CM (1977) The rectilinear distance location-allocation problem. AIIE Trans 9:136–143
23. Sherali AD, Tuncbilek CH (1992) A squared-Euclidean distance location-allocation problem. Naval Res Logist 39:447–469
24. Sherali HD (1991) Capacitated, balanced, sequential location-allocation problems on chain and trees. Math Program 49:381–396
25. Sherali HD, Nordai FL (1988) NP-hard, capacitated, balanced p-median problems on a chain graph with a continuum of link demands. Math Oper Res 13:32–49
26. Wendell RE, Hurter AP (1973) Location theory, dominance and convexity. Oper Res 21:314–320

MINLP: Applications in Blending and Pooling Problems

VISWANATHAN VISWESWARAN
SCA Technologies LLC, Pittsburgh, USA

MSC2000: 90C90, 90C30

Article Outline

Keywords

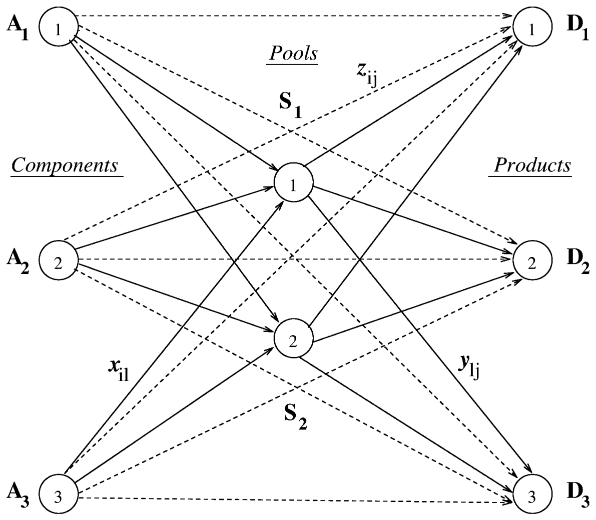
Pooling; Blending; Multiperiod optimization

Pooling and blending is inherent in many manufacturing plants with limited tankage available to store the intermediate streams produced by various processes. Also, chemical products often need to be transported as a mixture, either in a pipeline, a tank car or a tanker. In each case, blended or pooled streams are then used in further downstream processing. In modeling these processes, it is necessary to model not only product flows but the properties of intermediate streams as well. The presence of these pools can introduce nonlinearities and nonconvexities in the model of the process, resulting in difficult problems with multiple local optima.

Given a set of components i, a set of products j, a set of pools k and a set of qualities l, let x_{il} be the amount of component i allocated to pool l, y_{lj} be the amount going from pool l to product j, z_{ij} be the amount of component i going directly to product j and p_{lk} be the level of quality k in pool l. Furthermore, let A_i, D_j and S_l be upper bounds for component availabilities, product demands and pool sizes respectively, let C_{ik} be the level of quality k in component i, P_{jk} be upper bounds on product qualities, c_i be the unit price of component i and d_j be the unit price of product j. The general pooling and blending model can then be written as [1]:

$$\begin{cases} \max & -\sum_{i,l} c_i x_{il} + \sum_{l,j} d_j y_{lj} + \sum_{i,j} (d_j - c_i) z_{ij} \\ \text{s.t.} & \sum_l x_{il} + \sum_j z_{ij} \le A_i \\ & \sum_l y_{lj} + \sum_i z_{ij} \le D_j \\ & \sum_i x_{il} - \sum_j y_{lj} = 0 \\ & \sum_i x_{il} \le S_l \\ & -\sum_i C_{ik} x_{il} + p_{lk} \sum_j y_{lj} = 0 \\ & \sum_l (p_{lk} - P_{jk}) y_{lj} \\ & \quad + \sum_i (C_{ij} - P_{jk}) z_{ij} \le 0 \\ & x_{il}, y_{lj}, z_{ij}, p_{lk} \ge 0. \end{cases}$$

The first two sets of constraints ensure that the amount of components used and products made do not exceed the respective availabilities or demands. The third and fourth set of constraints are material balance constraints around each pool, which ensure that there is no accumulation or overflow of material in the pools. The fifth set of constraints relates the quality of each pool to the quality of the components going into the pool (in this case, the qualities are assumed to blend linearly, that is, the pool quality is an average of the qualities of the components). Finally, the sixth set of equations ensures that any upper bound specifications on product qualities are met. These last two sets of equations are



MINLP: Applications in Blending and Pooling Problems, Figure 1
General pooling and blending problem

bilinear, and can cause significant problems in solving these models.

The general blending problem has a similar formulation as above, except that the pools need not be present; the components can be blended directly to make various products. It should also be noted that there are various other formulations possible, involving multiple time periods, tanks and inventories for components and products, and costs for pooling. Moreover, not all the components need go through all pools. One example of a simplified pooling model, due to C.A. Haverly [8,9], is given in Fig. 2, where three components with varying sulfur contents are to be blended to form two products. There is a maximum sulfur restriction on each product. The components have values of 6, 13 and 10, respectively, while the products have values of 9 and 15, respectively. The mathematical model for the problem consists of writing mass and sulfur balances for the various streams, and can be formulated as

$$\begin{cases} \max & 9\cdot(y_{11}+z_{31})+15\cdot(y_{12}+z_{32}) \\ & -6x_{11}-13x_{21}-10\cdot(z_{31}+z_{32}) \\ \text{s.t.} & x_{11}+x_{21}-y_{11}-y_{12}=0 \\ & p\cdot y_{11}+2z_{31}-2.5(y_{11}+z_{31})\leq 0 \\ & p\cdot y_{12}+2z_{32}-1.5(y_{12}+z_{32})\leq 0 \\ & p\cdot(y_{11}+y_{12})-3x_{11}-x_{21}=0 \\ & y_{11}+z_{31}\leq 100 \\ & y_{12}+z_{32}\leq 200. \end{cases}$$

The variable p represents the sulfur content of the pool (and of y_{11} and y_{12}) and is determined as an average of the sulfur contents of x_{11} and x_{21}.

Characteristics of Pooling and Blending Problems

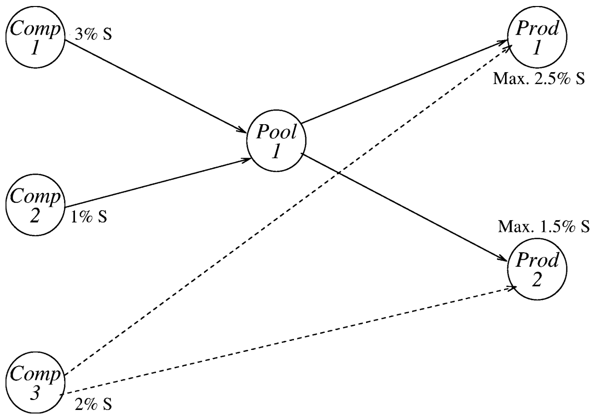
Multiple Solutions

The presence of nonconvex constraints needed to define pool and product qualities often results in multiple local solutions in these models. For example, consider the optimal solution of the Haverly pooling problem as a function of the pool quality p, as shown in Fig. 3.

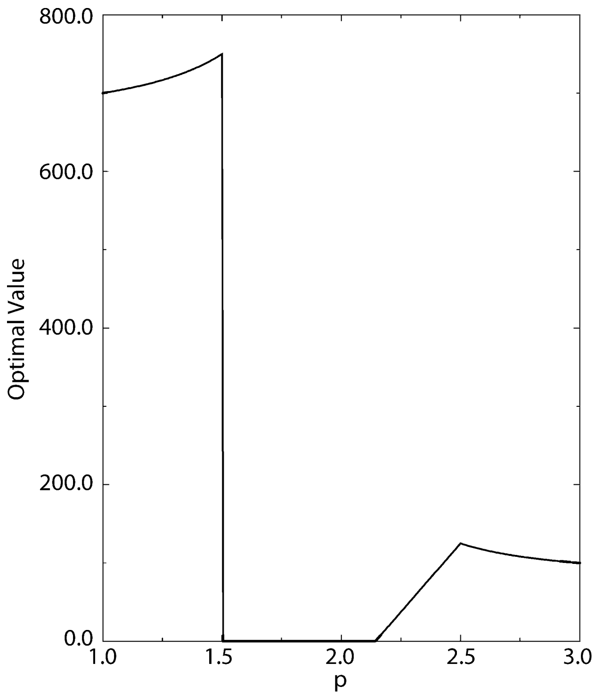
It can be seen that the problem has three solutions:

1) A local maximum of 125 at p = 2.5 with x_{11} = 75, x_{21} = 25, y_{11} = 100 and all other variables zero;
2) a saddle point region with 1 < p < 2, all flows zero and profit of zero; and
3) a global maximum of 750 at p = 1.5 with x_{11} = 50, x_{21} = 150, y_{12} = 200 and all other variables zero.

It is not uncommon for a large pooling problem to have many dozen local optima, with the objective function varying by small amounts but with all the flow and quality variables taking on vastly different values.



MINLP: Applications in Blending and Pooling Problems, Figure 2
Haverly pooling problem



MINLP: Applications in Blending and Pooling Problems, Figure 3
Optimal solution to Haverly pooling problem

Nonlinear Blending

For the sake of simplicity, it is often assumed in formulating these models that the qualities to be tracked blend linearly by volume or weight of each component. In practice, however, this is rarely the case. For example, one of the properties commonly tracked in refinery blends is the *Reid vapor pressure* (RVP), which measures the volatility of a blend. The most commonly used blending rule for RVP is the *Chevron method*:

$$\left(\sum_i x_i\right) R^{1.25} = \sum_i x_i r_i^{1.25},$$

where r_i is the RVP of component i, x_i is its volume, and R is the RVP of the blend. Including such a nonlinear equation in the model can cause difficulty in its solution. Fortunately, this can be avoided by introducing a *blending index*, defined as

$$\overline{r_i} = r_i^{1.25}, \quad \overline{R} = R^{1.25}.$$

Then, all specifications on the blend RVP can be converted using the same index. For example, if there is a lower bound R^L on the blend RVP, then using the blending index results in the constraints as:

$$\left(\sum_i x_i\right) \overline{R} = \sum_i x_i \overline{r_i}, \qquad \overline{R} \geq (R^L)^{1.25}.$$

In some cases, the properties (such as octane number or pour point) can require complex blending rules which cannot be simplified using the blending index, and the full nonlinear blending equation must be included in the model as is.

Single versus Multiperiod Models

Since components are pooled or blended in the plants on a regular basis, it is often advantageous to model these processes using multiple periods. With *multiperiod models*, it is possible to accumulate material in the pools or blend tanks, thereby facilitating the allocation of stocks ahead of time in anticipation of a future lifting of a valuable product. This requires the model to incorporate inventories (carry-over stock) in each tank or pool, resulting in more complex models. It is important to note that each period does not need to be of the same duration. Often, the results of the multiperiod models will only be implemented for the first period, with results for future periods being used for planning purposes. Therefore, initial periods are typically of shorter duration (say a day each) while later periods might be as long as a month. This way, the same multiperiod model can be used as an operating tool for the present and a planning tool for the future.

Another important consideration in multiperiod models is the disposition of stocks at the end of the final period. If the final inventories/stocks are included simply as variables, the optimal solution will almost always set them to zero. In practice, however, this is unrealistic since it is not desired to run down stocks. This can be dealt with in several ways:

a) set the final inventory levels to reasonable values (say the same as inventory levels at the beginning of the first period);
b) assign a value to final inventory; this way the model can decide if it is worthwhile to produce stock to sell at the end of the final period.

Logical Constraints and MINLP Formulations

It is often necessary to impose additional logical constraints that dictate how various components are to be blended in relation to each other. Modeling such constraints often requires the addition of integer variables, as discussed below.

a) If a component is to be used in a particular blend, then it must be present in at least a certain amount in the blend. This arises from the fact that it is usually not practical to blend in infinitesimally small quantities.
   If x represents the volume of such a component, then introducing a new binary variable δ (i. e. δ is either 0 or 1) and the constraints

$$x - M\delta \leq 0,$$
$$x - m\delta \geq 0$$

   are sufficient to ensure this condition is satisfied. Here, M is a sufficiently large number, while m represents the threshold value below which a component should not be blended in.
b) Each product can have at most k components in its blend. This is typically imposed by limitations on how many streams can be physically blended in a reasonable amount of time. Again, introducing the

new variables and constraints as below:

$$x_1 - m\delta_1 \geq 0,$$
$$\dots$$
$$x_n - m\delta_n \geq 0,$$
$$\delta_1 + \cdots + \delta_n \leq k,$$
$$\delta_1, \dots, \delta_n \in \{0-1\}^n,$$

ensures this condition is met.

c) If component A is to be present in the blend, then component B must also be present:

$$x_A - m\delta_A \geq 0,$$
$$x_B - m\delta_B \geq 0,$$
$$\delta_B \geq \delta_A.$$

Each of these logical constraints results in a mixed integer nonlinear programming (MINLP) model (cf. also ► Mixed integer nonlinear programming). To date (2000), such models have not been used extensively in the practical solution of these problems in industry.

Complexity of Models

With the various options of single versus multiperiod and linear versus nonlinear blending, the models for pooling and blending can vary significantly in complexity. This is shown pictorially in Fig. 4.

Solution Methods

Pooling problems can be solved using a variety of solution algorithms. These can be broadly classified as local and global solution methods.

Local Optimization Approaches

Traditionally, pooling and blending problems have been solved using various recursion and successive linear programming (SLP) techniques. The first published approach for solving the pooling problem was due to Haverly [8], who proposed the following recursion approach for solving the problem given in Fig. 2:

```
1  Start with a guess for the pool quality $p$.
2  Solve the remaining linear problem for all
   other variables.
3  Calculate a new value for $p$ from the solution
   in 2).
```

Unfortunately, this rather simple recursion will converge to a suboptimal solution regardless of the starting value for p. This can be partially addressed by using a 'distributed recursion' approach, where an additional recursion coefficient f and two additional 'correction vectors' are introduced, modifying the inequalities in the model as follows:

$$\begin{aligned} &p \cdot y_{11} + 2z_{31} - 2.5(y_{11} + z_{31}) \\ &\quad + f(\text{over} - \text{under}) \leq 0, \\ &p \cdot y_{12} + 2z_{32} - 1.5(y_{12} + z_{32}) \\ &\quad + (1-f)(\text{over} - \text{under}) \leq 0. \end{aligned}$$

This formulation serves to distribute the error made in estimating the pool quality to the two pool destinations. Recursing on both p and f has a better likelihood of identifying the optimal solution.

SLP algorithms solve nonlinear models through a sequence of linear programs (LPs), each of which is a linearized version of the model around some base point. These methods consist of replacing nonlinear constraints of the form

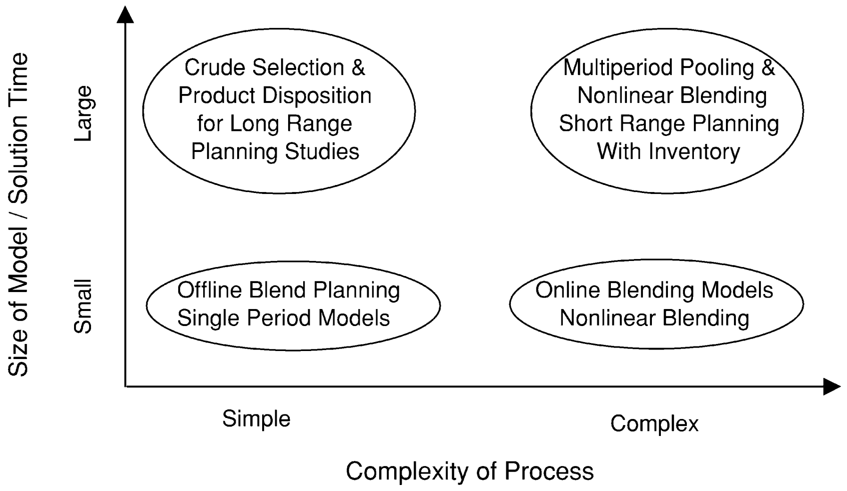
$$g(x) \leq 0, \qquad h(x) = 0,$$

with the linearizations

$$\begin{aligned} &g(\overline{x}^k) + \nabla g(\overline{x}^k) \cdot (x - \overline{x}^k) \leq 0, \\ &h(\overline{x}^k) + \nabla h(\overline{x}^k) \cdot (x - \overline{x}^k) = 0 \end{aligned}$$

around a base point $\overline{x}^k$ at the kth iteration. The linearized problems can be solved using standard LP methods. The solution to the problem is used to provide a value for $\overline{x}^{k+1}$. As long as there is an improvement in the objective function value as well as the feasibility of the original constraints, these methods can be shown to converge to a local optimum. They work well for largely linear problems and have therefore found widespread use in the refining industry for solving pooling, blending and general refinery planning problems [4,11]. However, when there are nonlinear blending constraints, the linearization in the SLP is often a bad approximation of the original problem, leading to poor convergence rates and large solution times.

Pooling and blending problems can also be solved using other nonlinear programming (NLP) methods such as generalized reduced gradient, successive quadratic programming or penalty function methods.


Size of Model / Solution Time
Large
Small
Crude Selection & Product Disposition for Long Range Planning Studies
Multiperiod Pooling & Nonlinear Blending Short Range Planning With Inventory
Offline Blend Planning Single Period Models
Online Blending Models Nonlinear Blending
Simple
Complex
Complexity of Process

MINLP: Applications in Blending and Pooling Problems, Figure 4
Types of pooling problems

In general, these methods have not found large acceptance in solving these problems, mainly due to difficulties with convergence and stability.

Global Optimization Approaches

The recursive, SLP and conventional NLP techniques all suffer from the drawback that the solution found is highly dependent on the starting point, and in general cannot guarantee convergence to the global solution. In the last dozen years, numerous approaches have been proposed for the solution of quadratically constrained optimization problems (such as the pooling/blending problem). Surveys of these algorithms can be found in [10,12]. These approaches can generally be classified as either decomposition-based or branch and bound algorithms.

One of the common approaches to dealing with the nonconvexities in the pooling problem is to reduce the bilinear terms to linear terms over a convex envelope [2]. Noting that for any bilinear term $p \cdot y$,

$$\begin{aligned}
&(p - p^L) \cdot (y - y^L) \geq 0,\\
&(p - p^U) \cdot (y - y^U) \geq 0,\\
&(p - p^L) \cdot (y - y^U) \leq 0,\\
&(p - p^U) \cdot (y - y^L) \leq 0,
\end{aligned}$$

where $[p^L, p^U]$ and $[y^L, y^U]$ define the ranges for the variables p and y. This allows the term $p \cdot y$ to be replaced by a set of linear inequalities in the model, resulting in a linearized problem which provides an upper bound on the global solution to the original problem. After solving this problem, the rectangle defined by the bounds on p and y can be subdivided into smaller rectangles, and a new linearized problem can be solved over each of these subrectangles. By continuously subdividing these rectangles, the upper bound can be made to asymptotically approach the global solution. See [7] for the solution of several pooling problems using this approach.

Note that the pooling problem is a partially linear problem. That is, it can be formulated as

$$\begin{cases} \min\limits_{x,p} & c^\top x \\ \text{s.t.} & A(p)x \leq b, \end{cases} \tag{1}$$

where p represents the pool quality and x represents all component flow rates. For such problems, decomposition approaches provide a natural solution mechanism. For a fixed value of p, this problem is linear, and provides an upper bound on the global solution. The solution to this linear problem (called the 'primal' problem) can be used to generate a *Lagrange function* of the form

$$L(x, p) = c^\top x + \lambda \cdot (A(p)x - b)$$

where λ represents the multipliers or marginal values for the constraints from the primal problem. Then, the 'dual' problem

$$\begin{cases} \min\limits_{x,p,\mu} & \mu \\ \text{s.t.} & \mu \geq L(x,p) \end{cases} \tag{2}$$

provides an upper bound on the global solution. Problem (2) contains bilinear terms of the form $A(p)x$, which can be underestimated in a variety of ways. C.A. Floudas and V. Visweswaran [5,6] have developed the GOP algorithm based on this approach. By alternating between the primal problem and a series of relaxed dual problems (developed by successively partitioning the feasible region), the GOP algorithm guarantees convergence to the global solution. In [13,14], they show that it is possible to develop properties that reduce the number of relaxed dual problems that need to be solved, thus speeding up the overall algorithm. They also report the solution of numerous pooling and blending problems using this approach.

Instead of fixing p for the primal problem, it is possible to solve (1) directly using local optimization techniques. For example, nonsmooth optimization techniques can be effective in finding local solutions to these problems [1]. The dual problem can also be solved this way, with the region for p being refined by partitioning. See [1] for the solution of several pooling problems using this approach.

It is important to note that these global optimization approaches (and others) for solving the pooling problem can be computationally intensive. Invariably, a large number of subproblems need to be solved before convergence to a global solution can be guaranteed. Because the subproblems are usually of the same structure, varying only slightly in the data for the problems, they can be solved in parallel. See [3] for an implementation of a distributed parallel version of the GOP algorithm and a successful application to solve pooling problems of medium size.

Applications

The most common application of pooling and blending models is in the *refining* and *petrochemical industries*. Crude oil from various sources is often brought into the refinery and stored in common tanks before being processed downstream. Similarly, intermediate streams from various refinery processes (alkylation, reforming, cracking) are usually sent to common pools from which finished products such as gasoline and diesel oil are made. In both cases, it is important to know various qualities of the stream coming out of the pool (such as chemical compositions like sulfur or physical properties such as vapor pressure).

In addition to refinery processes, blending is a feature of various other manufacturing processes. These include

- *agriculture*, where blending livestock feeds or fertilizers at minimum cost is very important;
- mining, where different ores are often mixed to achieve a desired quality;
- various aspects of food manufacturing; and
- pulp and paper, involving blending of raw materials used to produce paper.

See also

▶ Chemical Process Planning
▶ Extended Cutting Plane Algorithm
▶ Generalized Benders Decomposition
▶ Generalized Outer Approximation
▶ MINLP: Application in Facility Location-allocation
▶ MINLP: Applications in the Interaction of Design and Control
▶ MINLP: Branch and Bound Global Optimization Algorithm
▶ MINLP: Branch and Bound Methods
▶ MINLP: Design and Scheduling of Batch Processes
▶ MINLP: Generalized Cross Decomposition
▶ MINLP: Global Optimization with αBB
▶ MINLP: Heat Exchanger Network Synthesis
▶ MINLP: Logic-based Methods
▶ MINLP: Outer Approximation Algorithm
▶ MINLP: Reactive Distillation Column Synthesis
▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
▶ Mixed Integer Nonlinear Programming

References

1. Ben-Tal A, Eiger G, Gershovitz V (1994) Global minimization by reducing the duality gap. Math Program 63:193
2. Al-Khayyal FA, Falk JE (1983) Jointly constrained biconvex programming. Math Oper Res 8(2):273
3. Androulakis IP, Isweswaran VV, Floudas CA (1995) Distributed decomposition-based approaches in global optimization. In: Floudas CA, Pardalos PM (eds) Proc. State

of the Art in Global Optimization: Computational Methods and Applications. Kluwer, Dordrecht, pp 285–301
4. Baker TE, Lasdon LS (1994) Successive linear programming at Exxon. Managem Sci 31(3):264
5. Floudas CA, Visweswaran V (1990) A global optimization algorithm (GOP) for certain classes of nonconvex NLPs: I. Theory. Comput Chem Eng 14:1397
6. Floudas CA, Visweswaran V (1993) A primal-relaxed dual global optimization approach. J Optim Th Appl 78(2):187
7. Foulds LR, Haugland D, Jörnsten K (1990) A bilinear approach to the pooling problem. Chr. Michelsen Inst. Working Paper 90-3
8. Haverly CA (1978) Studies of the behaviour of recursion for the pooling problem. ACM SIGMAP Bull 25:19
9. Haverly CA (1979) Behaviour of recursion model–more studies. ACM SIGMAP Bull 26:22
10. Horst R, Tuy H (1993) Global optimization: Deterministic approaches. second Springer, Berlin
11. Lasdon LS, Waren AD, Sarkar S, Palacios-Gomez F (1979) Solving the pooling problem using generalized reduced gradient and successive linear programming algorithms. ACM SIGMAP Bull 27:9
12. Pardalos PM, Rosen JB (1987) Constrained global optimization: Algorithms and applications. Lecture Notes Computer Sci, vol 268. Springer, Berlin
13. Visweswaran V, Floudas CA (1996) Computational results for an efficient implementation of the GOP algorithm and its variants. In: Grossmann IE (ed) Global Optimization in Engineering Design. Nonconvex Optim Appl. Kluwer, Dordrecht, pp 111–154
14. Visweswaran V, Floudas CA (1996) New formulations and branching strategies for the GOP algorithm. In: Grossmann IE (ed) Global Optimization in Engineering Design. Kluwer, Dordrecht, pp 75–110

MINLP: Applications in the Interaction of Design and Control

Carl A. Schweiger, Christodoulos A. Floudas
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90C11, 49M37

Article Outline

Keywords

Mixed integer nonlinear optimization; Parametric optimal control; Interaction of design and control

In the development of a process, the steady state design aspects and dynamic operability issues are usually handled sequentially. First, the design engineers develop and synthesize the structure of the flowsheet and determine the operating parameters and steady-state operating conditions. Then, the control engineer takes the fixed design and develops a control system to maintain the system at the desired specifications. During the first step, the dynamic operation of the process is generally not considered, and in the second step, changes to the flowsheet and operating conditions generally can not be made.

Process design seeks to determine the arrangement of processing units that will convert the given raw materials into the desired products. The idea is to develop a process flowsheet from the large number of possible design alternatives. Numerous process design methods and techniques exist for determining the best process flowsheet and operating conditions. This best design is determined by optimizing some economic criteria and the quality of the design is based on its economic value. Hence, the process is designed to operate at steady state and issues relating to the *process dynamics*, *operability*, and *controllability* are usually not considered.

Once the process has been designed, the plans are handed over to the *process control* engineer whose task is to ensure the stable dynamic performance of the process. The control engineer is concerned with developing a control system which maintains the operation of the process at the desired steady state in the presence ever-changing external influences. Issues such as disturbances, uncertainty, and changes in production rates must be addressed so as to maintain product quality and safe operation. By addressing the design and control sequentially, the inherent connection between the two is neglected. For instance, the steady-state design of a process may appear to produce great economic profits. However, unfavorable dynamic operation may lead to a product which does not meet the required specifications. This may result in an economic loss due to disposal or reworking costs. Thus, a process design with good controllability aspects may have better economic value that an economically optimal steady state design when the dynamic operation is considered. This trade-off between the steady state design and the dynamic controllability motivates the treatment of the issues simultaneously.

There are additional incentives for employing a simultaneous approach. Due to economic and environmental reasons, the recent trend in process design has been towards more highly integrated process in terms of both material and energy flows. Processes are also required to operate under much tighter operating conditions due to environmental and safety issues. Both of these lead to designs with increased dynamic interactions and processes which are generally more difficult to control. Thus, the dynamic operation of the process must be considered at the early stages of the design.

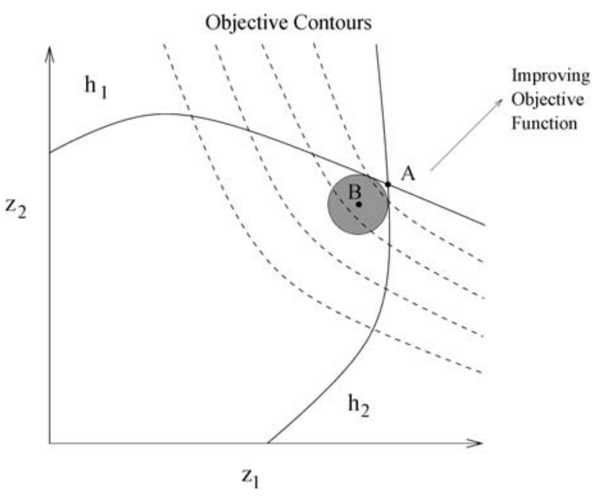
A systematic method for analyzing the *interaction of design and control* requires quantitative *controllability measures* of the process. Such measures have been derived to quantify certain qualitative concepts about the controllability of the process such as inversion, interaction effects, and directionality problems. A common measure for controllability is the integral squared error (ISE) between outputs and their desired levels. Although it is easy to measure, it is not of direct interest in practice. Other performance criteria such as maximum deviation of output variables, maximum magnitude of control variables, or time to return to steady state can also be used.

Most of the work in the development of controllability measures has focused on linear dynamic models. The control objective is the robust performance of the process without any restrictions on the controller structure [15]. One such measure is the structured singular value, σ, which indicates the performance in the presence of uncertainty. The condition number, γ, has been developed as an indicator of closed-loop sensitivity to model error while the disturbance conditions number, γ_d, indicates the sensitivity of the process to disturbances. The relative gain array (RGA), Λ, is used as an indicator of the relationship between control error and set point changes while the closed-loop disturbance gain (CLDG) is used to measure the relation between control error and disturbances. These measures have been used extensively in applications for controllability assessment; however, they can be misleading. While these indicators give ideas as to the closed loop performance of the process, their impact on the economics of the process is not clear.

Previous Work

In comparison to the amount of research on the controllability measures, relatively little work has been placed on methods for systematically determining the *trade-offs* between steady-state economics and dynamic controllability. Although economics continues to be the driving force in the design of a process, there is no straightforward method for evaluating the economics of the dynamic operation of the process. Several methods have been proposed to address these issues. M. Morari and J.D. Perkins [14] discuss the concept of controllability and emphasize that the design of a control system for a process is part of the overall design of the process. Noting that a great amount of effort has been placed on the assessment of controllability, particularly for linear dynamic models, they indicate that very little has been published on algorithmic approaches for determination of process designs where economics and controllability are traded off systematically.

In order to deal with the controllability issues on a economic level, a *back-off* method was presented in [18] to determine the economic impact of disturbances on the system. The basic idea is to determine the optimal steady-state operating point such that the feasible operation is maintained with respect to all constraints in the presence of uncertainties and disturbances. This operating point is compared to the optimal steady-state operating point determined in the absence of disturbances. The economic penalty incurred by backing away from the disturbances-free operating point to the feasible operating point can be determined and thus the cost of the disturbance can be evaluated. This concept is illustrated in Fig. 1. Point A indicates the nominal steady-state design, and point B is the back-off point which corresponds to the design which will not violate the constraints h_1 and h_2 in the presence of uncertainties and disturbances.



MINLP: Applications in the Interaction of Design and Control, Figure 1
Illustration of the back-off approach

The method is further developed in [17], where the control structure selection problem is analyzed. Perfect control assumptions are used along with a linearized model to formulate a mixed integer linear program (MILP) where the integer variables indicate the pairings between the manipulated and controlled variables. The back-off approach incorporated the dynamic operation of the process into the design, but it only ensures the feasible operation of the process and does not directly address controllability aspects.

An approach for determining process designs which are both steady-state and operationally optimal was presented in [2]. The controllability of potential designs is evaluated along with their economic performance by incorporating a model predictive control algorithm into the process design optimization algorithm. This coordinated approach uses an objective function which is a weighted sum of economic and controllability measures.

A multi-objective approach was proposed in [9,10] to simultaneously consider both controllability and economic aspects of the design. This approach incorporates both design and control aspects into a process synthesis framework where the trade-offs between various open-loop controllability measures and the economics of the process can be observed. The problem is formulated as a *mixed integer nonlinear program* (*MINLP*), where integer variables are utilized for structural alternatives in the process flowsheet. Through the application of multi-objective techniques, a process design which is both economic and controllable is determined.

A screening approach was proposed in [4], where the variability in the product quality is used to compare different steady-state process designs. The dynamic controllability is measured economically by calculating the amount of material produced that is off-specification and on-specification. The on-specification material leads to profits while the off-spec material results in costs for reworking or disposal.

A back-off technique was also developed in [1] for the design of steady-state and open-loop dynamic processes. Both uncertainties and disturbances are considered for determining the amount of back-off. In order to address the fact that back-off approaches address the feasible operation and do not address controllability aspects, [5] introduces a recovery factor which is defined as the ratio of the amount of penalty recovered with control to the penalty with no control. This ratio is then used to rank different control strategies.

The advantage of the back-off approaches is that they determine the cost increase associated with moving to the back-off position which is attributed to the uncertainties and disturbances. A limitation of this approach is that it can lead to rather conservative designs since the worst-case uncertainty scenario is considered. Although the probability of the worst-case uncertainty occurring may not be high, this is the basis for the final design. Also, the method has not been applied to the design/synthesis problem. A fixed design is considered and then the back-off is considered as a modification of this design.

The optimal design of dynamic systems under uncertainty was addressed in [13]. Flexibility aspects as well as the control design were considered simultaneously with the process design. The algorithm is used to find the economic optimum which satisfies all of the constraints for a given set of uncertainties and disturbances when the control system is included.

S. Walsh and Perkins [23] outline the use of optimization as a tool for the design/control problem. They note that the advances in computational hardware and optimization tools have made it possible to solve the complex problems that arise in design/control. Their assessment focuses on the control structure selection problem where the economic cost of a disturbance is balanced against the performance of the controller.

The increasing importance of design and control issues had lead to more and more discussion on the topic. One contribution to the area has been [11]. The fundamental design and control concepts are described and several quantitative examples are given which illustrate the interaction of design and control.

Most of the previous work does not address synthesis issues and does not treat the problem quantitatively. Two methods employ the optimization approach in process synthesis to arrive at mathematical programming formulations which are solved to determine the trade-offs between the steady-state design and dynamic controllability. The first method [9,10] uses steady state linear controllability measures while the second method [20] uses full nonlinear dynamic models of the process.

Process Synthesis

Mathematical programming has been found to be a very useful tool for process synthesis. Its application in analyzing the interaction of design and control has followed directly along the process synthesis methodology.

The goal in process synthesis to determine the structure and operating conditions of the process flowsheet. The optimization approach to the synthesis problem involves three steps:

1) The representation of process design alternatives of interest through a process *superstructure*.
2) The *mathematical modeling* of the superstructure.
3) The *algorithmic development* of solution procedure to extract the optimal process flowsheet from the superstructure and solution of the optimization problem.

The key aspect is the postulation of a superstructure which contains all possible design alternatives of interest. The superstructure must be sufficiently rich so as to include the numerous design possibilities yet succinct enough to eliminate redundancies and reduce complexities.

The mathematical model is characterized by the variables and equations used in the model. Continuous variables are used to represent flowrates, compositions, temperatures, etc. Binary variables are used to represent structural alternatives such as the existence of process units. The modeling of steady-state processes leads to algebraic equations and constraints and results in an MINLP. When dynamic models are to be used, the continuous variables are partitioned into dynamic state variables, control variables, and time invariant variables, and the resulting formulation is classified as a *mixed integer optimal control problem* (MIOCP).

Steady-State Modeling Approach

This approach was outlined in [9,10] and follows the optimization approach for process synthesis. A systematic procedure is presented for incorporating open-loop steady-state controllability measures into the process synthesis problem. The problem is formulated mathematically as a MINLP and a *multi-objective optimization* problem is solved to quantitatively determine the best-compromise solution among the economic and control objectives. The ϵ-constraint method is used to determine the *noninferior solution set* where one objective can be improved only at the expense of another, and the best-compromise solution is determined using a *cutting plane algorithm*.

In order to apply the process synthesis approach, the controllability measure must be expressed as a function of the unknown design parameters. Steady-state controllability measures are used to simplify the problem and reduce implementation difficulties that arise when considering controllability measures as functions of frequency. The steady-state gains of the process can be written in an analytical form thus allowing for an algebraic representation.

The starting point for the controllability analysis is the linear multiple input/multiple output system written in the Laplace domain as

$$\mathbf{z}(s) = \mathbf{G}(s)\mathbf{u}(s) + \mathbf{G}_d(s)\mathbf{d}(s),$$

where $\mathbf{z}$ are the output variables, $\mathbf{u}$ are the control variables, $\mathbf{G}(s)$ is the process transfer function matrix, and $\mathbf{G}_d(s)$ is the disturbance transfer function matrix.

Closed-loop control can be considered by expressing the control variable $\mathbf{u}(s)$ as

$$\mathbf{u}(s) = \mathbf{G}_c(s)(\mathbf{z}^*(s) - \mathbf{z}(s)),$$

where $\mathbf{G}_c(s)$ is the controller transfer function and $\mathbf{z}^*$ is the desired set-point. This requires that the form of controller transfer function be known as well as the method for calculating the parameters. Since this causes problems in the formulation of the optimization problem, the controllability is viewed as a property inherent to the process and independent of the particular control system design. The analysis thus considers only the open-loop controllability measures which depend only on the process itself.

Since both the process design and controllability measures can be expressed as functions of the unknown design parameters, the synthesis problem can be expressed as a multi-objective MINLP:

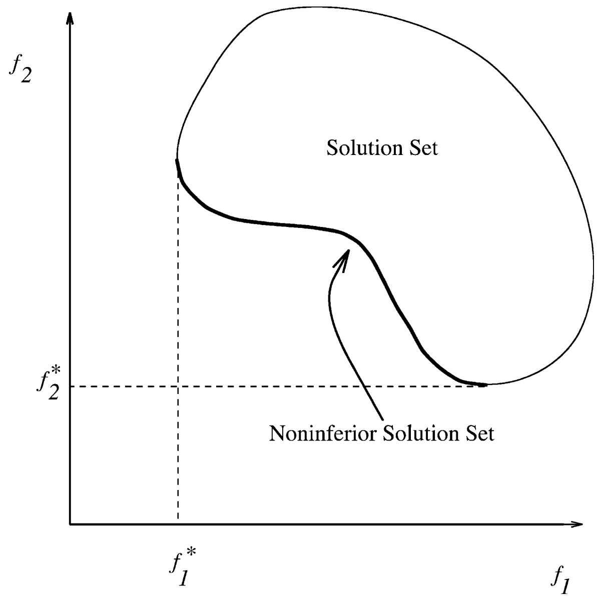
$$\begin{cases} \min & \mathbf{J}(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & \boldsymbol{\eta} = \overline{\mathbf{h}}(\mathbf{x},\mathbf{y}) \\ & \mathbf{x} \in X \subseteq \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q . \end{cases}$$

In this formulation, **J** is a vector of objectives which includes both the economic objectives and controllability objectives. The expressions **h** and **g** represent material and energy balances, thermodynamic relations, and other constraints. The controllability measures are included in the formulation as $\boldsymbol{\eta}$. The variables in this problem are partitioned as continuous **x** and binary **y**.

The problem is posed with multiple objectives representing the competing economic and open-loop controllability measures. Different techniques have been developed in order to assess the trade-offs among the objectives quantitatively. In this approach, the noninferior solution set is generated to determine the set of solutions in which one objective can be improved only at the expense of the other(s). The noninferior solution set for a two objective problem is visually depicted in Fig. 2.

This noninferior solution set is generated using an *ϵ-constraint method* where one objective is optimized and the others are included as constraints less than a parameter ϵ. The problem is reduced to a single objective optimization problem which is iteratively solved for varying values of ϵ to generate the noninferior solution set.

By reducing the problem to a single objective problem, MINLP optimization techniques can be applied to solve the problem. These MINLP techniques include *generalized Benders decomposition* (GBD) [7,19], outer approximation (OA) [3], outer approximation with equality relaxation (OA/ER) [8], and outer approximation with equality relaxation and augmented penalty [22]. These are discussed in detail in [6].



MINLP: Applications in the Interaction of Design and Control, Figure 2
Noninferior solution set for a problem with two objectives

Once the noninferior solution set is determined, the best compromise solution is determined by applying a *cutting plane algorithm*. The trade-offs among the objectives are quantitatively assessed using weight factors which come from the slope of the noninferior solution set.

Dynamic Modeling Approach

The major limitation of the above approach is that is does not consider the dynamic behavior of the process. This approach considers the full dynamic model of the process and a dynamic controllability measure. An optimization approach is applied which involves a dynamic optimization problem.

One of the initial difficulties with this method is defining a controllability measure for nonlinear dynamic systems. As in the previous method, the controllability measure must be capable of being expressed as a function of the unknown design parameters. One possible choice for the controllability measure is the integral square error (ISE). The benefit of this measure is that it is easy to calculate and and does reflect the dynamics of the process albeit only in the outputs of the process. One downside of this measure is that there is no one to one correspondence between the the control structure and the ISE measure. Thus, different dynamic characteristics of the process may not be reflected in the ISE.

The superstructure is the same as in the previous approach, but a dynamic model is used instead of a steady-state model. The dynamic modeling of the superstructure leads to a problem that includes *differential and algebraic equations* (DAEs) and the formulation is a multi-objective MIOCP. New algorithmic techniques must be developed for the solution of the formulation.

The general formulation for the multi-objective MIOCP is as follows:

$$\left\{\begin{array}{ll}\min & \mathbf{J}(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{u}(t_i), \mathbf{x}, \mathbf{y}) \\ \text{s.t.} & \mathbf{f}_1(\dot{\mathbf{z}}_1(t), \mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{u}(t), \mathbf{x}, \mathbf{y}, t) = \mathbf{0} \\ & \mathbf{f}_2(\mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{u}(t), \mathbf{x}, \mathbf{y}, t) = \mathbf{0} \\ & \mathbf{z}_1(t_0) = \mathbf{z}_1^0 \\ & \mathbf{z}_2(t_0) = \mathbf{z}_2^0 \\ & \mathbf{h}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{u}(t_i), \mathbf{x}, \mathbf{y}) = \mathbf{0} \\ & \mathbf{g}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{u}(t_i), \mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\ & \mathbf{h}''(\mathbf{x}, \mathbf{y}) = \mathbf{0} \\ & \mathbf{g}''(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\ & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q \\ & t_i \in [t_0, t_N] \\ & i = 0, \ldots, N. \end{array}\right. \tag{1}$$

Here, $\mathbf{z}_1(t)$ is a vector of n dynamic variables whose time derivatives, $\dot{\mathbf{z}}_1(t)$, appear explicitly, and $\mathbf{z}_2(t)$ is a vector of m dynamic variables whose time derivatives do not appear explicitly, $\mathbf{x}$ is a vector of p time invariant continuous variables, $\mathbf{y}$ is a vector of q binary variables, and $\mathbf{u}(t)$ is a vector of r control variables. Time t is the independent variable for the DAE system where t_0 is the fixed initial time, t_i are time instances, and t_N is the final time. The DAE system is represented by $\mathbf{f}_1$, the n differential equations, and $\mathbf{f}_2$, the m dynamic algebraic equations. The constraints $\mathbf{h}'$ and $\mathbf{g}'$ are point constraints where t_i represents the time instance at which the constraint is enforced and $\mathbf{h}''$ and $\mathbf{g}''$ are general constraints. The objective functions for the economic and controllability measures are represented by the vector $\mathbf{J}$.

The initial condition for the above system is determined by specifying n of the $2n + m$ variables $\mathbf{z}_1(t_0)$, $\dot{\mathbf{z}}_1$ (t_0), $\mathbf{z}_2$ (t_0). For DAE systems with index 0 or 1, the remaining $n + m$ values can be determined. In this work, DAE systems of index 0 or 1 are considered and the initial conditions for $\mathbf{z}_1(t)$ and $\mathbf{z}_2(t)$ are $\mathbf{z}_1^0$ and $\mathbf{z}_2^0$ respectively.

Note that in this general formulation, the $\mathbf{y}$ variables appear in the DAE system as well as in the point constraints and general constraints. This has implications on the solution strategy.

A similar approach to that of the previous approach is applied to address the multi-objective nature of the problem. An ϵ-constraint method is applied to reduce to problem to an iterative solution of single objective MIOCPs.

MIOCP Solution Algorithm

The strategy for solving the MIOCP is to apply iterative decomposition strategies similar to existing MINLP algorithms with extensions for handling the DAE system. The algorithm developed for the solution of the MIOCP closely parallels existing algorithms for MINLP optimization (GBD, OA, OA/ER, OA/ER/AP). The presence of the $\mathbf{y}$ variables in DAE system for the general case prohibits the use of Outer Approximation and its variants. For the special cases where the $\mathbf{y}$ variables do not appear in the DAEs and do participate in a linear and separable fashion, outer approximation and its variants can be applied to the problem. The GBD algorithm can be applied to the solution of the general problem, and the algorithmic development closely follows those of GBD.

The GBD algorithm is an iterative procedure which generates upper and lower bounds on the solution of the MINLP formulation. The upper bound results from the solution of an NLP primal problem and the lower bound from an MILP master problem. The bounds on the solution converge in a finite number of iterations to yield the solution to the MINLP model. A similar methodology is applied to the MIOCP problem, but the forms of the primal and master problems have to be altered.

Primal Problem

The primal problem is obtained by fixing the **y** variables which leads to an optimal control problem. For fixed values of $\mathbf{y} = \mathbf{y}^k$, the MIOCP has the following form:

$$\left\{\begin{array}{ll}\min & \mathbf{J}(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{u}(t_i), \mathbf{x}, \mathbf{y}^k) \\ \text{s.t.} & \mathbf{f}_1(\dot{\mathbf{z}}_1(t), \mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{u}(t), \mathbf{x}, \mathbf{y}^k, t) = \mathbf{0} \\ & \mathbf{f}_2(\mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{u}(t), \mathbf{x}, \mathbf{y}^k, t) = \mathbf{0} \\ & \mathbf{z}_1(t_0) = \mathbf{z}_1^0 \\ & \mathbf{z}_2(t_0) = \mathbf{z}_2^0 \\ & \mathbf{h}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{u}(t_i), \mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & \mathbf{g}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{u}(t_i), \mathbf{x}, \mathbf{y}^k) \leq \mathbf{0} \\ & \mathbf{h}''(\mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & \mathbf{g}''(\mathbf{x}, \mathbf{y}^k) \leq \mathbf{0} \\ & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^p \\ & t_i \in [t_0, t_N] \\ & i = 0, \dots, N. \end{array}\right. \tag{2}$$

The solution of this optimal control problem can be handled in several ways: complete discretization, solution of the necessary conditions, dynamic programming, and *control parameterization*. This work focuses on the control parameterization techniques which parameterize only the control variables $\mathbf{u}(t)$ in terms of time invariant parameters. At each step of the optimization procedure, the DAEs are solved for given values of the decision variables and a feasible path for $\mathbf{z}(t)$ is obtained. This solution is used to evaluate the objective function and remaining constraints. The control parameterization can either be open loop as described in [21] or closed-loop such as that described in [17] and [16] which also includes the control structure selection.

The basic idea behind the control parameterization is to express the control variables $\mathbf{u}(t)$ as functions of time invariant parameters. This parameterization can be done in terms of the independent variable t (open loop):

$$\mathbf{u}(t) = \phi(\mathbf{w}, t).$$

Alternatively, the parameterization can be done in terms of the state variables $\mathbf{z}(t)$ (closed-loop):

$$\mathbf{u}(t) = \psi(\mathbf{w}, \dot{\mathbf{z}}(t), \mathbf{z}(t)).$$

In both cases, **w** are the time invariant control parameters. The set of time invariant parameters, **x**, is now expanded to include the control parameters:

$$\mathbf{x} = \{\mathbf{x}, \mathbf{w}\}.$$

The set of DAEs (**f**) is expanded to include parameterization functions

$$\mathbf{f}(\cdot) = \{\mathbf{f}(\cdot), \phi(\cdot), \psi(\cdot)\}$$

and the control variables are converted to dynamic state variables:

$$\mathbf{z} = \{\mathbf{z}, \mathbf{u}\}.$$

Through the application of the control parameterization, the control variables are effectively removed from the problem and the following problem results:

$$\left\{\begin{array}{ll}\min & J(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{x}, \mathbf{y}^k) \\ \text{s.t.} & \mathbf{f}_1(\dot{\mathbf{z}}_1(t), \mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{x}, \mathbf{y}^k, t) = \mathbf{0} \\ & \mathbf{f}_2(\mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{x}, \mathbf{y}^k, t) = \mathbf{0} \\ & \mathbf{z}_1(t_0) = \mathbf{z}_1^0 \\ & \mathbf{z}_2(t_0) = \mathbf{z}_2^0 \\ & \mathbf{h}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & \mathbf{g}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{x}, \mathbf{y}^k) \leq \mathbf{0} \\ & \mathbf{h}''(\mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & \mathbf{g}''(\mathbf{x}, \mathbf{y}^k) \leq \mathbf{0} \\ & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^p \\ & t_i \in [t_0, t_N] \\ & i = 0, \dots, N. \end{array}\right. \tag{3}$$

This problem is a nonlinear program with differential and algebraic constraints (NLP/DAE). This problem is solved using a parametric method where the DAE system is solved as a function of the **x** variables. The solution of the DAE system is achieved through an integration routine which returns the values of the **z** variables at the time instances, $\mathbf{z}(t_i)$, along with their sensitivities with respect to the parameters, $\mathrm{d}\mathbf{z}/\mathrm{d}\mathbf{x}(t_i)$. The resulting problem is an NLP optimization over the space of **x** variables which has the form:

$$\begin{cases} \min & J(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{x}, \mathbf{y}^k) \\ \text{s.t.} & \mathbf{h}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & \mathbf{g}'(\dot{\mathbf{z}}_1(t_i), \mathbf{z}_1(t_i), \mathbf{z}_2(t_i), \mathbf{x}, \mathbf{y}^k) \le \mathbf{0} \\ & \mathbf{h}''(\mathbf{x}, \mathbf{y}^k) = \mathbf{0} \\ & \mathbf{g}''(\mathbf{x}, \mathbf{y}^k) \le \mathbf{0} \\ & \mathbf{x} \in \mathcal{X} \\ & t_i \in [t_0, \dots, t_N] \\ & i = 0, \dots, N, \end{cases} \tag{4}$$

where the variables $\dot{\mathbf{z}}_1(t_i)$, $\mathbf{z}_1(t_i)$, and $\mathbf{z}_2(t_i)$ are determined through the solution of the DAE system by integration:

$$\begin{cases} \mathbf{f}_1(\dot{\mathbf{z}}_1(t), \mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{x}, \mathbf{y}^k, t) = \mathbf{0}, \\ \mathbf{f}_2(\mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{x}, \mathbf{y}^k, t) = \mathbf{0}, \\ \mathbf{z}_1(t_0) = \mathbf{z}_1^0, \\ \mathbf{z}_2(t_0) = \mathbf{z}_2^0. \end{cases} \tag{5}$$

The functions $J(\cdot)$, $\mathbf{g}'(\cdot)$, and $\mathbf{h}'(\cdot)$ are functions of $\mathbf{z}(t_i)$ which are implicit functions of the **x** variables through the integration of the DAE system. For the solution of the NLP the objective and constraints evaluations, along with their gradients with respect to **x**, are required. These are evaluated directly for the constraints $\mathbf{g}''(\mathbf{x})$ and $\mathbf{h}''(\mathbf{x})$. However, for the functions $J(\cdot)$, $\mathbf{g}'(\cdot)$, and $\mathbf{h}'(\cdot)$, the values $\mathbf{z}(t_i)$, and the gradients $d\mathbf{z}/d\mathbf{x}(t_i)$, as returned from the integration, are used. The functions $J(\cdot)$, $\mathbf{g}'(\cdot)$, and $\mathbf{h}'(\cdot)$ are evaluated directly and the gradients $dJ/d\mathbf{x}$, $d\mathbf{g}_i'/d\mathbf{x}$, and $d\mathbf{h}'/d\mathbf{x}$ are evaluated by using the chain rule:

$$\begin{cases} \frac{dJ}{d\mathbf{x}} = \left(\frac{\partial J}{\partial \mathbf{z}}\right)\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) + \left(\frac{\partial J}{\partial \mathbf{x}}\right), \\ \frac{d\mathbf{h}_i'}{d\mathbf{x}} = \left(\frac{\partial \mathbf{h}_i'}{\partial \mathbf{z}}\right)\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) + \left(\frac{\partial \mathbf{h}_i'}{\partial \mathbf{x}}\right), \\ \frac{d\mathbf{g}_i'}{d\mathbf{x}} = \left(\frac{\partial \mathbf{g}_i'}{\partial \mathbf{z}}\right)\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) + \left(\frac{\partial \mathbf{g}_i'}{\partial \mathbf{x}}\right). \end{cases} \tag{6}$$

Standard gradient based optimization techniques can be applied to solve this problem as an NLP. The solution of this problem provides values of the **x** variables and trajectories for $\mathbf{z}(t)$.

The master problem is formulated using *dual information* and the solution of the primal problem. Provided that the **y** variables participate linearly, the problem is an MILP whose solution provides a lower bound and **y** variables for the next primal problem. Dual information is required from all of the constraints including the DAEs whose dual variables, or adjoint variables, are dynamic. The constraints and their corresponding dual variables are listed in Table 1.

MINLP: Applications in the Interaction of Design and Control, Table 1
Constraints and their corresponding dual variables

constraint	dual variable
$\mathbf{f}_1$	$\boldsymbol{\nu}_1(t)$
$\mathbf{f}_2$	$\boldsymbol{\nu}_2(t)$
$\mathbf{g}'$	$\boldsymbol{\mu}'$
$\mathbf{h}'$	$\boldsymbol{\lambda}'$
$\mathbf{g}''$	$\boldsymbol{\mu}''$
$\mathbf{h}''$	$\boldsymbol{\lambda}''$

The dual variables $\boldsymbol{\mu}'$, $\boldsymbol{\lambda}'$, $\boldsymbol{\mu}''$, and $\boldsymbol{\lambda}''$ are generally obtained from the solution technique for the primal problem. Dual information from the DAE system is obtained by solving the *adjoint problem* for the DAE system which has the following formulation:

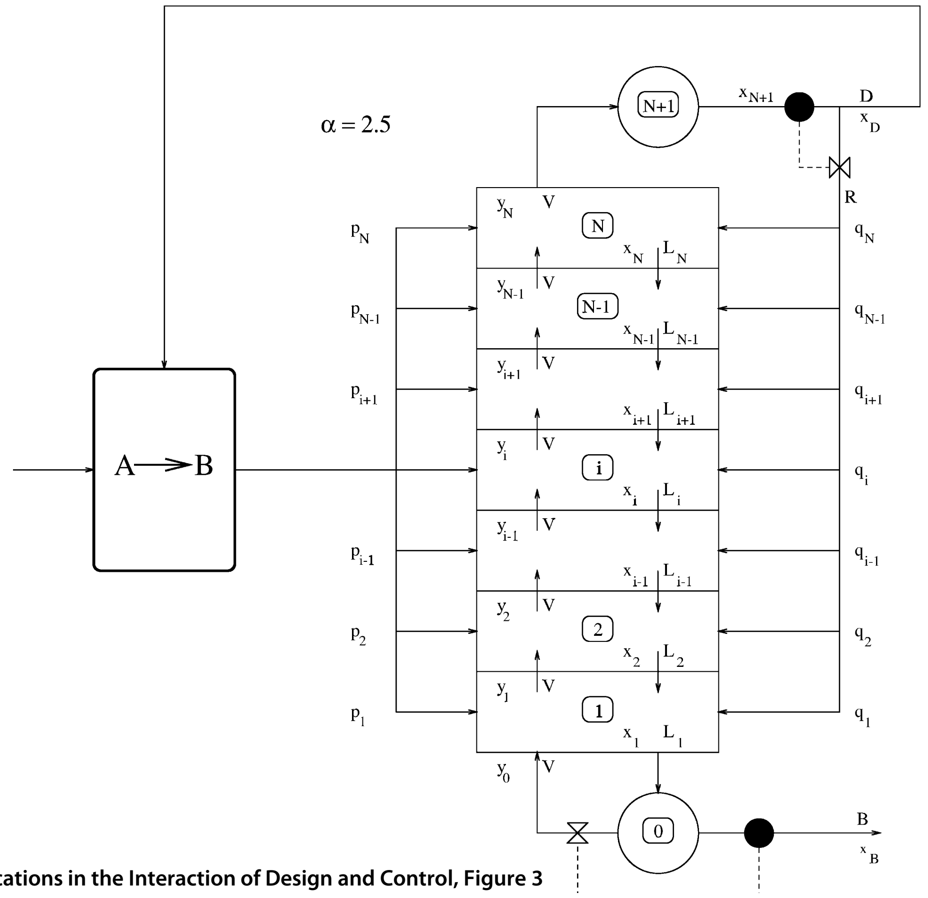
$$\begin{cases} \mathbf{p} = \boldsymbol{\nu}_1^\top \frac{d\mathbf{f}_1}{d\dot{\mathbf{z}}_1}, \\ \dot{\mathbf{p}} = \boldsymbol{\nu}_1^\top \frac{d\mathbf{f}_1}{d\mathbf{z}_1} + \boldsymbol{\nu}_2^\top \frac{d\mathbf{f}_2}{d\mathbf{z}_1}, \\ \mathbf{0} = \boldsymbol{\nu}_1^\top \frac{d\mathbf{f}_1}{d\mathbf{z}_2} + \boldsymbol{\nu}_2^\top \frac{d\mathbf{f}_2}{d\mathbf{z}_2}. \end{cases} \tag{7}$$

This is a set of DAEs where the solutions for $d\mathbf{f}_1/d\dot{\mathbf{z}}_1$, $d\mathbf{f}_1/d\mathbf{z}_1$, $d\mathbf{f}_2/d\mathbf{z}_1$, $d\mathbf{f}_1/d\mathbf{z}_2$, and $d\mathbf{f}_2/d\mathbf{z}_2$ are known functions of time obtained from the solution of the primal problem. The variables $\boldsymbol{\nu}_1(t)$ and $\boldsymbol{\nu}_2(t)$ are the adjoint variables and the solution of this problem is a backward integration in time with the following final time conditions:

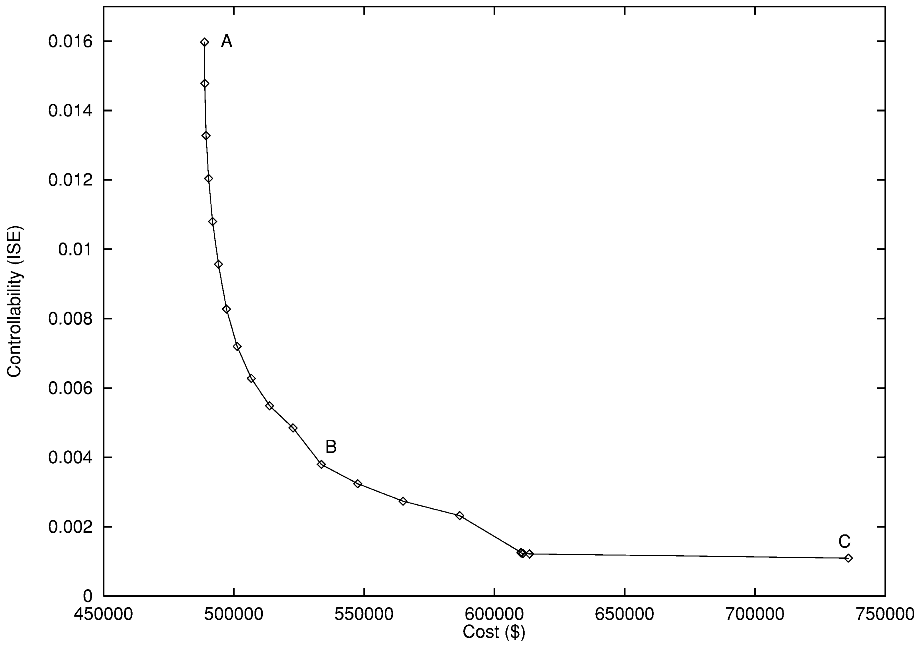
$$\frac{d\mathbf{J}}{d\mathbf{z}_1} + \boldsymbol{\lambda}' \frac{d\mathbf{h}'}{d\mathbf{z}_1} + \boldsymbol{\mu}' \frac{d\mathbf{g}'}{d\mathbf{z}_1} + \boldsymbol{\nu}_1^\top \frac{d\mathbf{f}_1}{d\dot{\mathbf{z}}_1} = 0.$$

Thus, the Lagrange multipliers for the end-time constraints are used as the final time conditions for the adjoint problem and are not included in the master problem formulation.

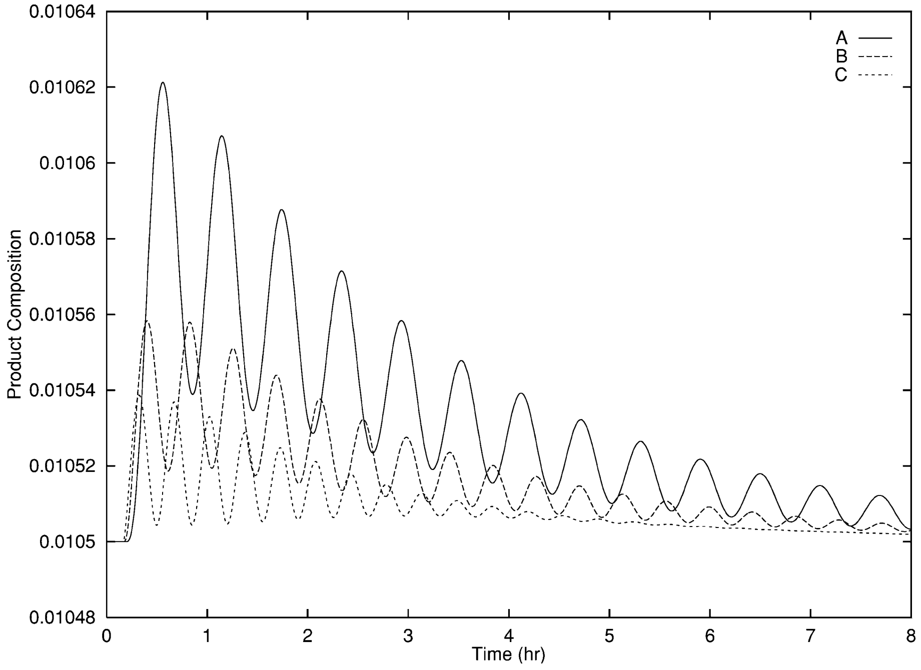
The master problem is formulated using the solution of the primal problem, $\mathbf{x}^k$ and $\mathbf{z}^k(t)$, along with



MINLP: Applications in the Interaction of Design and Control, Figure 3
Superstructure for reactor-separator-recycle system



MINLP: Applications in the Interaction of Design and Control, Figure 4
Noninferior solution set for the reactor-separator-recycle system



MINLP: Applications in the Interaction of Design and Control, Figure 5
Dynamic responses of product compositions for three designs

the dual information, $\boldsymbol{\mu}''^k$, $\boldsymbol{\lambda}''^k$, and $\boldsymbol{\nu}^k(t)$. The master problem has the following form:

$$\begin{cases} \min\limits_{\mathbf{y},\mu_b} & \mu_b \\ \text{s.t.} & \mu_b \geq J(\mathbf{x}^k,\mathbf{y}) \\ & \quad + \int_{t_0}^{t_N} \nu_1^k(t)\mathbf{f}_1(\dot{\mathbf{z}}_1^k(t),\mathbf{z}_1^k(t),\mathbf{z}_2^k(t),\mathbf{x}^k,\mathbf{y},t)\,dt \\ & \quad + \int_{t_0}^{t_N} \nu_2^k(t) \\ & \quad \mathbf{f}_2(\mathbf{z}_1^k(t),\mathbf{z}_2^k(t),\mathbf{x}^k,\mathbf{y},t)\,dt \\ & \quad + \boldsymbol{\mu}''^k\mathbf{g}''(\mathbf{x}^k,\mathbf{y}) + \boldsymbol{\lambda}''^k\mathbf{h}''(\mathbf{x}^k,\mathbf{y}), \\ & \qquad k \in K_{\text{feas}}, \\ & \mathbf{0} \geq \int_{t_0}^{t_N} \nu_1^k(t) \\ & \quad \mathbf{f}_1(\dot{\mathbf{z}}_1^k(t),\mathbf{z}_1^k(t),\mathbf{z}_2^k(t),\mathbf{x}^k,\mathbf{y},t)\,dt \\ & \quad + \int_{t_0}^{t_N} \nu_2^k(t)\mathbf{f}_2(\mathbf{z}_1^k(t),\mathbf{z}_2^k(t),\mathbf{x}^k,\mathbf{y},t)\,dt \\ & \quad + \boldsymbol{\mu}''^k\mathbf{g}''(\mathbf{x}^k,\mathbf{y}) + \boldsymbol{\lambda}''^k\mathbf{h}''(\mathbf{x}^k,\mathbf{y}), \\ & \qquad k \in K_{\text{infeas}}, \\ & \mathbf{y} \in \{0,1\}^q. \end{cases} \tag{8}$$

The integral term can be evaluated since the profiles for $\mathbf{z}^k(t)$ and $\nu^k(t)$ both are fixed and known. Note that this formulation has no restrictions on whether or not $\mathbf{y}$ variables participate in the the DAE system.

Example: Reactor-Separator-Recycle System

The example problem considered here is the design of a process involving a reaction step, a separation step, and a recycle loop. Fresh feed containing A and B flow into a an isothermal reactor where the first order irreversible reaction $A \rightarrow B$ takes place. The product from the reactor is sent to a distillation column where the unreacted A is separated from the product B and sent back to the reactor. The superstructure is shown in Fig. 3.

The model equations for the reactor (CSTR) and the separator (ideal binary distillation column) can be found in [12]. The specific problem design follows the work in [10].

For this problem, the single output is the product composition. The bottoms (product) composition is controlled by the vapor boil-up and the distillate composition is controlled by the reflux rate. Since only the product composition is specified, the distillate composition set-point is free and left to be determined through the optimization.

The cost function includes column and reactor capital and utility costs.

$$\begin{aligned}
\text{cost}_{\text{reactor}} &= 17639 D_r^{1.066} (2D_r)^{0.802}, \\
\text{cost}_{\text{column}} &= 6802 D_c^{1.066} (2.4 N_t)^{0.802} \\
&\quad + 548.8 D_c^{1.55} N_t, \\
\text{cost}_{\text{exchangers}} &= 193023 V_{ss}^{0.65}, \\
\text{cost}_{\text{utilities}} &= 72420 V_{ss}, \\
\text{cost}_{\text{total}} &= \frac{\text{cost}_{\text{reactor}} + \text{cost}_{\text{column}} + \text{cost}_{\text{exchangers}}}{\beta_{\text{pay}}} \\
&\quad + \beta_{\text{tax}} [\text{cost}_{\text{utilities}}].
\end{aligned}$$

The controllability measure is the time weighted ISE for the product composition:

$$\frac{d\mu}{dt} = t(x_B - x_B^*)^2 .$$

The noninferior solution set is shown in Fig. 4, and Table 2 lists the solution information for three of the designs in the noninferior solution set. The dynamic profile for these three designs are shown in Fig. 5.

All of the designs in the noninferior solution set are strippers. Since the feed enters at the top of the column, there is no reflux and thus no control loop for the distillate composition. The controllability of the process is increased by increasing the size of the reactor and decreasing the size of the column. The most controllable design has a large reactor and a single flash unit.

MINLP: Applications in the Interaction of Design and Control, Table 2
Solution information for three designs

Solution	A	B	C
Cost($)	489, 000	534, 000	736, 000
Capital($)	321, 000	364, 000	726, 000
Utility($)	168, 000	170, 000	10, 000
ISE	0.0160	0.00379	0.0011
Trays	19	8	1
Feed	19	8	1
V_r(kmol)	2057.9	3601.2	15000
V(kmol/hr)	138.94	141.25	85.473
K_V	90.94	80.68	87.40
τ_V (hr)	0.295	0.0898	0.0156

See also

- ▶ Chemical Process Planning
- ▶ Control Vector Iteration
- ▶ Duality in Optimal Control with First Order Differential Equations
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Optimal Control of a Flexible Arm
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control

References

1. Bahri PA, Bandoni JA, Romagnoli JA (1996) Effect of disturbances in optimizing control: Steady-state open-loop backoff problem. AIChE J 42(4):983–994

2. Brengel DD, Seider WD (1992) Coordinated design and control optimization of nonlinear processes. Comput Chem Eng 16(9):861–886
3. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307–339
4. Elliott TR, Luyben WL (1995) Capacity-based approach for the quantitative assessment of process controllability during the conceptual design stage. Industr Eng Chem Res 34:3907–3915
5. Figueroa JL, Bahri PA, Bandoni JA, Romagnoli JA (1996) Economic impact of disturbances and uncertain parameters in chemical processes – A dynamic back-off analysis. Comput Chem Eng 20(4):453–461
6. Floudas CA (1995) Nonlinear and mixed integer optimization: Fundamentals and applications. Oxford Univ. Press, Oxford
7. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10(4):237–260
8. Kocis GR, Grossmann IE (1987) Relaxation strategy for the structural optimization of process flow sheets. Industr Eng Chem Res 26(9):1869
9. Luyben ML, Floudas CA (1994) Analyzing the interaction of design and control–1. A multiobjective framework and application to binary distillation synthesis. Comput Chem Eng 18(10):933–969
10. Luyben ML, Floudas CA (1994) Analyzing the interaction of design and control–2. Reactor-separator-recycle system. Comput Chem Eng 18(10):971–994
11. Luyben ML, Luyben WL (1997) Essentials of process control. McGraw-Hill, New York
12. Luyben WL (1990) Process modeling, simulation, and control for chemical engineers, 2nd edn. McGraw-Hill, New York
13. Mohideen MJ, Perkins JD, Pistikopoulos EN (1996) Optimal design of dynamic systems under uncertainty. AIChE J 42(8):2251–2272
14. Morari M, Perkins J (1994) Design for operations. FOCAPD Conf Proc
15. Morari M, Zafiriou E (1989) Robust process control. Prentice-Hall, Englewood Cliffs, NJ
16. Narraway LT, Perkins JD (1993) Selection of control structure based on economics. Comput Chem Eng 18:S511–515
17. Narraway LT, Perkins JD (1993) Selection of process control structure based on linear dynamic economics. Industr Eng Chem Res 32:2681–2692
18. Narraway LT, Perkins JD, Barton GW (1991) Interaction between process design and process control: Economic analysis of process dynamics. J Process Control 1:243–250
19. Paules IV GE, Floudas CA (1989) APROS: Algorithmic development methodology for discrete-continuous optimization problems. Oper Res 37(6):902–915
20. Schweiger CA, Floudas CA (1997) Interaction of design and control: Optimization with dynamic models. In: Hager WW, Pardalos PM (eds) Optimal Control: Theory, Algorithms, and Applications. Kluwer, Dordrecht, pp 388–435
21. Vassiliadis VS, Sargent RWH, Pantelides CC (1994) Solution of a class of multistage dynamic optimization problems 1. Problems without path constraints. Industr Eng Chem Res 33:2111–2122
22. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer approximation method for MINLP optimization. Comput Chem Eng 14(7):769–782
23. Walsh S, Perkins JD (1996) Operability and control in process synthesis and design. In: Anderson JL (eds) Adv Chem Engin, vol 23. Acad. Press, New York, pp 301–402

MINLP: Branch and Bound Global Optimization Algorithm

IOANNIS P. ANDROULAKIS
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 90C10, 90C26

Article Outline

Keywords

Mixed integer nonlinear programming; Global optimization; Branch and bound algorithms

A wide range of nonlinear optimization problems involve integer or discrete variables in addition to continuous ones. These problem are denoted as *mixed integer nonlinear programming* (MINLP) problems. Integer variables correspond to logical decision describing whether certain actions do or do not take place, or modeling the sequence according to which those decisions take place. The nonlinear nature of the MINLP models may arise from:

- nonlinear relations in the integer domain only
- nonlinear relations in the continuous domain only
- nonlinear relations in the joint domain, i. e., products of continuous and binary/integer variables.

The general mathematical formulation of the MINLP problems can be stated as follows:

$$\begin{cases} \min\limits_{x,y} & f(x,y) \\ \text{s.t.} & h(x,y) = 0 \\ & g(x,y) \leq 0 \\ & x \in X \subset \mathbb{R}^n \\ & y \in Y \quad \text{(integer)}. \end{cases}$$

Here, x represents a vector of n continuous variables, y is a vector of integer variables, $f(x, y)$, $h(x, y)$, $g(x, y)$ represent the objective function, equality and inequality constraints, respectively. It should be noted, that every problem of the form just presented, can be transformed into one where all integer variables have been transformed into binary, i. e., 0–1, variables, by realizing that every integer $y^L \leq y \leq y^U$ can be expressed through 0–1 variables, $z = (z_1, \ldots, z_N)$, as:

$$y = y^L + z_1 + 2z_2 + 4z_3 + \cdots + 2^{N-1} z_N ,$$
$$N = 1 + \text{INT} \left\{ \frac{\log(y^U - y^L)}{\log 2} \right\} .$$

Therefore, any MINLP problem can be written as:

$$\begin{cases} \min\limits_{x,y} & f(x,y) \\ \text{s.t.} & h(x,y) = 0 \\ & g(x,y) \leq 0 \\ & x \in X \subset \mathbb{R}^n \\ & y \in Y = \{0,1\}^m . \end{cases}$$

In the analysis of MINLP problems two issues are of paramount importance:

- combinatorial explosion of computational requirements as the number of binary variables increases
- *NP*-hard nature of the problem of determining the *global minimum solution* of general nonconvex MINLP problems.

A complexity analysis of the former is presented in [16], while the complexity of determining global minimum solutions of MINLPs is discussed in [15].

Various methods exist for identifying a locally optimum solution of MINLP problems. These are discussed in great detail in [9] and in a recent thorough review paper, [6], which presents a comprehensive account of the various approaches for addressing issues related to the solution of mixed integer nonlinear optimization problems.

The main objective in a general branch and bound algorithm is to perform an enumeration of the alternatives without examining all 0–1 combinations of the binary variables. A key element in such an enumeration if the representation of alternatives via a *binary tree*. The basic ideas in a branch and bound algorithm are the following. First, a reasonable effort is made in solving the original problem, by considering for instance the continuous relaxation of it. If the relaxation does not result in an integer-feasible solution, i. e., one in which the binary variables achieve 0–1 at the optimal point, them the root node is separated into two candidate subproblems which are subsequently solved. The separation aims at creating simpler instances of the original problem. Until the problem is successfully solved this process of generating candidate subproblems is repeated. Branch and bound algorithms are also known as *divide-and-conquer* for that very reason. A basic principle common to all branch and bound algorithms is that the solution of the subproblems aims at generating *valid lower bounds* on the original MINLP through its relaxation to a continuous problem. The relaxation, in the case of MINLP, results in a nonlinear programming problem (NLP) which, in the general case, is nonconvex and needs to be solved to global optimality so as to provide a valid lower bound. If the NLP relaxation renders an integer solution, then this solution is referred to as *valid upper bound*. The generation of the sequence of valid upper and lower bounds is called *bounding step*. The way subproblems are created is by forcing some of the binary variables to take on a value of 0 or 1. This is known as the *branching step*. Nodes in the tree are pruned when the corresponding valid lower bound exceeds the valid upper bound, this stage is know as the *fathoming step*. The selection of the branching node, the branching variable and the generation of the lower bound are very crucial steps whose importance becomes even more pronounced when addressing nonconvex MINLP problems. Two basic strategies exists regarding the selection of the branching node depending on whether one designs a branch and bound based on a *depth-first* or a *breadth-first* approach. In the former, the last node created is selected for branching, in the latter the node that generated the best lower bound is selected. It is not clear which strategy is the best and it is often that the one that minimizes the computational requirement is selected, [13]. Another alternative is to select nodes based on the deviation of the solution from integrality, [12]. The most common strategy for selecting a branching variable is to select the variable whose value at the solution of some relaxed problem is the farthest from integer, i. e., the most fractional variable, [17]. In [12] a method based on the concept of *pseudocosts* which quantifies the effect of binary variables is also proposed, which assigns essentially priorities on the order of branching variables. Finally, one of the most important computational step is the generation of the lower bound, in other words the solution of the relaxed problem. The effectiveness of a branch and bound depends of the quality of the lower bound that is generated. At every node of the branch and bound tree a nonlinear-nonconvex NLP is solved. Two issues are important: the lower bound must be valid, in other words the relaxation at a particular node must underestimate the solution of the original problem for this node, and the lower bounds must be tight so as to enhance the fathoming step. The key complexity when dealing with nonconvex MINLPs is that the relaxation solved at each node is, of course, a nonconvex NLP that has to be solved to global optimality. With the exception of problems which are convex in the x and relaxed y-space for which variants of the branch and bound algorithms will lead the correct solution, [18], in all other cases *global optimization* algorithms have to be employed for the generation of valid lower bounds.

In [19] the scope of branch and bound algorithms was extended to problems for which valid convex underestimating NLPs can be constructed for the convex relaxations. The problems included bilinear and separable problems for which convex underestimators can be build [14]. A number of very useful tests were proposed to accelerate the reduction of solution space. Namely:

1) Optimality based range reduction tests: For the first set of tests, an upper bound U on the nonconvex MINLP must be computed and a convex lower bounding NLP must be solved to obtain a lower bound L. If a bound constraint for variable x_i, with $x_i^L \leq x_i \leq x_i^U$, is active at the solution of the convex NLP and has multiplier $\lambda_i^* > 0$, the bounds on x_i can be updated as follows:
   a) If $x_i - x_i^U = 0$ at the solution of the convex NLP and $\kappa_i = x_i^U - (U - L)/\lambda_i^*$ is such that $\kappa_i > x_i^L$, then $x_i^L = \kappa_i$.
   b) If $x_i - x_i^L = 0$ at the solution of the convex NLP and $\kappa_i = x_i^L + (U - L)/\lambda_i^*$ is such that $\kappa_i < x_i^U$, then $x_i^U = \kappa_i$.

   If neither bound constraint is active at the solution of the convex NLP for some variable x_j, the problem can be solved by setting $x_j = x_j^U$ or $x_j = x_j^L$. Tests similar to those presented above are then used to update the bounds on x_j.
2) Feasibility based range reduction tests: In addition to ensuring that tight bounds are available for the variables, the constraint underestimators are used to generate new constraints for the problem. Consider the constraint $g_i(x, y) \leq 0$. If its underestimating function $\underline{g}_i(x, y) = 0$ at the solution of the convex NLP and its multiplier is $\mu_i^* > 0$, the constraint
$$\underline{g}_i(x, y) \geq -\frac{U - L}{\mu_i^*}$$
   can be included in subsequent problems.

A global optimization algorithm branch and bound algorithm has been proposed in [20]. It can be applied to problems in which the objective and constraints are functions involving any combination of binary arithmetic operations (addition, subtraction, multiplication and division) and functions that are either concave over the entire solution space (such as ln) or convex over this domain (such as exp).

The algorithm starts with an automatic reformulation of the original nonlinear problem into a problem that involves only linear, bilinear, linear fractional, simple exponentiation, univariate concave and univariate convex terms. This is achieved through the introduction of new constraints and variables. The reformulated problem is then solved to global optimality using a branch and bound approach. Its special structure allows the construction of a convex relaxation at each node of the tree. The integer variables can be handled in two ways during the generation of the convex lower bounding problem. The integrality condition on the variables can be relaxed to yield a convex NLP which can then be solved globally. Alternatively, the integer variables can be treated directly and the convex lower bounding MINLP can be solved using a branch and bound algorithm as described earlier. This second approach is more computationally intensive but is likely to result in tighter lower bounds on the global optimum solution. In order to obtain an upper bound for the optimum solution, several methods have been suggested. The MINLP can be transformed to an equivalent nonconvex NLP by relaxing the integer variables. For example, a variable $y \in \{0,1\}$ can be replaced by a continuous variable $z \in [0,1]$ by including the constraint $z - z \cdot z = 0$. The nonconvex NLP is then solved locally to provide an upper bound. Finally, the discrete variables could be fixed to some arbitrary value and the nonconvex NLP solved locally.

In [1] *SMIN* was proposed which is designed to address the following class of problems to global optimality:

$$\begin{cases} \min & f(x) + x^\top A_0 y + c_0^\top y \\ \text{s.t.} & h(x) + x^\top A_1 y + c_1^\top y = 0 \\ & g(x) + x^\top A_2 y + c_2^\top y \leq 0 \\ & x \in X \subset \mathbb{R}^n \\ & y \in Y \quad \text{(integer)}, \end{cases}$$

where $c_0^\top$, $c_1^\top$ and $c_2^\top$ are constant vectors, A_0, A_1 and A_2 are constant matrices and $f(x)$, $h(x)$ and $g(x)$ are functions with continuous second order derivatives. The solution strategy is an extension of the αBB algorithm for twice-differentiable NLPs [4,5,7]. It is based on the generation of two converging sequences of upper and lower bounds on the global optimum solution. A rigorous underestimation and convexification strategy for functions with continuous second order derivatives allows the construction of a lower bounding MINLP problem with convex functions in the continuous variables. If no mixed-bilinear terms are present ($A_i = 0, \forall i$), the resulting MINLP can be solved to global optimality using the *outer approximation algorithm* (OA), [8]. Otherwise, the *generalized Benders decomposition* (GBD) can be used, [10], or the Glover transformations [11] can be applied to remove these bilinearities and permit the use of the OA algorithm. This convex MINLP provides a valid lower bound on the original MINLP. An upper bound on the problem can be obtained by applying the OA algorithm or the GBD to find a local solution. This bound generation strategy is incorporated within a branch and bound scheme: a lower and upper bound on the global solution are first obtained for the entire solution space. Subsequently, the domain is subdivided by branching on a binary or a continuous variable, thus creating new nodes for which upper and lower bounds can be computed. At each iteration, the node with the lowest lower bound is selected for branching. If the lower bounding MINLP for a node is infeasible or if its lower bound is greater than the best upper bound, this node is fathomed. The algorithm is terminated when the best lower and upper bound are within a pre-specified tolerance of each other.

Before presenting the algorithmic procedure, an overview of the underestimation and convexification strategy is given, and some of the options available within the algorithm are discussed.

In order to transform the MINLP problem of the form just described into a convex problem which can be solved to global optimality with the OA or GBD algorithm, the functions $f(x)$, $h(x)$ and $g(x)$ must be convexified. The underestimation and convexification strategy used in the αBB algorithm has previously been described in detail [3,4,5]. Its main features are exposed here.

In order to construct as tight an underestimator as possible, the nonconvex functions are decomposed into a sum of convex, bilinear, univariate concave and general nonconvex terms. The overall function underestimator can then be built by summing up the convex underestimators for all terms, according to their type. In particular, a new variable is introduced to replace each bilinear term, and is bounded by its convex envelope. The univariate concave terms are linearized. For each nonconvex term $nt(x)$ with Hessian matrix $H_{nt}(x)$, a convex underestimator $L(x)$ is defined as

$$L(x) = nt(x) - \sum_i \alpha_i (x_i^U - x_i)(x_i - x_i^L), \qquad (1)$$

where x_i^U and x_i^L are the upper and lower bounds on variable x_i, respectively, and the α parameters are nonnegative scalars such that $H_{nt}(x) + 2\,\mathrm{diag}(\alpha_i)$ is positive semidefinite over the domain $[x^L, x^U]$. The rigorous computation of the α parameters using interval Hessian matrices is described in [3,4,5].

The underestimators are updated at each node of the branch and bound tree as their quality strongly depends on the bounds on the variables. An unusual feature of the SMIN-αBB algorithm is the strategy used to select branching variables. It follows a hybrid approach where branching may occur both on the integer and the continuous variables in order to fully exploit the structure of the problem being solved. After the node with the lowest lower bound has been identified for branching, the type of branching variable must be determined according to one of the following two criteria:

1) Branch on the binary variables first.
2) Solve a continuous relaxation of the nonconvex MINLP locally. Branch on a binary variable with a low degree of fractionality at the solution. If there is no such variable, branch on a continuous variable.

The first criterion results in the creation of an integer tree for the first q levels of the branch and bound tree, where q is the number of binary variables. At the lowest level of this integer tree, each node corresponds to a nonconvex NLP and the lower and upper bounding problems at subsequent levels of the tree are NLP problems. The efficiency of this strategy lies in the minimization of the number of MINLPs that need to be solved. The combinatorial nature of the problem and its nonconvexities are handled sequentially. If branching occurs on a binary variable, the selection of that variable can be done randomly or by solving a relaxation of the nonconvex MINLP and choosing the most fractional variable at the solution.

The second criterion selects a binary variable for branching only if it appears that the two newly created nodes will have significantly different lower bounds.Thus, if a variable is close to integrality at the solution of the relaxed problem, forcing it to take on a fixed value may lead to the infeasibility of one of the nodes or the generation of a high value for a lower bound, and therefore the fathoming of a branch of the tree. If no binary variable is close to integrality, a continuous variable is selected for branching.

A number of rules have been developed for the selection of a continuous branching variable. Their aim is to determine which variable is responsible for the largest separation distances between the convex underestimating functions and the original nonconvex functions. These efficient rules are exposed in [2]. Variable bound updates performed before the generation of the convex MINLP have been found to greatly enhance the speed of convergence of the αBB algorithm for continuous problems [2]. For continuous variables, the variable bounds are updated by minimizing or maximizing the chosen variable subject to the convexified constraints being satisfied. In spite of its computational cost, this procedure often leads to significant improvements in the quality of the underestimators and hence a noticeable reduction in the number of iterations required.

In addition to the update of continuous variable bounds, the SMIN-αBB algorithm also relies on binary variable bound updates. Through simple computations, an entire branch of the branch and bound tree may be eliminated when a binary variable is found to be restricted to 0 or 1. The bound update procedure for a given binary variable is as follows:

1) Set the variable to be updated to one of its bounds $y = y_B$.
2) Perform interval evaluations of all the constraints in the nonconvex MINLP, using the bounds on the solution space for the current node.
3) If any of the constraints are found infeasible, fix the variable to $y = 1 - y_B$.
4) If both bounds have been tested, repeat this procedure for the next variable to be updated. Otherwise, try the second bound.

In [1] *GMIN*, which operates within a classical branch and bound framework, was proposed. The main difference with similar branch and bound algorithms [12,17] is its ability to identify the global optimum solution of a much larger class of problems of the form

$$\begin{cases} \min\limits_{x,y} & f(x, y) \\ \text{s.t.} & h(x, y) = 0 \\ & g(x, y) \leq 0 \\ & x \in X \subset \mathbb{R}^n \\ & y \in N^q , \end{cases}$$

where N is the set of nonnegative integers and the only condition imposed on the functions $f(x, y)$, $g(x, y)$ and $h(x, y)$ is that their continuous relaxations possess continuous second order derivatives. This increased applicability results from the use of the αBB global optimization algorithm for continuous twice-differentiable NLPs [4,5,7].

At each node of the branch and bound tree, the nonconvex MINLP is relaxed to give a nonconvex NLP, which is then solved with the αBB algorithm. This allows the identification of rigorously valid lower bounds and therefore ensures convergence to the global optimum. In general, it is not necessary to let the αBB algorithm run to completion as each one of its iterations generates a lower bound on global solution of the NLP being solved. A strategy of early termination leads to a reduction in the computational requirements of each node of the binary branch and bound tree and faster overall convergence.

The GMIN-αBB algorithm selects the node with the lowest lower bound for branching at every iteration. The branching variable selection strategy combines several approaches: branching priorities can be specified for some of the integer variables. When no variable has a priority greater than all other variables, the solution of the continuous relaxation is used to identify either the most fractional variable or the least fractional variable for branching.

Other strategies have been implemented to ensure a satisfactory convergence rate. In particular, bound updates on the integer variables can be performed at each level of the branch and bound tree. These can be carried out through the use of interval analysis. An integer variable, y^*, is fixed at its lower (or upper) bound and the range of the constraints is evaluated with interval arithmetic, using the bounds on all other variables. If the range of any constraint is such that this constraint is violated, the lower (or upper) bound on variable y^* can be increased (or decreased) by one. Another strategy for bound updates is to relax the integer variables, to convexify and underestimate the nonconvex constraints and to minimize (or maximize) a variable y^* in this convexified feasible region. The resulting lower (or upper) bound on relaxed variable y^* can then be rounded up (or down) to the nearest integer to provide an updated bound for y^*.

See also

- ▶ αBB Algorithm
- ▶ Chemical Process Planning
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Disjunctive Programming
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ Global Optimization in Batch Design Under Uncertainty
- ▶ Global Optimization in Generalized Geometric Programming
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization in Phase and Chemical Reaction Equilibrium
- ▶ Interval Global Optimization
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming
- ▶ Reformulation-linearization Methods for Global Optimization
- ▶ Smooth Nonlinear Nonconvex Optimization

References

1. Adjiman CS, Androulakis IP, Floudas CA (1997) Global optimization of MINLP probelms in process synthesis and design. Comput Chem Eng 21:S445–S450
2. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, αBB for twice-differentiable NLP's – II. Implementation and computational results. Comput Chem Eng 22:1137–1158
3. Adjiman CS, Androulakis IP, Maranas CD, Floudas CA (1996) A global optimization method, αBB, for process design. Comput Chem Eng 20:S419–S424
4. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, αBB for twice-differentiable NLP's – I. Theoretical Advances. Comput Chem Eng 22:1159–1179
5. Adjiman CS, Floudas CA (1996) Rigorous convex underestimators for general twice-differentiable problems. J Global Optim 9:23–40
6. Adjiman CS, Schweiger CA, Floudas CA (1998) Mixed-integer nonlinear optimization in process synthesis. In: Du D-Z and Pardalos PM (eds) Handbook Combinatorial Optim. Kluwer, Dordrecht

7. Androulakis IP, Maranas CD, Floudas CA (1995) αBB, a global optimization method for general constrained nonconvex problems. J Global Optim 7:337–363
8. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307–339
9. Floudas CA (1995) Nonlinear and mixed-integer optimization: Fundamentals and applications. Oxford Univ. Press, Oxford
10. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
11. Glover F (1975) Improved linear integer programming formulations of nonlinear integer problems. Managem Sci 22:445–452
12. Gupta OK, Ravindran R (1985) Branch and bound experiments in convex nonlinear integer programming. Managem Sci 31:1533–1546
13. Lawler EL, Wood DE (1966) Branching and bound methods: A survey. Oper Res 14:699–719
14. McCormick GP (1976) Computatbility of global solutions to factorable nonconvex programs; Part I – convex underestimating problems. Math Program 10:147–175
15. Murty KG, Kabadi SN (1987) Some NP-complete problems in quadratic and nonlinear prgramming. Math Program 39:117–123
16. Neumhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
17. Ostrovsky GM, Mikhailov GW (1990) Discrete optimization of chemical processes. Comput Chem Eng 14:111–124
18. Quesada I, Grossmann IE (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Comput Chem Eng 16:937–947
19. Ryoo HS, Sahinidis NV (1995) Global optimization of nonconvex NLPs and MINLPs with applications in process design. Comput Chem Eng 19:551–566
20. Smith EMB, Pantelides CC (1997) Global optimization of nonconvex MINLPs. Comput Chem Eng 21:S333–S338

MINLP: Branch and Bound Methods

Brian Borchers
Department of Mathematics, New Mexico Tech., Socorro, USA

MSC2000: 90C11

Article Outline

Keywords

Mixed integer programming; Branch and bound; MINLP

A general *mixed integer nonlinear programming problem* (*MINLP*) can be written as

$$(\text{MINLP})\quad \begin{cases} \min & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = 0 \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \le 0 \\ & \mathbf{x} \in \mathbb{R}^n \\ & \mathbf{y} \in \mathbb{Z}^m. \end{cases}$$

Here $\mathbf{x}$ is a vector of n continuous variables and $\mathbf{y}$ is a vector of m *integer variables*. In many cases, the integer variables $\mathbf{y}$ are restricted to the values 0 and 1. Such variables are called *binary variables*. The function f is a scalar valued objective function, while the vector functions $\mathbf{h}$ and $\mathbf{g}$ express linear or nonlinear constraints. Problems of this form have a wide variety of applications, in areas as diverse as IR spectroscopy [6], finance [3], chemical process synthesis [9], topological design of transportation networks [12], and marketing [10].

The earliest work on *branch and bound* algorithms for mixed integer linear programming dates back to the early 1960s [7,13,15]. Although the possibility of applying branch and bound methods to mixed integer nonlinear programming problems was apparent from the beginning, actual work on such problems did not begin until later. Early papers on branch and bound algorithms for mixed integer nonlinear programming include [11,14].

A branch and bound algorithm for solving (MINLP) requires the following data structures. The algorithm maintains a list L of unsolved subproblems. The algorithm also maintains a record of the best integer solution that has been found. This solution, ($\mathbf{x}^*$, $\mathbf{y}^*$), is called the incumbent solution. The incumbent solution provides an upper bound, ub, on the objective value of an optimal solution to (MINLP).

The basic branch and bound procedure is as follows.

1) *Initialize*: Create the list L with (MINLP) as the initial subproblem. If a good integer solution is known,

then initialize $\mathbf{x}^*$, $\mathbf{y}^*$, and *ub* to this solution. If there is no incumbent solution, then initialize *ub* to $+\infty$.

2) *Select*: Select an unsolved subproblem, *S*, from the list *L*. If *L* is empty, then stop: If there is an incumbent solution, then that solution is optimal; If there is no incumbent solution, then (MINLP) is infeasible.
3) *Solve*: Relax the integrality constraints in *S* and solve the resulting nonlinear programming relaxation. Obtain a solution $\widehat{\mathbf{x}}$, $\widehat{\mathbf{y}}$, and a lower bound, *lb*, on the optimal value of the subproblem.
4) *Fathom*: If the relaxed subproblem was infeasible, then *S* will clearly not yield a better solution to (MINLP) than the incumbent solution. Similarly, if $lb \geq ub$, then the current subproblem cannot yield a better solution to (MINLP) than the incumbent solution. Remove *S* from *L*, and return to step 2.
5) *Integer Solution*: If $\widehat{\mathbf{y}}$ is integer, then a new incumbent integer solution has been obtained. Update $\mathbf{x}^*$, $\mathbf{y}^*$, and *ub*. Remove *S* from *L* and return to step 2.
6) *Branch*: At least one of the integer variables y_k takes on a fractional value in the solution to the current subproblem. Create a new subproblem, S_1 by adding the constraint

$$y_k \leq \lfloor \widehat{y}_k \rfloor .$$

Create a second new subproblem, S_2 by adding the constraint

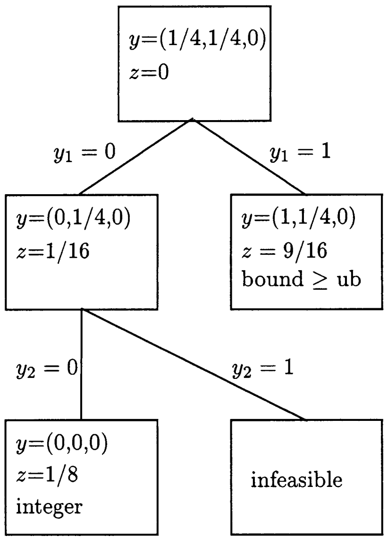
$$y_k \geq \lceil \widehat{y}_k \rceil .$$

Remove *S* from *L*, add S_1 and S_2 to *L*, and return to step 2.

The following example demonstrates how the branch and bound algorithm solves a simple (MINLP):

$$\begin{cases} \min & (y_1 - \frac{1}{4})^2 + (y_2 - \frac{1}{4})^2 + y_3^2 \\ & -2y_1 + 2y_2 \leq 1 \\ & \mathbf{y} \text{ binary.} \end{cases}$$

The optimal solution to the initial nonlinear programming relaxation is $y = (1/4, 1/4, 0)$, with an objective value of $z = 0$. Both y_1 and y_2 take on fractional values in this solution, so it is necessary to select a branching variable. The algorithm arbitrarily selects y_1 as the branching variable, and creates two new subproblems in which y_1 is fixed at 0 or 1. In the subproblem with y_1 fixed at 0, the optimal solution is $y = (0, 1/4, 0)$, with $z = 1/16$. Since the optimal value of y_2 is fractional, the algorithm again creates two new subproblems, with y_2 fixed at 0 and 1. The optimal solution to the subproblem with $y_1 = 0$ and $y_2 = 0$ is $y = (0, 0, 0)$, with $z = 1/8$. This establishes an incumbent integer solution. The subproblem with $y_1 = 0$ and $y_2 = 1$ is infeasible and can be eliminated from consideration. The subproblem with $y_1 = 1$ has an optimal solution with $y = (1, 1/4, 0)$ and objective value $z = 9/16$. Since 9/16 is larger than the objective value of the incumbent solution, this subproblem can be eliminated from consideration. Thus the optimal solution to the example problem is $y^* = (0, 0, 0)$ with objective value $z^* = 1/8$.



MINLP: Branch and Bound Methods, Figure 1
Branch and bound tree for a sample problem

Since each subproblem *S* creates at most two new subproblems, the set of subproblems considered by the branch and bound algorithm can be represented as a binary tree. The above figure shows the branch and bound tree for the example problem.

There are a number of important issues in the implementation of a branch and bound algorithm for (MINLP).

The first important issue is how to solve the nonlinear programming relaxations of the subproblems in step 3. If the objective function *f* and the constraint functions **g** are convex, while the constraint functions **h** are linear, then the nonlinear programming subproblems in step 3 are convex and thus relatively easy to solve. A variety of methods have been used to solve these subproblems including generalized reduced gradient (GRG) methods [11], sequential quadratic programming (SQP) [4], active set methods for quadratic programming [8], and interior point methods [16].

However, if the nonlinear programming subproblems are nonconvex, then it can be extremely difficult to solve the nonlinear programming relaxation of S or even obtain a lower bound on the optimal objective function value. For some specialized classes of nonconvex optimization problems, including *indefinite quadratic programming*, *bilinear programming*, and *fractional linear programming*, convex functions which underestimate the nonconvex objective function are known. These *convex underestimators* are widely used in branch and bound algorithms for nonconvex nonlinear programming problems. Branch and bound techniques for nonconvex continuous optimization problems can also been used within a branch and bound algorithm for nonconvex mixed integer nonlinear programming problems. For instance, the *BARON* system uses this approach to solve a variety of nonconvex mixed integer nonlinear programming problems [17,18]. This approach is also used in the GMIN-αBB algorithm to solve nonconvex $0-1$ mixed integer nonlinear programming problems with twice differentiable objective and constraint functions [1].

The choice of the next subproblem to be solved in step 2 can have a significant influence on the performance of the branch and bound algorithm. In mixed integer linear programming, a variety of heuristics are employed to select the next subproblem [2]. One popular heuristic used in branch and bound algorithms for MILP is the *'best bound rule'*, in which the subproblem with the smallest lower bound is selected. The best bound rule is widely used within branch and bound algorithms for (MINLP) [4,11,18]

In step 6, there may be a choice of several variables with fractional values to be the branching variable. A simple approach is to select the variable whose value $\widehat{y}_k$ is furthest from being an integer [4,11]. In mixed integer linear programming, estimates of the increase in the objective function that will result from forcing a variable to an integer value are often made. These estimates, called *'pseudocosts'* or *'penalties'*, are used to select the branching variable. Penalties have also been used in branch and bound algorithms for mixed integer nonlinear programming problems [11,18].

The performance of the branch and bound algorithm can be improved by computing lower bounds on the optimal value of a subproblem without actually solving the subproblem. In [8], lower bounds on the optimal objective value of a subproblem are derived from an optimal dual solution to the subproblem's parent problem. If this lower bound is larger than the objective value of the incumbent solution, then the subproblem can be eliminated from consideration. In [4], Lagrangian duality is used to compute lower bounds during the solution of a subproblem. When the lower bound exceeds the value of the incumbent solution, the current subproblem can be discarded.

Another way to improve the performance of a branch and bound algorithm for (MINLP) is to tighten the formulation of the nonlinear programming subproblems before solving them. In the BARON package, dual information from the solution to a nonlinear programming subproblem is used to restrict the ranges of variables and constraints in the children of the subproblem [17,18].

In *branch and cut* approaches, constraints called cutting planes are added to the nonlinear programming subproblems [3,19]. These additional constraints are selected so that they reduce the size of the feasible region of nonlinear programming subproblems without eliminating any integer solutions from consideration. This tightens the formulations of the subproblems and thus increases the probability that a subproblem can be fathomed by bound. Furthermore, the use of cutting planes can make it more likely that an integer solution will be obtained early in the branch and bound process. A variety of cutting planes developed for use in branch and cut algorithms for integer linear programming have been adapted for use in branch and cut algorithms for nonlinear integer programming. These include *mixed integer rounding cuts* [3], *knapsack cuts* [3], *intersection cuts* [3], and *lift-and-project cuts* [19].

To date, little work has been done to compare the performance of branch and bound methods for (MINLP) with other approaches such as *outer approximation* and *generalized Benders decomposition*. B. Borchers and J.E. Mitchell (1997) compared an experimental branch and bound code with a commercially available outer approximation code on a number of test problems [5]. This study found that the branch and bound code and outer approximation code were roughly comparable in speed and robustness. R. Fletcher and S. Leyffer (1998) compared the performance of their branch and bound code for mixed integer convex quadratic programming problems with their implementations of outer approximation, generalized Benders decomposition, and an algorithm that combines branch and bound and outer approximation approaches [8]. Fletcher and Leyffer found that their branch and bound solver was consistently faster than the other codes by about an order of magnitude.

See also

- ▶ Chemical Process Planning
- ▶ Disjunctive Programming
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming
- ▶ Reformulation-linearization Methods for Global Optimization

References

1. Adjiman CS, Androulakis IP, Floudas CA (1997) Global optimization of MINLP problems in process synthesis and design. Comput Chem Eng 21(1001):S445–S450
2. Beale EML (1977) Integer programming. In: Jacobs D (ed) The State of the Art in Numerical Analysis. Acad. Press, New York p 409–448
3. Bienstock D (1996) Computational study of a family of mixed-integer quadratic programming problems. Math Program 74(2):121–140
4. Borchers B, Mitchell JE (1994) An improved branch and bound algorithm for mixed integer nonlinear programs. Comput Oper Res 21(4):359–367
5. Borchers B, Mitchell JE (1997) A computational comparison of branch and bound and outer approximation algorithms for 0–1 mixed integer nonlinear programs. Comput Oper Res 24(8):699–701
6. Brink A, Westerlund T (1995) The joint problem of model structure determination and parameter estimation in quantitative IR spectroscopy. Chemometrics and Intelligent Laboratory Systems 29:29–36
7. Dakin RJ (1965) A tree–search algorithm for mixed integer programming problems. Comput J 8:250–255
8. Fletcher R, Leyffer S (1998) Numerical experience with lower bounds for MIQP branch-and-bound. SIAM J Optim 8:604–616
9. Floudas CA (1995) Nonlinear and mixed-integer optimization: fundamentals and applications. Oxford Univ. Press, Oxford
10. Gavish B, Horsky D, Srikanth K (1983) An approach to the optimal positioning of a new product. Managem Sci 29(11):1277–1297
11. Gupta OK, Ravindran A (1985) Branch and bound experiments in convex nonlinear integer programming. Managem Sci 31(12):1533–1546
12. Hoang Hai Hoc (1982) Topological optimization of networks: A nonlinear mixed integer model employing generalized Benders decomposition. IEEE Trans Autom Control 27:164–169
13. Land A, Doig A (1960) An automatic method of solving discrete programming problems. Econometrika 28(3):497–520
14. Laughhunn DJ (1970) Quadratic binary programming with applications to capital-budgeting problems. Oper Res 18(3):454–461
15. Lawler EL, Wood DE (1966) Branch and bound methods: A survey. Oper Res 14(4):699–719
16. Lee EK, Mitchell JE (1997) Computational experience of an interior point algorithm in a parallel branch-and-cut framework. In: Proc. Eighth SIAM Conf. Parallel Processing for Sci. Computing, Minneapolis, March 1997. www.siam.org/catalog/mcc07/heath97.htm
17. Ryoo HS, Sahinidis NV (1996) A branch–and–reduce approach to global optimization. J Global Optim 8(2):107–139
18. Sahinidis NV (1996) BARON: A general purpose global optimization software package. J Global Optim 8(2):201–205

19. Stubbs RA, Mehrotra S (1999) A branch-and-cut method for 0–1 mixed convex programming. Math Program 80:515–532

MINLP: Design and Scheduling of Batch Processes

CHRISTODOULOS A. FLOUDAS[1], S. T. HARDING[2], MARIANTHI IERAPETRITOU[3]
[1] Department of Chemical Engineering, Princeton University, Princeton, USA
[2] Advanced Process Combinatorics, Inc., Purdue Technology Center, West Lafayette, USA
[3] Department Chemical and Biochemical Engineering, Rutgers University, Piscataway, USA

MSC2000: 90C26

Article Outline

Keywords

Batch process; Design; Scheduling; Continuous and discrete time models

The design of batch processes has been a major area of research for the past several decades. In conjunction with the design of batch plants, many different approaches have been proposed for the determination of an optimal schedule for the plant. It has been recognized for some time that in order to increase the efficiency of batch processes, the two tasks of design and scheduling should be considered simultaneously.

The problem is to design a batch process consisting of M processing steps, in which N products are made, where all materials follow the same path through the process. This is commonly known as a *multiproduct batch plant*, or a *flow-shop*.

There are two predominant methods for formulating the batch process design and scheduling problem. The first is a *continuous-time formulation* in which the scheduling information is incorporated through a *planning horizon* constraint. This problem can be formulated as a NLP or MINLP depending on whether the number of parallel units is fixed or variable. The solution of this problem does not give the actual schedule, but does guarantee that a feasible schedule exists. A separate problem, typically a MILP, must be solved to find the actual schedule.

The second method for formulating the batch process design and scheduling problem is based on a *state-task-network* (STN) representation. In this approach, the planning horizon is discretized into time steps. Each task must be assigned to both a unit and a time slot. The formulation results in a large MINLP whose solution provides both the plant design and the actual schedule.

Continuous-Time Formulations

The early work of [10] was based on the *single product campaign* (SPC) scheduling policy. In a single product campaign, all batches of one product are processed one after the other, followed by all of the batches of the next product, and so on.

In this approach, the scheduling information is incorporated by way of a planning horizon constraint. This constraint requires that all products must be completed before the planning horizon, H, is reached. In a single product campaign, the time between batches of product i is based on the maximum processing time over all of the stages,

$$t_{Li} = \max_j(t_{ij}),$$

where t_{Li} is the 'limiting' time for product i. The planning horizon constraint can be written as the sum over all of the products of the limiting time multiplied by the number of batches of each product

$$\sum_i \frac{Q_i}{B_i} T_{Li} \leq H,$$

where Q_i is the total production of i and B_i is the batch size for i. Because Q_i and B_i are variables, this results in a NLP.

In [4] the authors formulated the batch process design and scheduling problem as a MINLP. Their model was based on the SPC model of [10]. In this problem, more than one piece of equipment per stage is available for use in parallel. Rather than solve the MINLP rigorously, they relaxed the number of units per stage to be continuous and solved the resulting NLP. [5] formulated the MINLP using binary 0-1 variables and solved it with an outer approximation method. In addition to the combinatorial nature of the problem due to integer variables, the solution of the problem is complicated by the nonconvex form of the planning horizon constraint.

[2] developed extensions of the SPC formulation to allow more efficient utilization of the batch process equipment. They considered two *mixed-product campaign* (MPC) scheduling policies,

i) the *unlimited intermediate storage* (UIS) policy; and
ii) the *zero-wait* (ZW) policy.

As its name implies, a mixed product campaign allows batches of different products to be processed sequentially. For example, a SPC schedule for three batches each of two products A and B would be, AAABBB, while a MPC schedule could be ABABAB. In the zero-wait policy, when a product has completed processing in one stage, it must immediately begin processing in the next stage. Conversely, the UIS policy allows a product to be stored for a period of time before beginning the next processing step. [7] showed that for the case of zero cleanup times, the UIS policy is the most efficient mixed-product campaign policy, while the ZW policy is the most conservative. [2] incorporated the new scheduling policies into the batch process design problem by considering the characteristic cycle time for each policy. The cycle time becomes the basis upon which the planning horizon constraint is imposed.

[3] used the batch design formulation with mixed-product campaign schedules to formulate the batch synthesis, design and scheduling problem. In this formulation the number of stages, M, in the batch process is not fixed. Instead, each product is required to undergo the same sequence, T, of processing tasks. Units that each can perform one of the tasks are given, and in addition, 'superunits' are postulated that can combine two or more tasks. The problem is to assign tasks to units, size the units, and determine the number of parallel units in the batch process.

Problem Formulation

1) Binary variables

$$YEX_j = \begin{cases} 1 & \text{if unit } j \text{ exists} \\ 0 & \text{otherwise,} \end{cases}$$

$$YC_{cj} = \begin{cases} 1 & \text{if unit } j \text{ contains} \\ & \quad c \text{ parallel units} \\ 0 & \text{otherwise,} \end{cases}$$

$$Y_{tj} = \begin{cases} 1 & \text{if task } t \text{ is assigned} \\ & \quad \text{to unit } j \\ 0 & \text{otherwise,} \end{cases}$$

$$YF_{tj} = \begin{cases} 1 & \text{if } t \text{ is the first task} \\ & \quad \text{processed in unit } j \\ 0 & \text{otherwise.} \end{cases}$$

2) Design constraints

- Task volume requirement, V_t^T, depends on batch size, B_i, of each product and size factor, S_{it}, for each product in each task.

$$V_t^T \geq B_i S_{it}.$$

- The volume of a processing unit j must be large enough to accomodate task t if task t is assigned to unit j, ($Y_{tj} = 1$).

$$V_j \geq V_t^T - V_j^U (1 - Y_{tj}).$$

- The processing time, pt_{ij}, for each product in each unit is given by the corresponding time factor, t_{it}, for each product in task t if task t is assigned to unit j, ($Y_{tj} = 1$).

$$pt_{ij} \geq \sum_t t_{it} Y_{tj}.$$

- The number of batches, n_i, multiplied by the batch size must satisfy the production requirement, Q_i, for each product.

$$n_i B_i \geq Q_i.$$

3) Parallel equipment constraints

- The number of parallel units in each stage j is determined by the binary variable YC_{cj} multiplied by the number c,

$$N_j = \sum_c c \cdot YC_{cj}.$$

4) Scheduling constraint
   - For the UIS policy with zero cleanup times, the planning horizon constraint derived by [2] is used,

$$\sum_i n_i pt_{ij} \leq H \cdot N_j .$$

5) Logical constraints
   - If a stage *j* exists, then at least one processing task must be assigned to it,

$$\sum_t Y_{tj} \geq YEX_j .$$

   - If a stage *j* does not exist, there can be no tasks assigned to it,

$$Y_{tj} \leq YEX_j .$$

   - If a stage *j* exists, then one of the tasks assigned to it must be the first task assigned to stage *j*,

$$\sum_t YF_{tj} = YEX_j .$$

   - There cannot be more than one first task assigned to each stage,

$$\sum_t YF_{tj} \leq 1 .$$

   - A task can be the first task assigned to a stage only if the task is among those assigned to the stage,

$$YF_{tj} \leq Y_{tj} .$$

   - No tasks that occur before the first task assigned to stage *j* can be among those assigned to the stage,

$$Y_{t'j} \leq 1 - YF_{tj} \quad \text{for } t' < t .$$

   - If multiple tasks are assigned to a unit, they must be consecutive tasks,

$$Y_{tj} \leq YF_{tj} + Y_{t-1,j} .$$

   - One and only one binary variable that determines the number of parallel units in stage *j* must be active,

$$\sum_c YC_{cj} = 1 .$$

6) Objective function
   - The objective is to minimize the cost of the plant. [3] used a fixed-charge cost for each unit, γ_j, plus a nonlinear cost function on the size of the unit,

$$\text{Cost} = \sum_j N_j \left[\gamma_j + \alpha_j V_j^{\beta_j} \right] .$$

This formulation is a MINLP where all binary variables participate linearly and separably. However, it is a nonconvex problem due to the cost function, and the *bilinear* terms in the batch size constraints and the planning horizon constraints. [3] used the *outer approximation* method implemented in DICOPT ([11]) to solve a number of example problems. Due to the nonconvexities in the formulation, there is no guarantee of global optimality with the outer approximation method, but they report good results for the examples presented in the paper.

Two examples are briefly discussed to illustrate the proposed approach for multiproduct batch plants with a variety of scheduling policies. The first example consists of three products with four processing tasks and five potential units and superunits. The MINLP formulation with the SPC policy contains 33 binary variables and 54 continuous variables. With the ZW policy, the number of binary variables drops to 8, with 98 continuous variables. For the UIS policy, the formulation has 33 binary variables with 51 continuous variables.

The second example is larger and contains 6 products with 7 potential units and superunits. The SPC policy formulation contains 46 binary variables and 101 continuous variables. The MINLP formulation for the ZW policy has 11 binary and 374 continuous variables. The UIS policy formulation has 46 binary and 95 continuous variables. In all cases the examples were solved in less than 50 minutes using GAMS/DICOPT ++ on Microvax II.

Discrete-Time Formulations

A.P.F.D. Barbosa-Póvoa and S. Macchietto, [1], proposed a MILP formulation to address the problem of optimal batch design by simultaneously considering optimizing production schedule. They based their formulation on

a) an extended state-task-network (mSTN) representation of the batch plant; and

b) the discrete time representation using *uniform time discretization*.

In the STN representation, proposed in [6], all the materials are represented as states processed through a set of processing steps ('tasks'). In order to incorporate connectivity constraints the extended state-task-network (mSTN) is proposed involving the alternative design configurations considering all permitted equipment and connections allocations. Single campaign is assumed with a cyclic schedule of cycle time T repeated over a planning horizon H. A cycle represents a sequence of operations involving the production of all products and the utilization of all resources. The operational characteristics such as the allocation of equipments to tasks, batch sizes, task timings, transport of material and storage profiles are identical in each cycle. The mathematical formulation they proposed involves:

- allocation constraints for the assignment of the tasks to the units
- capacity constraints expressing the limiting equipment capability
- connectivity constraints for determining the connection of different units
- dedicated storage constraints
- mass balances
- production requirement constraints
- an objective function, which is chosen to be either the minimization of the capital cost or the maximization of plant profit.

The main variables of the formulation are:

a) binary structural variables representing the existence of an equipment;
b) binary allocation variables for the assignment of a task to a unit at the beginning of a time period;
c) continuous variables representing the capacity of a unit;
d) continuous variables corresponding to the batch size of a task to a unit at each time period;
e) amount of material delivered and received at each time period;
f) the amount of material transfered at each time period; and
g) the amount of material stored at each time period.

The proposed formulation correspond to a mixed integer linear programming (MILP) problem since they used linear cost functions to express the capital cost of equipments and time discretization to represent time. Three examples were solved illustrating:

a) the effect of limited connectivity and connection cost in the optimal design;
b) the advantages of considering simultaneously the plant design and plant connectivity rather than optimizing first the equipment sizes and then optimizing plant connectivity.

In later work, Barbosa-Póvoa and C.C. Pantelides, [1], proposed a new mathematical formulation for the optimization of batch plant design considering detailed operation characteristics (i. e., short term scheduling). This formulation also considers a uniform time discretization, the only difference lies in the plant representation. The resource-state-task (RTN) plant representation, [9], was used which corresponds to a more general and uniform description of all available production resources. However, the new formulation shares the main characteristics of the previous presented one with the same basic variables, and constraints.

Both formulations share the limitations of the discrete time formulations, which are that:

i) they correspond to an approximation of the time horizon; and
ii) they result in an unnecessary increase of the number of binary variables in particular, and in the overall size of the mathematical model.

A continuous-time formulation was proposed in [12], based on the STN representation and the scheduling formulation proposed in [13]. It gives rise to a mixed integer nonlinear programming problem which is solved using a stochastic MINLP optimizer based on an *evolutionary algorithm* (EA) with *simulated annealing* (SA) presented in [12]. The method is based on a guided stochastic generation of alternative vectors of decision variables, which explore promising areas of the search space through selection, crossover, and mutation operations applied to individuals in a population of solution candidates. It can be used to deal with nonconvex, nondifferentiable functions although it has no guarantee of convergence to even a local optimal solution. The proposed formulation involves the following basic variables:

- Main design variables representing the discrete decisions of selecting a unit (j), E_j, or a storage (s), E_s, or continuous decisions corresponding to the capacity of unit storage or utility, V_j, V_s, and U_u, respectively.

- Main operation variables corresponding to the discrete decision of allocation of task (i) in unit (j) at time T_l, W_{ijl}, and the decision of assigning task (i) in unit (j) between starting time T_l and end time $T_{l'}$, and continuous variables, the time of event (l), T_l, the batch size, the processing time and utility requirement of task (i) allocated to unit (j) starting at T_l, B_{ijl}, τ_{ijl}, U^u_{ijl}, respectively,

Based on these variables the proposed formulation involves:

1) Processing task models:

$$U^u_{ijl} = \alpha^u_{ijl} + \beta^u_{ij} B^{\gamma^u_{ij}}_{ijl},$$

expressing the consumption-generation of utilities as a function of batch size;

$$\tau_{ijl} = \alpha_{ijl} + \beta_{ij} B^{\gamma_{ij}}_{ijl} + \sum_u \mu^u_{ij} U^u_{ijl} + \sum_\alpha \mu^\alpha_{ij} A^\alpha_{jl},$$

expressing the dependence of processing time, τ_{ijl}, of batch size, B_{ijl}, utilities, U^u_{ijl}, and unit availabilities, A^α_{jl}.

2) Batch size constraints:

$$\phi^{\min}_{ij} V_j W_{ijl} \le B_{ijl} \le \phi^{\max}_{ij} V_j W_{ijl}$$

imposing the maximum and minimum capability of unit (j) when task (i) is performed.

3) Timing constraints:

$$W_{ijl}(\tau_{ijl} + T_l) = \sum_{l'>l} X_{ijll'} T_{l'},$$

which establish the relationship between processing time, τ_{ijl}, and time of event (l), T_l.

$$0 \le T_1 \le T_2 \le \cdots \le T_{l^{\max}} \le H,$$

expressing the monotonic increase in event times.

4) Allocation constraints:

$$0 \le \sum_{i\in I_j} \sum_{l''\le l'} W_{ijl''} - \sum_{i\in I_j} \sum_{l<l''} \sum_{l''\le l'} X_{ijll''} \le E_j,$$

$$\sum_{i\in I_j} \sum_{l''\le l^{\max}} W_{ijl''} = \sum_{i\in I_j} \sum_{l<l''} \sum_{l''\le l^{\max}} X_{ijll''},$$

$$W_{ijl} = \sum_{l'>l} X_{ijll'},$$

expressing the relationship between W_{ijl} and $X_{ijll'}$ operation variables, [13].

5) Material balances written for state s at event time T_l:

$$C_{sl'} = C_{sl'-1} + \sum_{i\in I_s} \sum_{j\in J_i} \rho^{in}_{sij} \sum_{l<l'} B_{ijl} X_{ijll'} - \sum_{i\in I_s} \sum_{j\in J_i} \rho^{out}_{sij} B_{ijl'},$$

$$0 \le C_{sl'} \le V_{s0} + V_s.$$

6) Utility constraints written for utility (u) at event time T_l:

$$U_{ul'} = U_{ul'-1} + \sum_{i\in I_u} \sum_{j\in J_i} \sum_{l<l'} U^u_{ijl} X_{ijll'} - \sum_{i\in I_u} \sum_{j\in J_i} U^u_{ijl'} W_{ijl'},$$

$$0 \le U_{ul'} \le U_u,$$

$$U_{uc} = \sum_{i\in I_u} \sum_{j\in J_i} \sum_{l} U^u_{ijl} T_{ijl} W_{ijl}.$$

7) Availability constraints written for unit (j) at event time T_l:

$$A^\alpha_{jl'+1} = \sum_{i\in I_j} A^\alpha_{jl'} \alpha^\alpha_{ij} W_{ijl'} - \sum_{i\in I_j} \beta^\alpha_{ij} W_{ijl'},$$

$$A^\alpha_{jl'} \le \sum_{i\in I_j} \gamma^\alpha_{ij} W_{ijl'}.$$

8) Existence constraints:

$$\sum_{i\in I_j} W_{ijl} \le E_j,$$

$$V^{\min}_j E_j \le V_j \le V^{\max}_j E_j,$$

$$V^{\min}_s E_s \le V_s \le V^{\max}_s E_s,$$

that correspond to logical restrictions on production unit and storage tank size if this unit-storage tank is present at the optimal design.

9) Production constraints:

$$C_{sl^{\max}} \ge R_s,$$

expressing the requirement of producing at least as much as the market demands for state (s).

10) Objective function:

$$\text{Profit} = \sum_{s\in S_p} p_s C_{s l^{\max}} + \sum_{s\in S_i} p_s (C_{s l^{\max}} - C_{s0}) - \sum_{s\in S_f} p_s C_{s0} - \sum_{u} c_u U_{uc};$$

the first two terms represent the revenue due to product and intermediate state production, respectively, whereas the last two terms express the cost of raw materials and utilities, respectively,

$$\text{Cost} = \sum_{j} (E_j \widetilde{\alpha}_j + \widetilde{\beta}_j V_j^{\widetilde{\gamma_j}}) + \sum_{s} (E_s \widetilde{\alpha}_s + \widetilde{\beta}_s V_s^{\widetilde{\gamma_s}});$$

the first term represent the cost of installing production unit (j), whereas the second term correspond to the cost of storage tank (s).

$$\text{Objective} = \text{Cost} - \text{Profit}.$$

This above formulation correspond to a MINLP problem with decision variables: W_{ijl}, $X_{ijll'}$, B_{ijl}, U^u_{ijl}, T_l that correspond to plant operation and E_j, E_s, V_j, V_s, U_u that represent design decisions. Nonconvexities appear in the timing constraints, material balances, utility constraints as bilinear products of binary and continuous variables and in the objective function in power form of the type $V_j^{\widetilde{\gamma_j}}$ and $V_s^{\widetilde{\gamma_s}}$. The authors proposed an evolutionary algorithm (EA) with simulated annealing (SA), [12], to solve this problem. They utilized simulated annealing to improve the poor local search ability of EA. A suitable encoding procedure is proposed which results in reduction in the number of constraints and variables by up to 50%. In particular, they explored the mathematical structure of the problem in the following sense. If $W_{ijl} = 1$ and $X_{ijll'} = 1$, unit j exists, it executes operation k which starts at $ST^j = l$ finishes at $FT^j_k = l'$ involving task $TS^j_k = i$ with batch size $BS^j_k = B_{ijl}$ and utility usage $U^j_{uk} = U^u_{ijl}$. So they proposed to replace W_{ijl}, $X_{ijll'}$, B_{ijl} and U^u_{ijl} by the operation sequence of tasks in units: task sequence $TS^j = (i_1, \dots, i_{N_j})$, task batch size $BS^j = (B_1, \dots, B_{N_j})$, task utility usage $U^j_u = (U^u_1, \dots, U^u_{N_j})$, start time $ST^j = (l_1, \dots, l_{N_j})$, finish time $FT^j = (l'_1, \dots, l'_{N_j})$. In this way the decision variables become (E_j, V_j, E_s, V_s, U_u, T_l, TS^j, BS^j, U^j_u, ST^j, FT^j). The algorithm starts with an initial guess and evolves a number of candidate instances for these variables. The allocation and the capacity constraints are automatically satisfied by each candidate solution and T_l are chosen so that the timing constraints are also satisfied. Two examples are presented to illustrate the applicability of the proposed approach to solve batch design problem involving detailed scheduling constraints. Linear and nonlinear task processing times and unit cost models are considered for both the examples. For the first example considering linear functions for processing times and unit cost models the results obtained are compared with a discrete time formulation, [8], and found to outperform it in terms of number of variables which is expected since the formulation is based on the continuous time description and the computational requirement for the solution of their model. Considering nonlinear models for processing times and unit costs, the resulting model for a problem with 4 production units, 4 storage tanks, 5 tasks and 4 states, involves 62 integer and 34 continuous variables and 122 constraints. This example was the largest presented in this work, and required considerable computational effort, 7849.23 CPU seconds on a SUN ULTRAstation-1.

See also

- ▶ Chemical Process Planning
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ Job-shop Scheduling Problem
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis

- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming
- ▶ Stochastic Scheduling
- ▶ Vehicle Scheduling

References

1. Barbosa-Póvoa APFD, Macchietto S (1994) Detailed design of multipurpose batch plants. Comput Chem Eng 18:1014–1042
2. Birewar DB, Grossmann IE (1989) Incorporating scheduling in the optimal design of multiproduct batch plants. Comput Chem Eng 13:141–161
3. Birewar DB, Grossmann IE (1990) Simultaneous synthesis, sizing and scheduling of multiproduct batch plants. Industr Eng Chem Res 29:2242–2251
4. Grossmann IE, Sargent RWH (1979) Optimum design of multipurpose batch plants. Industr Eng Chem Process Des Developm 18:343–348
5. Kocis GR, Grossmann IE (1989) Computational experience with DICOPT solving MINLP problems in process synthesis engineering. Comput Chem Eng 13:307–315
6. Kondili E, Pantelides CC, Sargent RWH (1993) A general algorithm for short-term scheduling of batch operations - I. MILP formulation. Comput Chem Eng 17:211–227
7. Ku H, Karimi I (1986) Scheduling in multistage serial batch processes with finite intermediate storage - Part I. MILP formulation; Part II. Approximate algorithms. AIChE Annual Meeting, Miami
8. Manual gBSS, general batch scheduling system - User manual and language reference, Imperial College
9. Pantelides CC (1994) Unified frameworks for the optimal proces planning and scheduling. Proc. Second Conf. Foundations of Computer Aided Operations, pp 253–274
10. Sparrow RE, Forder GJ, Rippin DWT (1975) The choice of equipment sizes for multiproduct batch plants. Heuristics vs. branch and bound. Industr Eng Chem Process Des Developm 14:197–203
11. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer-approximation method for MINLP optimization. Comput Chem Eng 14:769–782
12. Xia Q, Macchietto S (1997) Design and synthesis of batch plants- MINLP solution based on a stochastic method. Comput Chem Eng 21:S697–S702
13. Zhang X, Sargent RWH (1994) The optimal operation of mixed production facilities - general formulation and some solution approaches for the solution. Proc. 5th Internat. Symp. Process Systems Engin. (Kyongju, Korea), pp 171–177

MINLP: Generalized Cross Decomposition

KAJ HOLMBERG
Department Math., Linköping Institute Technol., Linköping, Sweden

MSC2000: 90C11, 90C30, 49M27

Article Outline

Keywords

Decomposition; Primal-dual; Nonlinear; Mixed integer

Decomposition methods, such as the classical Benders decomposition (cf. ▶ Generalized Benders decomposition), [1], and Dantzig–Wolfe decomposition, [3], have been used to solve many different large structured optimization problems, by decomposing them with the help of relaxation of constraints or fixation of variables. The success of such an approach depends very much on the structure of the problem. In some cases these methods are very efficient, but in other cases they are not competitive with other techniques.

However, the simple elegance of these basic principles has inspired many researchers to propose modifications of the basic methods, mostly aimed at improving the efficiency of the methods, but also aimed at extending the applicability of the approaches.

Dantzig–Wolfe decomposition, originally for linear programming problems, [3], has been extended to convex nonlinear programming problems, [2], under several names, for example generalized linear programming. We will here simply use the term 'nonlinear Dantzig–Wolfe decomposition'.

Benders decomposition, originally for linear mixed integer programming problems, [1], has been extended to partly convex nonlinear programming problems, [5], under the name 'generalized Benders decomposition'.

On the other hand, among the numerous suggestions for modifications to increase the efficiency, there is one which in a way shares the simplicity and clear principle of the basic methods, namely *cross decomposition*, [11]. Usually described as a combination of Benders decomposition and Dantzig–Wolfe decomposition, simultaneously using the two methods in an iterative manner, the method borrows its basic convergence properties from these two methods. However, one can also view cross decomposition as the more general method, and Benders and Dantzig–Wolfe decomposition as modifications of cross decomposition, obtained by excluding one of the subproblems and one of the master problems.

Cross decomposition was originally developed for linear mixed integer programming problems, [11], but the approach is more general and not restricted to such problems. The first application of cross decomposition was to the capacitated facility location problem, [12], and produced a solution method which is recognized as one of the most efficient existing methods for that problem. However, another early application was to the stochastic transportation problem (a convex problem with linear parts), [10].

Here we will describe 'generalized cross decomposition', which was first proposed in [6], and more thoroughly treated in [7]. The generalization of the procedure, parallel to that in [5] for generalized Benders decomposition, enables the solving of nonlinear programming problems with convex parts, for example nonlinear mixed integer programming problems, see for example [4].

The Problem

Consider the following general optimization problem.

$$
(\mathrm{P})\quad \begin{cases} v^* = & \min f(x, y) \\ \text{s.t.} & G_1(x, y) \le 0 \\ & G_2(x, y) \le 0 \\ & x \in X \\ & y \in Y \end{cases}
$$

where X and Y are compact, nonempty sets. Assume that X is convex and f, G_1 and G_2 are proper convex functions in x for any fixed $y \in Y$, i. e. that the problem is convex in x. Also assume that that f, G_1 and G_2 are bounded and Lipschitzian on (X, Y). Note that we do not assume any convexity in the y-variables. An important case is when Y is a (finite) set of integers.

Furthermore we assume the following (as was done in [5] for generalized Benders decomposition). The optimization with respect to x of the Lagrangian functions must be possible to do 'essentially independent' of y (called property P by A.M. Geoffrion). We therefore assume that the functions q_1, q_2, q_3 and q_4 exist, such that $f(x, y) + u_1^\top G_1(x, y) + u_2^\top G_2(x, y) = q_1(q_3(x, u), y, u)$, $\forall x, y, u$, and $\widetilde{u}_1^\top G_1(x, y) + \widetilde{u}_2^\top G_2(x, y) = q_2(q_4(x, \widetilde{u}), y, \widetilde{u})$, $\forall x, y, \widetilde{u}$, where q_3 and q_4 are scalar functions, q_1 and q_2 are increasing in their first argument, and $\widetilde{u}$ is assumed to belong to the set of all possible nonnegative, normalized directions $C = \{\widetilde{u} \ge 0 \colon e^\top \widetilde{u} = 1\}$, where e is a vector of ones. Since f, G_1 and G_2 are convex in x and bounded and Lipschitzian on (X, Y), the same applies to q_1 for any fixed $u \ge 0$, and to q_2 for any fixed $\widetilde{u} \in C$.

The optimal solution of P is denoted by (x^*, y^*). We will also mention the case when P is convex, i. e. where f, G_1 and G_2 are convex functions (in y too) and Y is a convex set. Lagrangian duality can be used to get a dual solution (the optimal Lagrange multipliers), denoted by $u^* = (u_1^*, u_2^*)$.

Let us for convenience introduce the following notation.

$$
\begin{aligned}
&L(x, y, u) = f(x, y) + u_1^\top G_1(x, y) + u_2^\top G_2(x, y),\\
&\widetilde{L}(x, y, \widetilde{u}) = \widetilde{u}_1^\top G_1(x, y) + \widetilde{u}_2^\top G_2(x, y),\\
&L_1(x, y, u_1) = f(x, y) + u_1^\top G_1(x, y),\\
&\widetilde{L}_1(x, y, \widetilde{u}_1) = \widetilde{u}_1^\top G_1(x, y).
\end{aligned}
$$

The Primal Master Problem

Using the primal structure of (P) we can rewrite it as

$$
v^* = \min_{y \in V} h(y),
$$

where $\forall y = \in V$,

$$\begin{cases} h(y) = & \min f(x,y) \\ \text{s.t.} & G_1(x,y) \le 0 \\ & G_2(x,y) \le 0 \\ & x \in X \end{cases}$$

and

$$V = \left\{ y \in Y \colon \exists x \in X : \begin{array}{c} G_1(x,y) \le 0, \\ G_2(x,y) \le 0 \end{array} \right\}.$$

The problem is convex in x, so we can use Lagrangian duality to get, $\forall y \in V$,

$$h(y) = \max_{u \ge 0} \min_{x \in X} L(x,y,u).$$

A similar expression can be obtained for V:

$$V = \left\{ y \in Y \colon \left(\max_{\widetilde{u} \in C} \min_{x \in X} \widetilde{L}(x,y,\widetilde{u}) \right) \le 0 \right\}.$$

The full primal master problem is given below:

$$\begin{cases} v^* & = \min q \\ \text{s.t.} & q \ge \min\limits_{x \in X} L(x,y,u), \quad \forall u \ge 0, \\ & 0 \ge \min\limits_{x \in X} \widetilde{L}(x,y,\widetilde{u}), \quad \forall \widetilde{u} \in C, \\ & y \in Y. \end{cases}$$

This problem has an infinite number of constraints, one for each nonnegative dual point and one for each nonnegative dual direction. Each constraint contains an optimization problem (minimization with respect to x), which should in theory be solved for all $y \in Y$ before the main problem, $\min_{y \in Y} h(y)$, can be solved. However, we have

$$\min_{x \in X} L(x,y,u) = q_1 \left(\min_{x \in X} q_3(x,u), y, u \right)$$

and

$$\min_{x \in X} \widetilde{L}(x,y,\widetilde{u}) = q_2 \left(\min_{x \in X} q_4(x,\widetilde{u}), y, \widetilde{u} \right).$$

Since q_1 and q_2 are proper, convex, bounded and Lipschitzian on X, and X is compact and convex, the optima in x (for fixed u and $\widetilde{u}$) will be attained. q_1 and q_2 are increasing in their first argument, so the minimization in x can be made in q_3 and q_4 instead, and the value of y will thus not influence the result of this minimization. The minimization over x can be made once (for any y) and the result will then be true for all $y \in Y$.

The *relaxed primal master problem* only contains a finite number of cuts (with index sets P_U and R_U) which gives an approximate description of $h(y)$ and V, and an optimal objective function value, $v_{PM} \le v^*$. Since the part of the problem that is described by the constraints is convex in x, v_{PM} will converge asymptotically towards v^* as the sets of constraints grow.

The constraints can now be expressed as

$$q \ge q_1 \left(\min_{x \in X} q_3(x, u^{(k)}), y, u^{(k)} \right), \quad \forall k \in P_U,$$
$$0 \ge q_2 \left(\min_{x \in X} q_4(x, \widetilde{u}^{(k)}), y, \widetilde{u}^{(k)} \right), \quad \forall k \in R_U.$$

The minimization in x can now be made independently in each constraint, since the other arguments in q_3 and q_4, namely u and $\widetilde{u}$, are fixed. Since the minima are attained, we use the notation $x^{(k)}$, $\forall k \in P_U$, and $\widehat{x}^{(k)}$, $\forall k \in R_U$, for the minimizers of q_3 and q_4.

Inserting this, we obtain the final form of the relaxed primal master problem.

$$\text{(PM)} \quad \begin{cases} v_{\text{PM}} & = \min q \\ \text{s.t.} & q \ge L(x^{(k)}, y, u^{(k)}), \quad \forall k \in P_U, \\ & 0 \ge \widetilde{L}(\widehat{x}^{(k)}, y, \widetilde{u}^{(k)}), \quad \forall k \in R_U, \\ & y \in Y. \end{cases}$$

The constraints in the first set are called *value cuts*, and those in the second set are called *feasibility cuts*.

The Dual Master Problem

Using Lagrangian duality on (P) yields a relaxation and a lower bound, v_L, on v^*:

$$v_L = \max_{u_1 \ge 0} g(u_1)$$

where, $\forall u_1 \geq 0$,

$$\begin{cases} g(u_1) & = \min L_1(x, y, u_1) \\ \text{s.t.} & G_2(x, y) \leq 0 \\ & x \in X \\ & y \in Y . \end{cases}$$

This leads to a dual master problem, which is a convexification of the problem. If (P) is not convex a duality gap might occur. We denote the subset of the solutions that are included by $(x^{(k)}, y^{(k)})$, $\forall k \in P_X$, and obtain the restricted dual master problem as

$$\text{(DM)} \quad \begin{cases} v_{DM} & = \max q \\ \text{s.t.} & q \leq L_1(x^{(k)}, y^{(k)}, u_1), \\ & \quad \forall k \in P_X , \\ & u_1 \geq 0. \end{cases}$$

The Subproblems

The primal subproblem is a convex problem in x, obtained by fixing y to $\overline{y}$.

$$\text{(PS)} \quad \begin{cases} h(\overline{y}) & = \min f(x, \overline{y}) \\ \text{s.t.} & G_1(x, \overline{y}) \leq 0 \\ & G_2(x, \overline{y}) \leq 0 \\ & x \in X . \end{cases}$$

A solution to (PS) is assumed to consist of both a primal solution, $x^{(k)}$, and a dual solution, $(u_1^{(k)}, u_2^{(k)})$. Due to the convexity we can use Lagrangian duality without creating a duality gap.

$$\text{(PSL)} \quad h(\overline{y}) = \sup_{u \geq 0} \min_{x \in X} L(x, \overline{y}, u) .$$

If (PS) is infeasible, (PSL) will be unbounded in u, and a solution is represented by a direction, $\widetilde{u}^{(k)}$. A valid cut for the primal master problem also requires a corresponding primal solution, $\widehat{x}^{(k)}$, obtained by solving

$$\min_{x \in X} \widetilde{L}(x, \overline{y}, \widetilde{u}^{(k)}) .$$

(Note that $\widehat{x}^{(k)}$ is not feasible in (PS).)

The dual subproblem is the following (nonconvex) problem, obtained by relaxing the first set of constraints in (P) and fixing the Lagrange multipliers u_1 to $\overline{u}_1$:

$$\text{(DS)} \quad \begin{cases} g(\overline{u}_1) & = \min L_1(x, y, \overline{u}_1) \\ \text{s.t.} & G_2(x, y) \leq 0 \\ & x \in X \\ & y \in Y \end{cases}$$

To handle unbounded dual solutions, $\widetilde{u}_1$, we can use the following subproblem:

$$\text{(UDS)} \quad \begin{cases} \widetilde{v}(\widetilde{u}_1) & = \min \widetilde{L}_1(x, y, \widetilde{u}_1) \\ \text{s.t.} & G_2(x, y) \leq 0 \\ & x \in X \\ & y \in Y . \end{cases}$$

(UDS) does not produce a bound on v^*, but if $\widetilde{v}(\widetilde{u}_1) \leq 0$ it yields a dual cut that will eliminate $\widetilde{u}_1$.

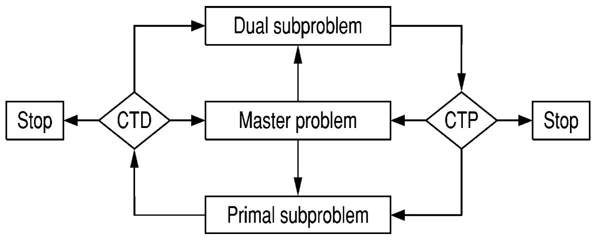
The Cross Decomposition Algorithm

In the subproblem phase of the cross decomposition method we iterate between the primal subproblem (PS) and the dual subproblem (DS) (or (UDS)).

The primal subproblem, (PS), supplies an upper bound, $h(\overline{y})$, on v^*, and $\overline{u}_1$ for the dual subproblem. The dual subproblem, (DS), supplies a lower bound, $g(\overline{u}_1)$, on v^*, and $\overline{y}$ for the primal subproblem. If (PS) has an unbounded solution, $\widetilde{u}_1$, we use (UDS) (instead of (DS)) to get $\overline{y}$.

Unfortunately, the lack of controllability for the important parts of the solutions, y and u_1, which occurs unless the problem is strictly convex, implies that this procedure alone cannot be expected to converge to the optimal solution.

We therefore need to use the master problems to ensure convergence. (PM) or (DM) can be solved with all the constraints generated by the subproblem solutions. We have all the known results for generalized Benders or nonlinear Dantzig–Wolfe decomposition to fall back on, so this technique is well known. After the solution of one master problem, the subproblem phase is reentered. (We do not switch to Benders or Dantzig–Wolfe decomposition completely.)



MINLP: Generalized Cross Decomposition, Figure 1

We will later describe convergence tests that tell us exactly when to use a master problem. The existence of such convergence tests is a very important aspect of cross decomposition. Let us, before getting any further, give below a short algorithm for cross decomposition algorithm.

Let us denote the convergence test in step 3 (before (PS)) by CTP and the convergence test in step 6 (before (DS)) by CTD. The optimality tests (step 2 and step 5) are included in the convergence tests, and the decision about where to go is based on the results of both tests. The algorithm is pictured in Fig. 1.

```
0    Get a starting $\overline{u}$.
1    Solve (DS) (or (UDS)).
2    IF optimal go to 8.
3    IF not convergence, go to 7A (or 7B).
4    Solve (PS).
5    IF optimal go to 8.
6    IF not convergence go to 7B (or 7A). ELSE
     go to 1.
7A   Solve (PM). Go to 4.
7B   Solve (DM). Go to 1.
8    Stop. The solution from (PS) is optimal.
```

We can start with either one of the subproblems, so a good primal starting solution can also be utilized.

If CTP indicates that (PS) will not give further convergence, we use (PM). If CTD indicates failure of convergence for (DS), we can use (DM) (which however gives certain convergence only if (P) is convex). After (PM) we go to (PS) and after (DM) we go to (DS), in order to make use of the output of the master problems. In the general nonconvex case, it is not necessary to use (DM). It is even possible to omit the convergence tests CTD if only (DM) is used.

The Convergence Tests

Returning to the question of convergence in the subproblem phase, we make the following definitions of ε-improvements.

'ε-bound-improvement' is an improvement of at least ε of the upper or lower bound.

'ε-cut-improvement' is a generation of a new, so far unknown cut, that is at least ε better (i. e. has a value of at least ε higher or lower) than all known cuts at some point.

Discussing *linear mixed integer problems*, as in [11], one can let $\varepsilon = 0$. In such a case we simply omit ε from the above notation.

Cut-improvement thus means that a new cut will be included in one of the restricted master problems and that the description of the functions $h(y)$ or $g(u_1)$ or the set Y is refined. By 'improvement' we will, in the rest of this paper mean bound-improvement and/or cut-improvement. When using unbounded solutions as input no finite bounds are obtained, so bound-improvement can not appear. Also, a cut giving a cut-improvement can be a value cut or a feasibility cut, i. e. generated by output in the form of unbounded as well as bounded solutions.

Let us by *primal cut-improvement* denote generation of a primal cut (for (PM)) and by *dual cut-improvement* denote generation of a dual cut (for (DM)). We also use the notation 'primal' or 'dual bound-improvement' to indicate which of the two subproblems that gave the improvement, i. e. *primal bound-improvement* means that $h(\overline{y}) < \overline{v}$ and *dual bound-improvement* means that $g(\overline{u}_1) > \underline{v}$. ($\overline{v}$ is the least upper bound known and $\underline{v}$ the largest lower bound known.)

The convergence tests are originally formulated to give the answers to the following questions.

- Can $\overline{y}$ give a bound-improvement in (PS)?
- Can $\overline{u}_1$ give a bound-improvement in (DS)?

Testing extreme rays, $\widetilde{u}_1$, for convergence, we note that the subproblem (UDS) can not give bound-improvement. We call the test of unbounded solutions CTDU.

We now give the convergence tests, CT, with strict inequalities, following [11]:

CTP	If $L(x^{(k)}, \overline{y}, u^{(k)}) < \overline{v}$, $\forall k \in P_U$, and $\widetilde{L}(\widehat{x}^{(k)}, \overline{y}, \widetilde{u}^{(k)}) < 0$, $\forall k \in R_U$, then $\overline{y}$ will give primal improvement. If not, use a master problem.
CTD	If $L_1(x^{(k)}, y^{(k)}, \overline{u}_1) > \underline{v}$, $\forall k \in P_X$, then $\overline{u}_1$ will give dual improvement. If not, use a master problem.
CTDU	If $\widetilde{L}_1(x^{(k)}, y^{(k)}, \widetilde{u}_1) > 0$, $\forall k \in P_X$, then $\widetilde{u}_1$ will give dual cut-improvement. If not, use a master problem.

We call CTD and the first part of CTP *value convergence tests* and CTDU and the second part of CTP *feasibility convergence tests*. This conforms to the notation of value and feasibility cuts in the master problems.

One can show that the convergence tests CTP and CTD are necessary for bound-improvement and sufficient for cut- or bound-improvement, see [7]. The convergence tests CTDU are sufficient for cut-improvement.

However, there can be an infinite number of primal and/or dual improvements, so one can not be certain that CT will fail within a finite number of steps. For this reason it is necessary to consider ε-improvements.

We need the following ε-convergence tests, CT ε:

CTPε	If $L(x^{(k)}, \overline{y}, u^{(k)}) \leq \overline{v} - \varepsilon$, $\forall k \in P_U$, and $\widetilde{L}(\widehat{x}^{(k)}, \overline{y}, \widetilde{u}^{(k)}) \leq -\varepsilon$, $\forall k \in R_U$, then $\overline{y}$ will give primal ε-improvement. If not, use a master problem.
CTDε	If $L_1(x^{(k)}, y^{(k)}, \overline{u}_1) \geq \underline{v} + \varepsilon$, $\forall k \in P_X$, then $\overline{u}_1$ will give dual ε-improvement. If not, use a master problem.
CTDUε	If $\widetilde{L}_1(x^{(k)}, y^{(k)}, \widetilde{u}_1) \geq \varepsilon$, $\forall k \in P_X$, then $\widetilde{u}_1$ will give dual ε-cut-improvement. If not, use a master problem.

The ε-value convergence tests correspond to the value cuts of the master problems, and the ε used corresponds directly to a change of ε of the bounds (ε-bound-improvement). The ε-feasibility convergence tests, on the other hand, correspond to feasibility cuts of the master problems, and the ε used corresponds to the 'infeasibility' it gives some previously feasible points, which is what we call ε-cut-improvement for feasibility cuts. While these ε-tests are sufficient for ε-improvement, they are not necessary. To prove necessity would require an inverse Lipschitz assumption, namely that for points a certain distance apart, the value of a function (the feasibility cut) should differ by at least a certain amount. The following result is proved in [7].

The ε-value convergence tests of CTP ε, the feasibility convergence tests of CTP and the ε-convergence tests CTD ε are necessary for ε-bound-improvement. The ε-convergence tests CT ε are sufficient for ε-bound- or ε-cut-improvement, in the sense that they are sufficient for one of the following.

I) ε-bound-improvement.
II) ε-cut-improvement.
III) ε_1-bound-improvement and ε_2-cut-improvement, where $\varepsilon_1 + \varepsilon_2 = \varepsilon$.

Now it is possible to verify finiteness of the convergence tests. A formal proof for this can be found in [7]. The following reasoning is used.

When the bounded set Y is completely described with an accuracy better than ε by either value cuts or feasibility cuts, the ε-convergence tests will fail (if not earlier). Each time the ε-convergence tests do not fail, we will get improvement according to one of the three cases mentioned above.

A finite number of ε-bound-improvements is obviously sufficient to decrease the finite distance between $\overline{v}$ and v^* to less than ε. After an ε-cut-improvement, the new cut describes $h(y)$ with an accuracy better than ε in the area around $\overline{y}$ where $h(y) < L(x^{(l)}, y, u^{(l)}) + \varepsilon$. Due to the Lipschitzian property of the functions f, G_1 and G_2, there is a least distance, δ, proportional to ε, from $\overline{y}$ to any point y violating this inequality, and the ε-convergence tests will fail for any point with a distance to $\overline{y}$ less then δ. The bounded set V can be completely covered by a finite number of such areas.

In the third case, an ε_1-bound-improvement together with an ε_2-cut-improvement, where $\varepsilon_1 + \varepsilon_2 = \varepsilon$, we can ignore the least of ε_1 and ε_2, leaving us with the other one greater or equal to $\varepsilon/2$. This yields one of the two cases above, so exchanging ε for $\varepsilon/2$ finiteness is still assured.

For unbounded solutions to (PS), any y satisfying $\widetilde{L}(\widehat{x}^{(l)}, y, \widetilde{u}^{(l)}) > -\varepsilon$ will make the ε-convergence tests fail, and because of the Lipschitzian property of G_1 and G_2 there is a least distance, δ (proportional to ε), from $\overline{y}$ to any y not making the ε-convergence tests fail. Thus an area of a certain least size is made 'infeasible', and the bounded set $Y \setminus V$ can be covered by a finite set of such areas. Thus CTP ε will fail within a finite number of steps.

Note that it is enough that CTP ε fails. To obtain finiteness we do not need to use CTD ε, even if it might be useful in practice. We cannot show that CTD ε will fail within a finite number of steps. Dual ε-bound-improvement can only occur a finite number of times, but dual ε-cut-improvement can occur an infinite number of times, since the area to be covered by the cuts is the nonnegative orthant of u_1.

We therefore require that (PM) is used regularly. (One could even skip (DM) completely.) The following is our main result.

Theorem 1 *The generalized cross decomposition algorithm equipped with ε-convergence tests CT ε finds an ε-optimal solution to (P) in a finite number of steps, if the generalized Benders decomposition algorithm does.*

All the results for generalized Benders decomposition can be directly used for generalized cross decomposition, especially the following two.

In [5] it is shown that generalized Benders decomposition has finite exact convergence if Y is a finite discrete set. The worst case is solving the primal subproblem with each possible $y \in Y$, which will give a perfect description of $h(y)$ and V on Y.

Therefore we know that if Y is a finite discrete set, the generalized cross decomposition algorithm will solve P exactly in a finite number of steps.

It is also shown in [5] that generalized Benders decomposition terminates in a finite number of steps to an ε-optimal solution, i. e. where $\overline{v} - \underline{v} \leq \varepsilon$ for any given $\varepsilon > 0$, if the set of interesting (u_1, u_2)-solutions (possible optimal solutions to the primal subproblem) is bounded and $Y \subseteq V$. This makes the primal feasibility cuts (and the corresponding convergence tests) unnecessary. So for generalized cross decomposition, we know the following.

If $h(y)$ is bounded from above for all $y \in Y$, i. e. (PS) has a feasible solution for every $y \in Y$, then the cross decomposition algorithm (without UDS and the ε-feasibility convergence tests of CT ε) will yield finite ε-convergence, i. e. yield $\overline{v} - \underline{v} \leq \varepsilon$ in a finite number of steps, for any given $\varepsilon > 0$.

If $Y \not\subseteq V$ one might get asymptotic convergence of the feasibility cuts, i. e. solutions getting closer and closer to the feasible set, but never actually becomes feasible. If one is reluctant to base a stopping criterion on ε-feasible solutions, one could use penalty functions, which transforms feasibility cuts to value cuts and gives better possibilities of handling cases where $Y \not\subseteq V$. One could also use artificial variables for this purpose. As for nonlinear penalty function techniques, one should not forget the Lipschitzian assumption made.

The practical motivation behind cross decomposition is to replace the hard primal master problem with the easier dual subproblem to the largest possible extent. Therefore the theoretical result that generalized cross decomposition equipped with ε-convergence tests does not have asymptotically weaker convergence than generalized Benders decomposition, is quite satisfactory.

Finally one might mention that these approaches also has been applied to pure (not mixed) integer programming problems in [8] (nonlinear) and [9] (linear). In such cases, various duality gaps appear, and exact solution is not possible. However, the approach may be useful for obtaining good bounds on the objective function value, which are to be used in branch and bound methods.

See also

- ▶ Chemical Process Planning
- ▶ Decomposition Principle of Linear Programming
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis

▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
▶ Mixed Integer Nonlinear Programming
▶ Simplicial Decomposition
▶ Simplicial Decomposition Algorithms
▶ Stochastic Linear Programming: Decomposition and Cutting Planes
▶ Successive Quadratic Programming: Decomposition Methods

References

1. Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Math 4:238–252
2. Dantzig GB (1963) Linear programming and extensions. Princeton Univ. Press, Princeton
3. Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. Oper Res 8:101–111
4. Floudas CA (1995) Nonlinear and mixed-integer optimization: Fundamentals and applications. Oxford Univ. Press, Oxford
5. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
6. Holmberg K (1985) Decomposition in large scale mathematical programming. PhD Thesis Dept. Math. Linköping Univ.
7. Holmberg K (1990) On the convergence of cross decomposition. Math Program 47:269–296
8. Holmberg K (1992) Generalized cross decomposition applied to nonlinear integer programming problems: Duality gaps and convexification in parts. Optim 23:341–356
9. Holmberg K (1994) Cross decomposition applied to integer programming problems: Duality gaps and convexification in parts. Oper Res 42(4):657–668
10. Holmberg K, Jörnsten K (1984) Cross decomposition applied to the stochastic transportation problem. Europ J Oper Res 17:361–368
11. Van Roy TJ (1983) Cross decomposition for mixed integer programming. Math Program 25:46–63
12. Van Roy TJ (1986) A cross decomposition algorithm for capacitated facility location. Oper Res 34:145–163

MINLP: Global Optimization with αBB

CLAIRE S. ADJIMAN, CHRISTODOULOS A. FLOUDAS
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 65K05, 90C11, 90C26

Article Outline

Keywords

Global optimization; Twice-differentiable MINLPs; Branch and bound; αBB algorithm

The αBB global optimization algorithm for continuous twice-differentiable NLPs (cf. ▶ αBB algorithm) [2,4,5,6,8,18] can be used to design global optimization algorithms for *mixed integer nonconvex problems* [1,3,7]. One such algorithm, the *special structure mixed integer αBB algorithm* (SMIN-αBB) is designed to address the class of *MINLPs* in which all the integer variables are binary variables that participate in linear or mixed-bilinear terms and in which the nonconvex functions in the continuous variables have continuous second order derivatives. This algorithm is an extension of the αBB algorithm and branching is performed on both the continuous and the binary variables. A second algorithm, the *general structure mixed integer αBB algorithm* (GMIN-αBB), guarantees convergence to the global optimum of a much broader class of problems. The integer variables may participate in the problem in a very general way, provided that the continuous relaxation of the MINLP is C^2 continuous. This article describes both algorithms.

The SMIN-αBB Algorithm

The *SMIN-αBB algorithm* [1,3,7] guarantees finite ϵ-convergence to the global solution of MINLPs belonging to the class

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x}) + \mathbf{x}^\top A_f \mathbf{y} + c_f^\top \mathbf{y} \\ \text{s.t.} & g_i(\mathbf{x}) + \mathbf{x}^\top A_{g,i}\mathbf{y} + c_{g,i}^\top \mathbf{y} \leq 0, \\ & \quad i = 1, \ldots, m, \\ & h(\mathbf{x}) + \mathbf{x}^\top A_{h,i}\mathbf{y} + c_{h,i}^\top \mathbf{y} = 0, \\ & \quad i = 1, \ldots, p, \\ & \mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U] \\ & \mathbf{y} \in \{0,1\}^q \end{cases} \tag{1}$$

where $f(\mathbf{x})$, $g(\mathbf{x})$, and $h(\mathbf{x})$, are continuous, twice-differentiable functions, m is the number of inequality constraints, p is the number of equality constraints, q is the dimension of the binary variable vector, A_f, $A_{g,\,i}$ and $A_{h,\,i}$ are $n \times q$ matrices, and c_f, $c_{g,\,i}$ and $c_{h,\,i}$ are q-dimensional vectors.

The main features of any *branch and bound algorithm* are the strategy used to generate valid lower and upper bounds for the problem and the selection criteria for the branching node and the branching variable. Optionally, a procedure to tighten the variable bounds may be considered. Each one of these issues is examined in the context of the SMIN-αBB algorithm.

Generation of Valid Upper and Lower Bounds

A local solution of the nonconvex MINLP (1) using one of the algorithms described in [13] constitutes a valid upper bound on the global optimum solution of that problem. The *generalized Benders decomposition* (GBD) [10,14] or a standard MINLP branch and bound algorithm (B&B) [9,11,15,19,20] may be used to obtain such a solution. When there are no mixed-bilinear terms, the *outer approximation with equality relaxation* (OA/ER) [12,16] may also be used. Alternatively, the binary variables may be fixed to a combination of 0 and 1 values and the resulting nonconvex NLP may be solved locally.

A relaxed problem which can be solved to global optimality must be constructed from problem (1) in order to obtain a valid lower bound. The class of MINLPs in which the continuous functions $f(\mathbf{x})$, $g_i(\mathbf{x})$, and $h_i(\mathbf{x})$, are convex can be solved to global optimality using the GBD or B &B algorithms, and, when there are no mixed-bilinear terms, the OA/ER algorithm. To identify a *guaranteed lower bound* on the solution of the problem, it therefore suffices to construct convex underestimators for the nonconvex functions $f(\mathbf{x})$, $g_i(\mathbf{x})$, and $h_i(\mathbf{x})$, and to solve the resulting problem with one of these algorithms. The rigorous *convexification/relaxation strategy* used in the αBB algorithm for nonconvex continuous problem [2,4,5,6] allows the construction of the desired lower bounding MINLP. This scheme is based on a decomposition of the functions into a sum of terms with special mathematical structure, such as linear, convex, bilinear, trilinear, fractional, fractional trilinear, univariate concave and general nonconvex terms. A different convex relaxation technique is then applied for each class of term. The fact that a summation of convex functions is itself a convex function is then used to construct overall function underestimators and arrive at a convex lower bounding MINLP.

Selection of Branching Node

A list of the lower bounds on all the nodes that have not yet been explored during the branch and bound procedure is maintained. A number of approaches can be used to select the next branching node, such as depth-first, breadth-first or smallest lower bound first. Since the purpose of the algorithm is to identify the global solution of the problem, all promising regions, that is, all regions for which the lower bound is less than or equal to the best upper bound on the solution, must be explored. The strategy that usually minimizes the number of nodes to be examined and therefore the CPU requirements of the algorithm is used to choose the next branching node in the SMIN-αBB algorithm. Thus, the node with the smallest lower bound is selected.

Selection of Branching Variable

Several strategies can be used to select the next variable to be branched on. If a continuous variable is judiciously chosen, the partition results in an improvement of the lower bound on the problem through a tightening of the convex relaxation of the nonconvex continuous functions. Binary variables have an indirect effect on the quality of the convex underestimators as they influence the range of values that the continuous variables can take on.

A first branching variable selection scheme exploits the direct relationship between the range of the continuous variables and the quality of the lower bounds and therefore branches only on these variables. One of the rules available for the αBB algorithm [2] is used for the selection. These are based on the size of the variable ranges, or on a measure of the quality of the underestimator for each term, or on a measure of each variable 's overall contribution to the quality of the underestimators.

A second approach aims to first tackle the combinatorial aspects of the problem by branching only on binary variables for the first q levels of the branch and bound tree, where q is the number of binary variables. The nonconvexities are dealt with on subsequent levels of the tree, by branching on the continuous variables. The specific binary variable used for branching is chosen randomly or from a priority assigned on the basis of its effect on the structure of the problem. In particular, the binary variables that influence the bounds on the greatest number of variables are given the highest priorities. Once all the binary variables have been fixed, the problems that must be considered are *continuous* nonconvex and convex problems for the upper and lower bound respectively. The bounding of the nodes below level q is therefore less computationally intensive than above that level.

A third approach also involves branching on the continuous and binary variables although the choice is no longer based on the level in the tree. To increase the impact of binary variable branching on the quality of the lower bound, such a variable is selected when a continuous relaxation of the problem indicates that the two children node will have significantly different lower bounds, and that one of them may even be infeasible. Thus, if one of the binary variables is close to 0 or 1 at a local solution of the continuous relaxation, it is branched on. The degree of closeness is an arbitrary parameter which can typically be set to 0.1 or 0.2. If no 'almost-integer' binary variable is found, a continuous variable is selected for branching. In general, this hybrid strategy results in a faster improvement in the lower bounds than the second approach, but it is more computationally intensive because a continuous relaxation must be solved before selecting a branching variable and a larger number of MINLP nodes may be encountered during the branch and bound search.

Variable Bound Updates

The tightening of variable bounds is a very important step because of its impact on the quality of the underestimators. For continuous variables, the strategies developed for the αBB algorithm may be used [2]. For the SMIN-αBB algorithm, they rely on the solution of several convex MINLPs in the optimization-based approach, or the iterative interval evaluation of the constraints in the interval-based approach. In this latter case, the binary variables are relaxed during the interval computation.

```
PROCEDURE binary variable bound update()
    Consider $R = \{(\mathbf{x}, \mathbf{y}) \in F : y_i = 0\}$;
    Test interval feasibility of $R$;
    IF infeasible, set $y_i^L = 1$;
    Consider $R = \{(\mathbf{x}, \mathbf{y}) \in F : y_i = 1\}$;
    Test interval feasibility of $R$;
    IF infeasible,
        IF $y_i^L = 1$, RETURN(infeasible node);
        ELSE, set $y_i^U = 0$;
    RETURN(new bounds $y_i^L$ and $y_i^U$);
END binary variable bound update;
```

Procedure for binary variable bound updates

In the case of binary variables, successful bound updates are beneficial in two ways. First, they indirectly lead to the construction of tighter underestimators as they affect the continuous variable bounds. Second, they allow a binary variable to be fixed and therefore decrease the number of combinations that potentially need to be explored. An interval-based strategy can be used to carry out binary variable bound updates. Given the current upper bound $\overline{f}^*$ on the global optimum solution, the feasible region F is defined by the constraints appearing in the nonconvex problem, a new constraint $f(\mathbf{x}) + \mathbf{x}^\top A_f \mathbf{y} + c_f^\top \mathbf{y} \le \overline{f}^*$, and the box $(\mathbf{x}, \mathbf{y}) \in [\mathbf{x}^L, \mathbf{x}^U] \times [\mathbf{y}^L, \mathbf{y}^U]$. Consider a variable $y_i \in \{0, 1\}$ whose bounds are being updated. The procedure above is used.

Algorithmic Procedure

The algorithmic procedure for the SMIN-αBB algorithm is as follows:

```
PROCEDURE SMIN-$\alpha$BB algorithm()
  Decompose functions in problem;
  Set tolerance $\epsilon$;
  Set $\underline{f}^* = \underline{f}^0 = -\infty$ and $\overline{f}^* = \overline{f}^0 = +\infty$;
  Initialize list of lower bounds $\{\underline{f}^0\}$;
  DO $\overline{f}^* - \underline{f}^* > \epsilon$
    Select node $k$ with smallest lower bound, $\underline{f}^k$, from list of lower bounds;
    Set $\underline{f}^* = \underline{f}^k$;
    ($Optional$) Update binary and continuous variable bounds;
    Select binary or continuous branching variable
    Partition to create new nodes;
    DO for each new node $i$
      Generate convex lower bounding MINLP;
      Find solution $\underline{f}^i$ of convex lower bounding MINLP;
      IF infeasible or $\underline{f}^i > \overline{f}^* + \epsilon$
        Fathom node;
      ELSE
        Add $\underline{f}^i$ to list of lower bounds;
        Find a solution $\overline{f}^i$ of nonconvex MINLP;
        IF $\overline{f}^i < \overline{f}^*$ THEN Set $\overline{f}^* = \overline{f}^i$;
    OD;
  OD;
  RETURN($\overline{f}^*$ and variables values at corresponding node);
END SMIN-$\alpha$BB algorithm;
```

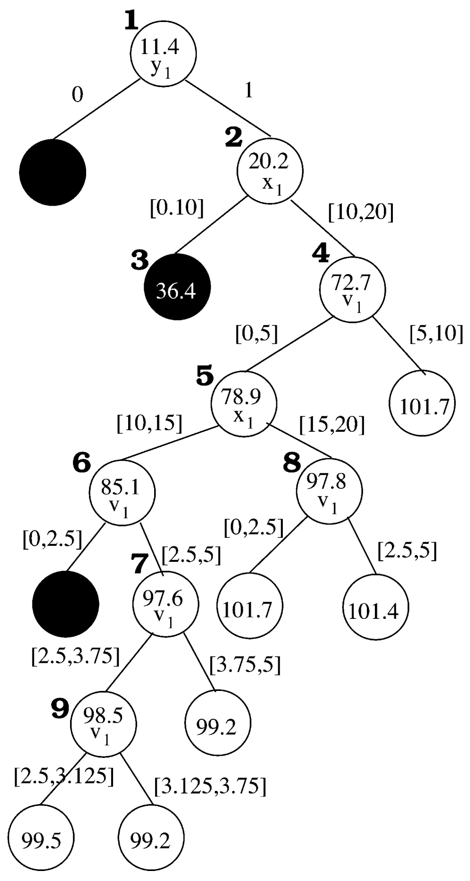
Pseudocode for the SMIN-αBB algorithm

In order to illustrate the algorithmic procedure, a small example proposed in [17] is used. It is a simple design problem where one of two reactors must be chosen to produce a given product at the lowest possible cost. It involves two binary variables, one for each reactor, and seven continuous variables. The formulation is:

$$\begin{cases} \min & 7.5y_1 + 5.5y_2 + 7v_1 + 6v_2 + 5x \\ \text{s.t.} & z_1 - 0.9\left(1 - e^{-0.5v_1}\right)x_1 = 0 \\ & z_2 - 0.8\left(1 - e^{-0.5v_2}\right)x_2 = 0 \\ & x_1 + x_2 - x = 0 \\ & z_1 + z_2 = 10 \\ & v_1 - 10y_1 \le 0 \\ & v_2 - 10y_2 \le 0 \\ & x_1 - 20y_1 \le 0 \\ & x_2 - 20y_2 \le 0 \\ & y_1 + y_2 = 1 \\ & 0 \le x_1, x_2 \le 20; \quad 0 \le z_1, z_2 \le 30 \\ & 0 \le v_1, v_2 \le 10; \quad 0 \le x \le 20 \\ & (y_1, y_2) \in \{0, 1\}^2 \end{cases}$$

Because of the linear participation of the binary variables, the SMIN-αBB algorithm is well-suited to solve this nonconvex MINLP. It identifies the global solution of 99.2 after nine iterations, when bound updates are performed at every iteration and branching takes place on the binary variables first. Branching variable selection takes place randomly for the binary variables and according to the term measures for the continuous variables. At the global solution, the binary variable values are $y_1 = 1$ and $y_2 = 0$. The steps of the algorithm are shown in Fig. 1. The boldface numbers next to the nodes indicate the order in which the nodes were explored. The lower bound is computed by solving a convex relaxation of the nonconvex problem is indicated inside each node, and the branching variable selected for the node is also specified. The domain to which this branching variable is restricted is displayed along each branch. A black node indicates the lower bounding problem was found infeasible and a shaded node is fathomed because its lower bound is greater than the current upper bound on the solution.

At the first node, the initial lower bound is 11.4 and an upper bound of 99.2 is found. The binary variable y_1 is selected as a branching variable. The region $y_1 = 0$ is infeasible and can therefore be fathomed (black node), while an improved lower bound is found for $y_1 = 1$. This latter region is therefore chosen for exploration at the second iteration. Variable bound updates reveal that $y_2 = 1$ is infeasible so that y_2 can be fixed to zero. Branch-



MINLP: Global Optimization with αBB, Figure 1
SMIN-αBB branch and bound tree

ing on the continuous variables may now begin. The first selected variable is x_1 and regions $0 \leq x_1 \leq 10$ and $10 \leq x_1 \leq 20$ are created. Since the left region has the lowest lower bound (36.4), it is examined at iteration 3. Variable bound updates show that this region is in fact infeasible and it is therefore eliminated without further processing. The algorithm proceeds to node 4 for which v_1 is selected as a branching variable. The right region, $5 \leq v_1 \leq 10$, is fathomed since it has a lower bound greater than 99.2. The algorithm progresses along the branch and bound until, at iteration 9, two nodes are left open with lower bounds of 99.2. This is within the accuracy required for this run so the procedure is terminated. One more iteration would reveal that the only global optimum lies in the right child of node 9.

The SMIN-αBB algorithm is especially effective for chemical process synthesis problem such as distillation network or heat exchanger network synthesis [1,3].

The GMIN-αBB Algorithm

The *GMIN-αBB algorithm* is designed to address the broad class of problems represented by

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{g}(\mathbf{x},\mathbf{y}) \leq 0 \\ & \mathbf{h}(\mathbf{x},\mathbf{y}) = 0 \\ & \mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U] \\ & \mathbf{y} \in [\mathbf{y}^L, \mathbf{y}^U] \cap \mathbb{N}^q \end{cases} \tag{2}$$

where $f(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$, and $\mathbf{h}(\mathbf{x}, \mathbf{y})$, are functions whose continuous relaxation is twice continuously differentiable.

The GMIN-αBB algorithm [2,3,7] extends the applicability of the standard branch and bound approaches for MINLPs [9,11,13,15,19,20] by making use of the αBB-algorithm. The most crucial characteristics of the algorithm are the branching strategy, the derivation of a valid lower bound on problem (2), and the variable bound update strategies.

Branching Variable Selection

Branching in the GMIN-αBB algorithm is carried out on the integer variables only. When it is a bisection, the partition takes place either at the midpoint of the range of the selected variable, or at the value of that variable at the solution of the lower bounding problem. It is also possible to branch on more than one variable at a given node, or to perform k-section on one of the variables. More than two children node may be created from a parent node when the structure of the problem is such that the bounds on a small fraction of the integer variables affect the bounds on many of the other variables in the problem. As in the SMIN-αBB algorithm, an integer variable is chosen randomly or according to branching priorities. An additional rule consists of selecting the most or least fractional variable at the solution of a continuous relaxation of the problem.

Generation of a Valid Lower Bound

A guaranteed lower bound on the global solution of the current node of the branch and bound tree is obtained by solving a continuous relaxation of the nonconvex MINLP at that node. When the integer variables that have not yet been fixed are allowed to vary continuously between their bounds, the problem becomes a nonconvex NLP. The validity of the lower bound can only be ensured if the global solution of this nonconvex NLP is identified or if a lower bound on this solution is found. On the other hand, when all integer variables have been fixed to integer values at a node, no additional partitioning of this node can take place and the global optimum solution of the nonconvex NLP is required to guarantee convergence of the GMIN-αBB. Based on these conditions, the αBB algorithm can be used as as subroutine to generate valid lower bounds:

- If at least one integer variable can be relaxed at the current node, run the αBB algorithm for a few iterations to obtain a valid lower bound on the global solution of the continuous relaxation *or* run the αBB algorithm to completion to obtain the global solution of the continuous relaxation.
- Otherwise, run the αBB algorithm to completion to obtain the global solution for the current node.

This strategy makes use of the convergence characteristics of the αBB algorithm to improve the performance of the GMIN-αBB algorithm. The rate of improvement of the lower bound on the global solution of a nonconvex NLP is usually very high at early iterations and then gradually tapers off. At later stages of an αBB run, the computationally expensive reduction of the gap between the bounds on the solution of the continuous relaxation does not result in a sufficiently significant increase in the lower bound to affect the performance of the GMIN-αBB algorithm and can therefore be bypassed.

Generation of a Valid Upper Bound

Because of the finite size of the branch and bound tree, it is not necessary to generate an upper bound on the nonconvex MINLP at each node in order to guarantee convergence of the GMIN-αBB algorithm. In the worst case, the integer variables are fixed at every node of the last level of the tree, and the solutions of the corresponding NLPs provide the upper bounds needed to identify the global optimum solution. However, upper bounds play a significant role in improving the convergence rate of the algorithm by allowing the fathoming of nodes whose lower bound is greater than the smallest upper bound and therefore reducing the final size of the branch and bound tree. An upper bound on the solution of a given node can be obtained in several ways. For example, if the solution of the continuous relaxation is integer-feasible, that is, all the relaxed integer variables have integer values at the solution, this solution is both a lower and an upper bound on the current node. If the αBB algorithm was run for only a few iterations and the relaxed integer variables are integer at the lower bound, they can be fixed to these integer values and the resulting nonconvex NLP can be solved locally to yield an upper bound on the solution of the node. Finally, a set of integer values satisfying the integer constraints can be used to construct a nonconvex NLP whose local solutions are upper bounds on the current node solution.

Variable Bound Updates

If the bounds on the integer variables at any given node can be tightened, the solution space can be significantly reduced due to the combinatorial nature of the problem. The allocation of computational resources for this purpose is therefore a potentially worthwhile investment. An optimization-based approach or an interval-based approach may be used to update the variable bounds. These approaches are similar to those developed for the αBB algorithm but they take advantage of the integrality of the variables. Thus, in the optimization approach, the lower or upper bound on variable y_i is improved by first relaxing the integer variables, and then solving the convex NLP

$$y^* = \begin{cases} \min \text{ or } \max_{\mathbf{x},\mathbf{y},\mathbf{w}} & y_i \\ \text{s.t.} & \breve{f}(\mathbf{x},\mathbf{y},\mathbf{w}) \leq \overline{f}^* \\ & \mathbf{C}(\mathbf{x},\mathbf{y},\mathbf{w}) \\ & \mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U] \\ & \mathbf{y} \in [\mathbf{y}^L, \mathbf{y}^U] \\ & \mathbf{w} \in [\mathbf{w}^L, \mathbf{w}^U] \end{cases} \tag{3}$$

where $\breve{f}(\mathbf{x},\mathbf{y},\mathbf{w})$ denotes the convex underestimator of objective function, $\overline{f}^*$ denotes the current best upper bound on the global optimum solution, $\mathbf{C}$ $(\mathbf{x},\mathbf{y},\mathbf{w})$ denotes the set of convexified constraints, and $\mathbf{w}$ is the set of new variables introduced during the convexification/relaxation procedure. Finally, the improved lower or upper bound is obtained by setting $y_i^L = \lceil y^* \rceil$ or $y_i^U = \lfloor y^* \rfloor$.

In the interval-based approach, an iterative procedure is followed based on an interval test which provides sufficient conditions for the infeasibility of the original constraints and the 'bound improvement constraint' $f(\mathbf{x}, \mathbf{y}) \leq \overline{f}^*$, given the relaxed region $(\mathbf{x}, \mathbf{y}) \in [\mathbf{x}^L, \mathbf{x}^U] \times [\mathbf{y}^L, \mathbf{y}^U]$. This set of constraints defines a region denoted by F. The procedure to improve the lower (upper) bound on variable y_i is as follows:

```
PROCEDURE interval-based bound update()
    Set initial bounds $L = y_i^L$ and $U = y_i^U$;
    Set iteration counter $k = 0$;
    Set maximum number of iterations $K$;
    DO $k < K$ and $L \neq U$
        Compute 'midpoint' $M = \lfloor (U + L)/2 \rfloor$;
        Set left region
            $\{(\mathbf{x}, \mathbf{y}) \in F : y_i \in [L, M]\}$;
        Set right region
            $\{(\mathbf{x}, \mathbf{y}) \in F : y_i \in [M + 1, U]\}$;
        Test interval feasibility of left(right) region;
        IF feasible,
            Set $U = M$ $(L = M)$;
        ELSE
            Test interval feasibility of right(left)
            region;
            IF feasible,
                Set $L = M$ $(U = M)$;
            ELSE
                IF $k = 0$,
                    RETURN(infeasible node);
                ELSE
                    Set $L = U$ $(U = L)$;
                    Set $U = y_i^U$ $(L = y_i^L)$;
        Set $k = k + 1$;
    OD;
    RETURN($y_i^L = L$ $(y_i^U = U)$);
END interval-based bound update;
```

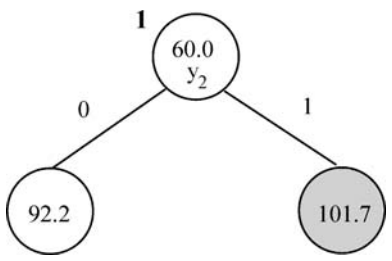
Interval-based bound update procedure

The variable bound tightening is performed before calling the αBB algorithm to obtain a lower bound on the solution of the current node. In many cases, during an αBB run, variable bound updates are also used to improve the quality of the generated lower bounds. Although the αBB algorithm treats the **y** variables as continuous, the bound update strategy within the αBB algorithm may be modified to account for the true nature of these variables. A larger reduction in the solution space can be achieved by adopting one of the *integer* bound update strategies described here for the relaxed **y** variables. This more stringent approach leads to a lower bound which is not necessarily a valid lower bound on the continuous relaxation, but which is always a lower bound on the global solution of the nonconvex MINLP.

The overall algorithmic procedure for the GMIN-αBB algorithm is shown below:

```
PROCEDURE GMIN-$\alpha$BB algorithm()
    Set tolerance $\epsilon$;
    Set $\underline{f}^* = \underline{f}^0 = -\infty$ and $\overline{f}^* = \overline{f}^0 = +\infty$;
    Initialize list of lower bounds $\{\underline{f}^0\}$;
    DO $\overline{f}^* - \underline{f}^* > \epsilon$
        Select node $k$ with smallest lower bound, $\underline{f}^k$,
        from list of lower bounds;
        Set $\underline{f}^* = \underline{f}^k$;
        $(Optional)$ Update $\mathbf{y}$ variable bounds;
        Select integer branching variable(s);
        Create new nodes by branching;
        DO for each new node $i$
            Obtain lower bound $\underline{f}^i$ on node
                IF all integer variables are fixed,
                    Find global solution $\underline{f}^i$ of nonconvex
                    NLP with $\alpha$BB algorithm;
                ELSE
                    Relax integer variables;
                    Run $\alpha$BB algorithm to completion or
                    for a few iterations to get $\underline{f}^i$
                        $(Optional)$ Use $integer$ bound
                        updates on $\mathbf{y}$ variables;
            IF $\underline{f}^i > \overline{f}^i + \epsilon$, THEN Fathom node;
            ELSE
                Add $\underline{f}^i$ to list of lower bounds;
                $(Optional)$ Obtain upper bound $\overline{f}^i$ on
                nonconvex MINLP;
                IF $\overline{f}^i < \overline{f}^*$ THEN Set $\overline{f}^* = \overline{f}^i$;
        OD;
    OD;
    RETURN($\overline{f}^*$ and variables values at corres-
    ponding node);
END GMIN-$\alpha$BB algorithm;
```

Pseudocode for the GMIN-αBB algorithm



MINLP: Global Optimization with αBB, Figure 2
GMIN-αBB branch and bound tree

The algorithmic procedure for the GMIN-αBB algorithm is illustrated using the same example as for the SMIN-αBB algorithm. The branch and bound tree is shown in Fig. 2, using the same notation as previously.

At the first node, the continuous relaxation of the nonconvex MINLP is solved for 10 αBB iterations to yield a lower bound of 60. No upper bound is found. Next, the binary variable y_2 is chosen for branching and the continuous relaxation of the problem with $y_2 = 0$ is solved. A lower bound of 92.2 is found as the global solution to this nonconvex NLP. In addition, this solution is integer feasible and therefore provides an upper bound on the global optimum solution of the nonconvex MINLP. The region $y_2 = 1$ is then examined and the global solution of the NLP is found to be 101.7 after 10 αBB iterations. This node can therefore be fathomed and the procedure terminated.

The GMIN-αBB algorithm has been used to solve nonconvex MINLPs involving nonconvex terms in the integer variables and some mixed nonconvex terms. Branching priorities combined with variable bound updates and a small number of αBB iterations for relaxed nodes allow the identification of the global optimum solution after the exploration of a small fraction of the maximum number of nodes and with small CPU requirements. In particular, the algorithm has been used on a pump network synthesis problem [2,3]. Some nonconvex integer problems have also been tackled by the same approach. For instance, the minimization of trim loss, a problem taken from the paper cutting industry, has also been addressed for medium order sizes [3].

Conclusions

The αBB algorithm for nonconvex NLPs can be incorporated within more general frameworks to address broad classes of nonconvex MINLPs. One extension of the algorithm is the SMIN-αBB algorithm which identifies the global optimum solution of problems in which binary variables participate in linear or mixed-bilinear terms and continuous variables appear in twice continuously differentiable functions. The partitioning of the solution space takes place in both the continuous and binary domains. The GMIN-αBB algorithm is designed to locate the global optimum solution of problems involving integer and continuous variables in functions whose continuous relaxation is twice continuously differentiable. The algorithm is similar to traditional branch and bound algorithms for mixed integer problems in that branching occurs on the integer variables only and a continuous relaxation of the problem is constructed during the bounding step. It uses the αBB algorithm for the efficient and rigorous generation of lower bounds. Both algorithms are widely applicable and have been successfully tested on a variety of medium-size nonconvex MINLPs.

See also

- ▶ αBB Algorithm
- ▶ Chemical Process Planning
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Convex Envelopes in Optimization Problems
- ▶ Disjunctive Programming
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ Global Optimization in Batch Design Under Uncertainty
- ▶ Global Optimization in Generalized Geometric Programming
- ▶ Global Optimization of Heat Exchanger Networks
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization in Phase and Chemical Reaction Equilibrium
- ▶ Interval Global Optimization
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control

- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Mass and Heat Exchanger Networks
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Heat Exchanger Network Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming
- ▶ Reformulation-linearization Methods for Global Optimization
- ▶ Smooth Nonlinear Nonconvex Optimization

References

1. Adjiman CS, Androulakis IP, Floudas CA (1997) Global optimization of MINLP problems in process synthesis. Comput Chem Eng 21:S445–S450
2. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, αBB, for general twice-differentiable constrained NLPs -II. Implementation and computational results. Comput Chem Eng 22:1159
3. Adjiman CS, Androulakis IP, Floudas CA (2000) Global optimization of mixed-integer nonlinear problems. Comput Chem Eng 46:1769–1797
4. Adjiman CS, Androulakis IP, Maranas CD, Floudas CA (1996) A global optimization method, αBB, for process design. Comput Chem Eng 20:S419–S424
5. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, αBB, for general twice-differentiable constrained NLPs -I. Theoretical advances. Comput Chem Eng 22:1137
6. Adjiman CS, Floudas CA (1996) Rigorous convex underestimators for twice-differentiable problems. J Global Optim 9:23–40
7. Adjiman CS, Schweiger CA, Floudas CA (1998) Mixed-integer nonlinear optimization in process synthesis. In: Du D-Z, Pardalos PM (eds) Handbook Combinatorial Optim. Kluwer, Dordrecht, pp 429–452
8. Androulakis IP, Maranas CD, Floudas CA (1995) αBB: A global optimization method for general constrained nonconvex problems. J Global Optim 7:337–363
9. Beale EML (1977) Integer programming. In: The State of the Art in Numerical Analysis. Acad. Press, New York, pp 409–448
10. Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numer Math 4:238
11. Borchers B, Mitchell JE (1991) An improved branch and bound algorithm for mixed integer nonlinear programs. Techn. Report Renssellaer Polytechnic Inst. 200
12. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307–339
13. Floudas CA (1995) Nonlinear and mixed integer optimization: Fundamentals and applications. Oxford Univ. Press, Oxford
14. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
15. Gupta OK, Ravindran R (1985) Branch and bound experiments in convex nonlinear integer programing. Managem Sci 31:1533–1546
16. Kocis GR, Grossmann IE (1987) Relaxation strategy for the structural optimization of process flow sheets. Industr Eng Chem Res 26:1869
17. Kocis GR, Grossmann IE (1989) A modelling and decomposition strategy for the MINLP optimization of process flowsheets. Comput Chem Eng 13:797–819
18. Maranas CD, Floudas CA (1994) Global minimum potential energy conformations of small molecules. J Global Optim 4:135–170
19. Ostrovsky GM, Ostrovsky MG, Mikhailow GW (1990) Discrete optimization of chemical processes. Comput Chem Eng 14:111
20. Quesada I, Grossmann IE (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Comput Chem Eng 16:937–947

MINLP: Heat Exchanger Network Synthesis

KEMAL SAHIN, KORHAN GÜRSOY, AMY CIRIC
Department Chemical Engineering,
University Cincinnati, Cincinnati, USA

MSC2000: 90C90

Article Outline

Keywords

MINLP; HEN synthesis; Network synthesis

Heat exchanger network synthesis problems arise in chemical process design when the heat released by hot process streams is used to satisfy the demands of cold process streams. These problems have been the subject of an intensive research effort, and over 400 publications have been written in the area. See [7,8,9] for reviews of the area, and [1,3] for detailed analysis of HEN synthesis.

The discovery by T. Umeda et al. [11] of a thermodynamic pinch point that limits heat integration in a heat exchanger network led to much of this research effort. They showed that setting minimum temperature approach, ΔT_{min}, places a lower bound on the utility consumption in a heat exchanger network and decomposed a heat exchanger network into independent subnetworks. This enables the heat exchanger network synthesis problem to be decomposed into four subproblems. The first subproblem finds the appropriate minimum temperature approach, the second subproblem minimizes the utility consumption, the third subproblem finds the minimum number of matches and identifies the matches and their heat duty, and the fourth finds and optimizes the actual network structure.

See [5] for a systematic scheme for solving these problems sequentially. First, the utility consumption is minimized using the linear programming (LP) transshipment model approach of [10]. Second, a set of process matches and their heat duties that minimize the total number of units is found with the mixed integer linear programming (MILP) strategy of [10]. Then, the network structure is found [5] by optimizing a superstructure that contains all possible network configurations embedded within it using a nonlinear programming (NLP) problem. When there is more than one combination of matches and heat duties that satisfies the minimum unit criterion, the best combination is found by exhaustive enumeration. The minimum temperature approach is optimized with a golden section search that solves all three of these optimization problems at each iteration.

In the late 1980s it was found, [4,12], that better network designs could be obtained by solving some of the heat exchanger network design subproblems simultaneously. C.A. Floudas and A.R. Ciric [4] combined the MILP stream matching problem with the NLP superstructure optimization problem formulated in [5], creating a mixed integer nonlinear programming problem (MINLP) that avoided the exhaustive search through all combinations of matches that minimize the number of units. In 1990, they [2] formulated the entire heat exchanger network design problem as a MINLP. The solution of this problem yields the optimal temperature approach, utility level, process matches, heat duties, and network structure, eliminating the need for a global section search for the optimum minimum temperature approach.

T.F. Yee and I.E. Grossmann [12] used a smaller superstructure proposed in [6] that embodies a sequential-parallel network structure to formulate an alternative MINLP for heat exchanger network synthesis. The solution of this MINLP yielded the utility consumption, matches and network structure and heat exchanger areas.

Problem Statement

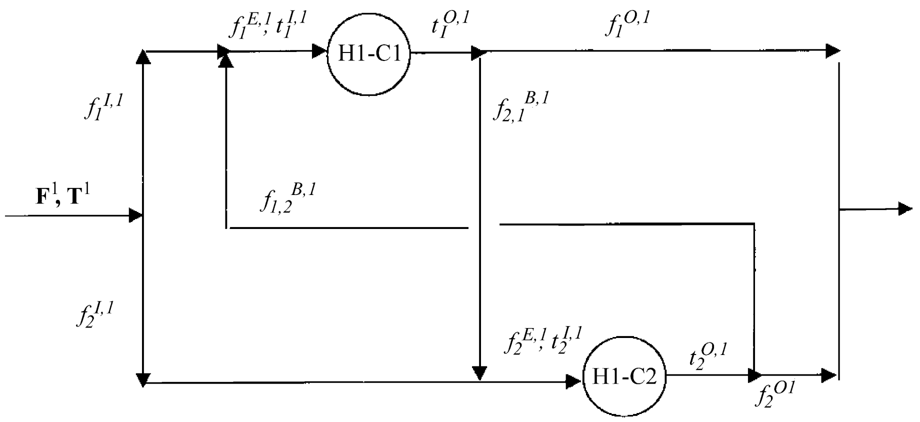
This article will explore two mixed integer nonlinear programming problems in heat exchanger network synthesis: combined match-network optimization and heat exchanger network synthesis without decomposition. The synthesis without decomposition problem can be stated as follows:

Given:

1) A set of hot process streams and hot utilities $i \in H$, their inlet and outlet temperatures $\mathbf{T}^i$, $\mathbf{T}^{O,i}$, and heat capacity flow rates $\mathbf{F}^i$;
2) A set of cold process streams and cold utilities $j \in C$, their inlet and outlet temperatures $\mathbf{T}^j$, $\mathbf{T}^{O,j}$, and heat capacity flow rates $\mathbf{F}^j$; and
3) Overall heat transfer coefficients $\mathbf{U}_{ij}$.

Determine:

A) The stream matches (ij), the heat duty $\mathbf{Q}_{ij}$ of match (ij), and the heat exchanger area $\mathbf{A}_{ij}$ of match (ij);
B) the piping structure for each stream in the network; and
C) the temperature and flowrate within each pipe of the network.



MINLP: Heat Exchanger Network Synthesis, Figure 1
A superstructure for one hot stream exchanging heat with two cold streams

In the match-network problem, one is also given

- the level of each utility; and
- a minimum temperature approach $\Delta T_{\min}$.

These problems can be solved using mixed integer nonlinear programming. The development and application of these approaches is described in more detail below.

Heat Exchanger Network Superstructures

Mixed integer nonlinear programming approaches to these problems begin with a superstructure that contains many alternative designs embedded within it. Two superstructures are particularly interesting.

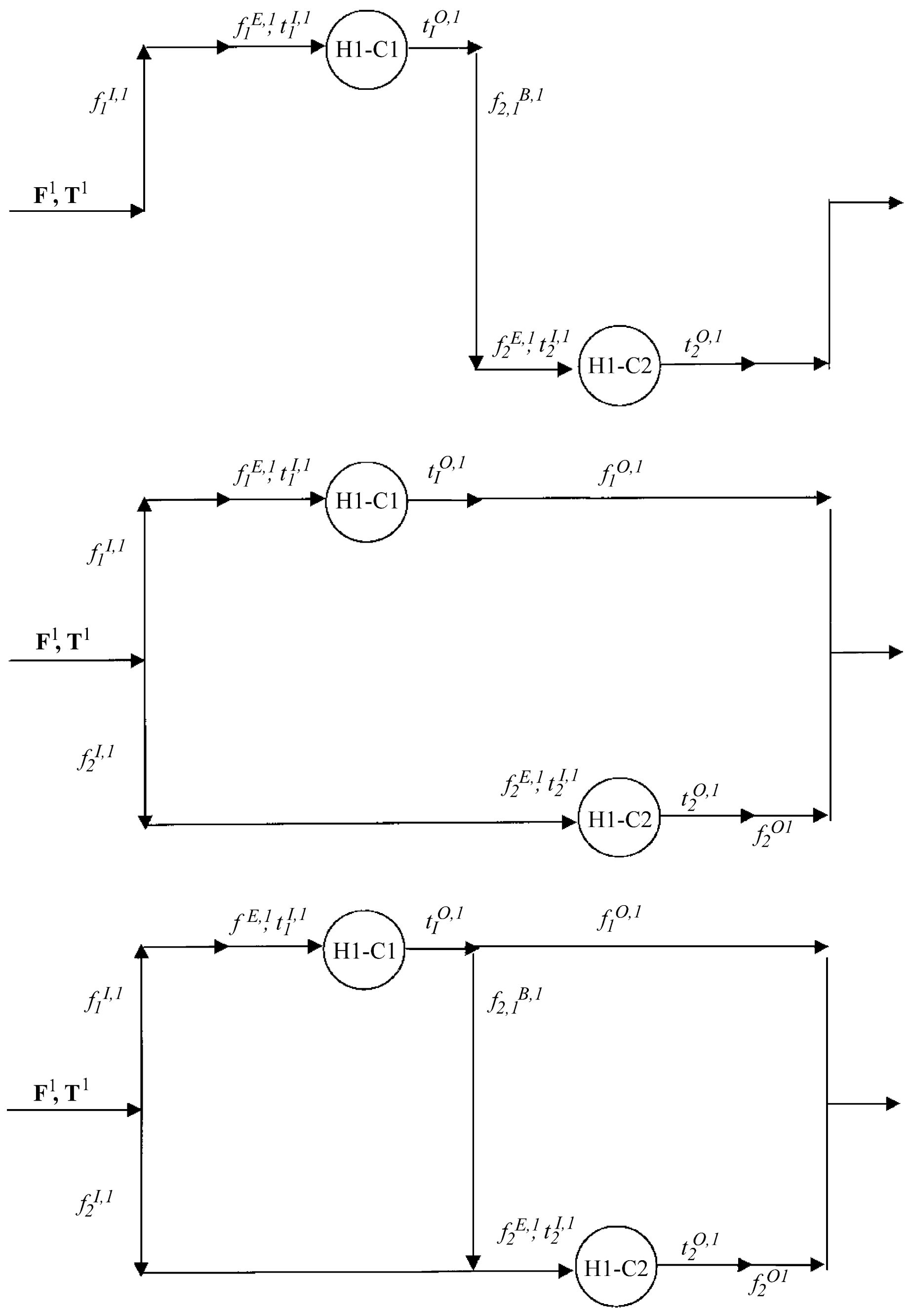
Figure 1 shows a superstructure of a hot stream, above the thermodynamic pinch point, that may exchange heat with two cold streams [5]. Notice that the stream can be piped in series, in parallel, and in split-mix-bypass configurations, as shown in Fig. 2. As we shall see, this richness leads to nonconvex constraints in the MINLP. The first network superstructure is created by constructing similar structures for every other stream above the pinch point.

Notice that in this subnetwork, streams H1 and C1, and all other pairs of hot and cold streams, can exchange heat no more than once. H1 and C1 may exchange heat again in the subnetwork below the thermodynamic pinch point. The thermodynamic pinch point has partitioned the temperature range into two intervals, and in each interval, individual process streams can only exchange heat once.

One could increase the number of times two streams can exchange heat by partitioning the temperature range further. This is the basic strategy behind the second superstructure [6,12] shown in Fig. 3. Here, the temperature range has been partitioned into many intervals, or stages. Within any particular stage, each hot stream may exchange heat with each cold stream; multiple intervals allow any particular match to take place many times in the network. Unlike the first superstructure, each stream in each stage is piped in a parallel configuration, and the inlet and outlet temperature of each parallel line is fixed by the temperature interval. Series piping structures arise when a stream exchanges heat only once per interval. The superstructure does not contain split-mix-bypass or series-parallel structures, but as we shall see that in exchange the nonconvex constraints that arise from the first superstructure have been eliminated.

Mathematical Models for HEN Synthesis using MINLPs

MINLP models of heat exchanger network synthesis arise when the process stream matches are selected while simultaneously optimizing the heat exchanger network; the former is a discrete decision modeled with integer variables, the latter, a nonlinear optimization problem. In this paper, we refer to this as the *match-network problem*. MINLPs may also be used to formulate an optimization problem that simultaneously minimizes the utility consumption, selects the stream matches, and optimizes the network layout, in *heat exchanger network synthesis without decomposition*.

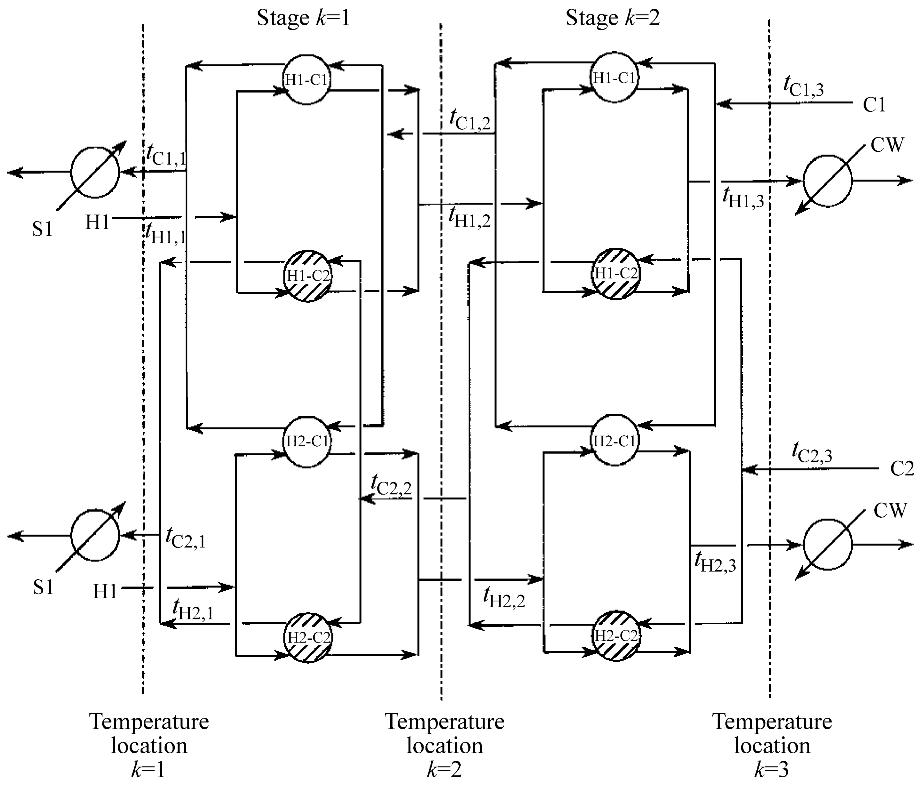


MINLP: Heat Exchanger Network Synthesis, Figure 2
Stream piping configurations embedded in the superstructure shown in Fig. 1

Match-Network Problem

The MINLP model of the match-network problem has three components: a *transshipment model* [10] that identifies feasible process stream matches and their heat duties, a *superstructure model* of all possible network structures, and an *objective function*.

The transshipment model partitions the temperature range into $t = 1, \ldots, T$ temperature intervals, using the inlet and outlet temperatures of the streams and



MINLP: Heat Exchanger Network Synthesis, Figure 3
Two-stage superstructure

the temperature interval approach temperature (TIAT). Hot streams release heat into the temperature intervals, where it either flows to the cold streams in the same interval or cascades down to the next colder interval. The binary variable Y_{ij} denotes the existence of a match between hot stream i and cold stream j, where heat loads are q_{ij} and Q_{ij}, and heat residuals are R_k. The model is composed by the following constraints:

$$\begin{cases} \sum\limits_{j\in C_t} q_{ijt} + R_{i,t} - R_{i,t-1} = Q^H_{it}, \quad i \in H, \quad j \in C_i, \\ \sum\limits_{i\in R_j} q_{ijt} = Q^C_{jt}, j \in C, \quad t = 1, \dots, T, \\ \sum\limits_{t=1}^{T} q_{ijt} = Q_{ij}, \quad i \in H, \quad j \in C_i, \\ Q_{ij} - UY_{ij} \le 0, \quad i \in H, \quad j \in C_i, \\ \sum\limits_{i\in H} \sum\limits_{j\in C_i} Y_{ij} \le N_{\max}. \end{cases}$$

The first two constraints in the transshipment model are the energy balances for each temperature interval. The total heat load in a match is given by the third constraint. The fourth constraint bounds the heat load using the binary variable Y_{ij} and a large fixed constant U. The last constraint in the above model puts an upper bound on the number of existing matches, which is the maximum number of units.

The second part of the match-network synthesis model is the hyperstructure topology model, which consists of mass and energy balances for the mixers and splitters, feasibility constraints, utility load constraint and bounds on the flow rate heat-capacities.

Mass balances for the splitters at the inlet of the superstructure:

$$\sum_{k'} f^{I,k}_{k'} = F^k, \quad k \in \text{HCT}.$$

Here, HCT is the set of all process streams and utilities. Mass balances for the mixers at the inlets of the exchangers:

$$f^{I,k}_{k'} + \sum_{k''} f^{B,k}_{k',k''} - f^{E,k}_{k'} = 0, \quad k', k \in \text{HCT}.$$

Mass balances for the splitters at the outlets of the exchangers:

$$f_{k'}^{O,k} + \sum_{k''} f_{k'',k'}^{B,k} - f_{k'}^{E,k} = 0, \quad k', k \in \mathrm{HCT}.$$

Energy balances for the mixers at the inlets of the exchangers:

$$T^k f_{k'}^{I,k} + \sum_{k''} f_{k',k''}^{B,k} t_{k'}^{O,k} - f_{k'}^{E,k} t_{k'}^{I,k} = 0,$$

$k', k \in \mathrm{HCT}$.

Energy balances over the heat exchangers:

$$Q_{ij} - f_j^{E,i}\left(t_j^{I,i} - t_j^{O,i}\right) = 0, \quad i \in H, \quad j \in C,$$
$$Q_{ij} - f_i^{E,j}\left(t_i^{O,j} - t_i^{I,j}\right) = 0.$$

The minimum temperature approach between a hot stream and a cold stream:

$$t_j^{I,i} - t_i^{O,j} \geq \Delta T_{\min},$$
$$t_j^{O,i} - t_i^{I,j} \geq \Delta T_{\min}.$$

Logical relations between the heat-capacity flow rates and the existence of a match:

$$f_j^{E,i} - F^i Y_{ij} \leq 0,$$
$$f_i^{E,j} - F^j Y_{ij} \leq 0.$$

Lower bounds on the heat-capacity flow rates through the exchanger:

$$f_j^{E,i} - \frac{Q_{ij}}{\Delta T_{ij,\max}} \geq 0,$$
$$f_i^{E,j} - \frac{Q_{ij}}{\Delta T_{ij,\max}} \geq 0,$$

where $\Delta T_{ij,\max}$ equals $T^i - T^j$. Lastly, the objective function minimizes the total investment cost:

$$\min \sum_{i \in H} \sum_{j \in C} \alpha \left(\frac{Q_{ij}}{U_{ij} \frac{t_j^{I,i} - t_i^{O,j} - t_j^{O,i} + t_i^{I,j}}{\ln \frac{t_j^{I,i} - t_i^{O,j}}{t_j^{O,i} - t_i^{I,j}}}} \right)^{\beta} Y_{ij}.$$

The model is a mixed integer nonlinear programming (MINLP) problem, as the objective function and the energy balances are nonlinear, and the decision variables Y_{ij} are binary. Notice that the energy balances are bilinear, creating a nonconvex feasible region.

MINLP: Heat Exchanger Network Synthesis, Table 1
Stream data for example problem

Stream	$T_{\text{in}}(C)$	$T_{\text{out}}(C)$	$FC_p(kW/C)$
H1	500	320	6
H2	480	380	4
H3	460	360	6
H4	380	360	20
H5	380	320	12
C1	290	660	18
F	700	700	
CW	300	320	

$U = 1.0kW/(m^2C)$
Annual cost$=1200A^{0.6}$ for all exchangers
$C_s = 140\$/kW$
$C_{cw} = 10\$/kW$

MINLP: Heat Exchanger Network Synthesis, Table 2
Match data for example problem; pseudo-pinch method [2]

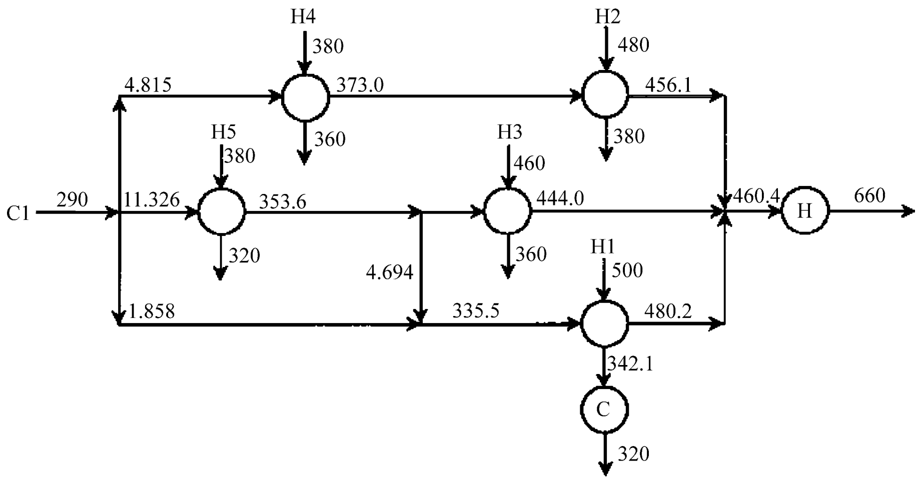
Match	$Q(kW)$	$A(m^2)$
H1-C1	948.454	79.391
H1-CW	131.546	6.280
H2-C1	400.000	29.057
H3-C1	600.000	57.488
H4-C1	400.000	14.880
H5-C1	720.000	25.509
S-C1	3591.546	32.112

Heat Exchanger Network Synthesis Without Decomposition

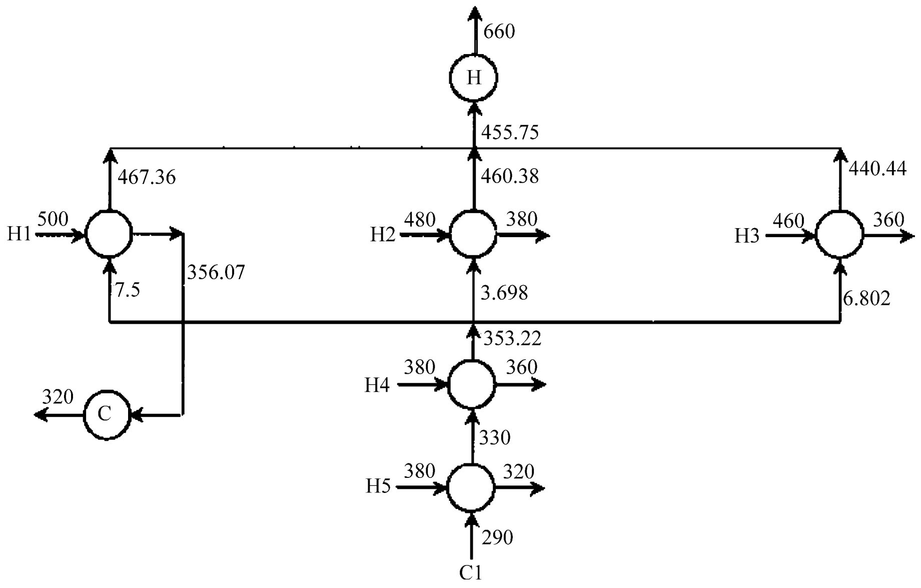
MINLP models that optimize utility consumption as well as process matches, heat duties, and network configurations can also be formulated. See [2] and [12] for pseudopinch approaches that set the TIAT to a small value and lets heat flow across the pinch. A strict decomposition at the pinch can also be maintained by letting TIAT vary, and using integer variables to model the changing structure of the temperature cascade.

Example 1 These techniques are demonstrated with a problem given in both [12] and [2]. The problem consists of two hot streams, two cold streams, one hot utility (steam), and one cold utility (cooling water). The stream data is given in Table 1.

Using the pseudopinch method with TIAT = 1C and $\Delta T_{\min} = 0.5C$, and allowing HRAT to vary between 1C



MINLP: Heat Exchanger Network Synthesis, Figure 4
Optimal network configuration for example problem; pseudopinch [2]



MINLP: Heat Exchanger Network Synthesis, Figure 5
Optimal network configuration for example problem; simultaneous approach [12]

and 30C, Ciric and Floudas [2] formulated the problem as a MINLP problem and solved it using the generalized Benders decomposition algorithm. The optimal network configuration is pictured in Fig. 3. The network consumes 3592.4kW of steam and 1312.4kW of cooling water, the HRAT is 8.42C. The annual cost of the network is $571,080. The match data of this solution is given in Table 2. Yee and Grossmann [12] used the same problem to demonstrate the simultaneous optimization approach. The problem is again formulated as a MINLP problem. The optimal network configuration is given in Fig. 4. The annual cost of this network is $576,640. HRAT is 13.1$C$. The match data of this network is given in Table 3.

Conclusions

Mixed integer nonlinear programming offer a powerful approach to heat exchanger network synthesis. Using these techniques, stream matching, the combina-

MINLP: Heat Exchanger Network Synthesis, Table 3
Match data for example problem; simultaneous approach [12]

Match	$Q(kW)$	$A(m^2)$
S-C1	3676.4	32.6
H1-C1	863.6	64.1
H2-C1	400.0	17.1
H3-C1	600.0	47.0
H4-C1	400.0	13.8
H1-CW	216.4	7.9
H5-C1	720.0	18.4

torial component of heat exchanger network synthesis, can be performed while simultaneously minimizing the utility consumption and selecting the cost-optimal heat exchanger network configuration. Merging these tasks leads to more cost-effective stream matches and lower exchanger costs.

See also

- ▶ Chemical Process Planning
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ Global Optimization of Heat Exchanger Networks
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Mass and Heat Exchanger Networks
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Heat Exchanger Network Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming

References

1. Biegler LT, Grossmann IE, Westerberg AW (1997) Systematic methods of chemical process design. Prentice-Hall, Englewood Cliffs, NJ
2. Ciric AR, Floudas CA (1990) Heat exchanger network synthesis without decomposition. Comput Chem Eng 15:385–396
3. Floudas CA (1995) Nonlinear and mixed-integer optimization. Oxford Univ. Press, Oxford
4. Floudas CA, Ciric AR (1989) Strategies for overcoming uncertainties in heat exchanger network synthesis. Comput Chem Eng 13(10):1133
5. Floudas CA, Ciric AR, Grossmann IE (1986) Automatic synthesis of optimum heat exchanger network configurations. AIChE J 32:276
6. Grossmann IE, Sargent RWH (1978) Optimum design of heat exchanger networks. Comput Chem Eng 2(1):1–7
7. Gundersen T, Naess L (1988) The synthesis of cost optimal heat exchanger networks: An industrial review of the state-of-the-art. Comput Chem Eng 12(6):503
8. Jezowski J (1994) Heat exchanger network grassroot and retrofit design: The review of the state-of-the-art: Part I. Hungarian J Industr Chem 22:279–294
9. Jezowski J (1994) Heat exchanger network grassroot and retrofit design: The review of the state-of-the-art: Part II. Hungarian J Industr Chem 22:295–308
10. Papoulias SA, Grossmann IE (1983) A structural optimization approach in process synthesis - II: Heat recovery networks. Comput Chem Eng 7:707
11. Umeda T, Harada T, Shiroko K (1979) A thermodynamic approach to the structure in chemical processes. Comput Chem Eng 3:373
12. Yee TF, Grossmann IE (1990) Simultaneous optimization models for heat integration - II: Heat exchanger network synthesis. Comput Chem Eng 14(10):1165

MINLP: Logic-based Methods

IGNACIO E. GROSSMANN
Carnegie Mellon University, Pittsburgh, USA

MSC2000: 90C10, 90C09, 90C11

Article Outline

Keywords

Generalized disjunctive programming; Disjunctive programming; Outer approximation method; generalized Benders decomposition; Mixed integer programming

There has been an increasing trend to representing linear and nonlinear discrete optimization problems by models consisting of algebraic constraints, logic disjunctions and logic relations ([1,7,8]). For instance, a mixed integer program can be formulated as a generalized disjunctive program as has been shown in [5]:

$$
\text{(DP1)}\quad \begin{cases}
\min & Z = \sum_k c_k + f(x) \\
\text{s.t.} & g(x) \le 0 \\
& \bigvee_{i\in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik}(x) \le 0 \\ c_k = \gamma_{ik} \end{bmatrix}, \\
& k \in SD, \\
& \Omega(Y) = \text{true} \\
& x \in \mathbb{R}^n, \quad c \in \mathbb{R}^m, \\
& Y \in \{\text{true}, \text{false}\}^m,
\end{cases}
$$

in which Y_{ik} are the boolean variables that establish whether a given term in a disjunction is true ($h_{ik}(x) \le 0$), while $\Omega(Y)$ are logical relations assumed to be in the form of propositional logic involving only the boolean variables. Y_{ik} are auxiliary variables that control the part of the feasible space in which the continuous variables, x, lie, and the variables c_{ik} represent fixed charges which are set to a value γ_{ik} if the corresponding term of the disjunction is true. Finally, the logical conditions, $\Omega(Y)$, express relationships between the disjunctive sets. In the context of optimal synthesis of process networks, the disjunctions in (DP1) typically arise for each unit i in the following form:

$$
\begin{bmatrix} Y_i \\ h_i(x) \le 0 \\ c_i = \gamma_i \end{bmatrix} \bigvee \begin{bmatrix} \neg Y_i \\ B^i x = 0 \\ c_i = 0 \end{bmatrix}, \quad i \in I, \tag{1}
$$

in which the inequalities h_i apply and a fixed cost γ_i is incurred if the unit is selected (Y_i); otherwise ($\neg Y_i$) there is no fixed cost and a subset of the x variables is set to zero with the matrix B^i. An important advantage of the above modeling framework is that there is no need to introduce artificial parameters for the 'big-M' constraints that are normally used in MINLP to model disjunctions.

M. Turkay and I.E. Grossmann [9] proposed a logic version of the outer approximation algorithm for MINLP [3] for solving problem (DP1), and in which the disjunctions are given as in equation (1), and all the functions are assumed to be convex. The algorithm consists of solving a sequence of NLP subproblems and master problems, which are as follows.

For fixed values of the boolean variables, $Y_{\widehat{i}k} = \text{true}$ and Y_{ik} = false for $\widehat{i} \neq i$, the corresponding NLP subproblem is as follows:

$$
\text{(NLPD)}\quad \begin{cases}
\min & Z = \sum_k c_k + f(x) \\
\text{s.t.} & g(x) \le 0 \\
& \text{for } Y_{\widehat{i}k} = \text{true}: \begin{cases} h_{ik}(x) \le 0 \\ c_k = \gamma_{ik} \end{cases} \\
& \text{for } Y_{ik} = \text{false}: \begin{cases} B^i x = 0 \\ c_k = 0 \end{cases} \\
& k \in SD, \\
& x \in \mathbb{R}^n, \quad c_i \in \mathbb{R}^1.
\end{cases}
$$

Note that for every disjunction $k \in SD$ only constraints corresponding to the boolean variable $Y_{\widehat{i}k}$ that is true are imposed. Also, fixed charges γ_{ik} are only applied to these terms. Assuming that K subproblems (NLPD) are solved in which sets of linearizations $l = 1, \dots, K$ are generated for subsets of disjunction terms $L(ik) = \{l: Y^l_{ik} = \text{true}\}$, one can define the following *disjunctive OA master problem*:

$$
\text{(MDP1)}\quad \min Z = \sum_k c_k + f(x)
$$

such that

$$
\begin{aligned}
&\alpha \ge f(x^l) + \nabla f(x^l)^\top (x - x^l), \\
&g(x^l) + \nabla g(x^l)^\top (x - x^l) \le 0, \\
&l = 1, \dots, L, \\
&\bigvee_{i\in D_k} \begin{bmatrix} Y_{ik} & \\ & h_{ik}(x^\ell) + \nabla h_{ik}(x^\ell)^\top (x - x^\ell) \le 0 \\ & c_k = \gamma_{ik} \end{bmatrix},
\end{aligned}
$$

$$k \in SD,$$
$$\Omega(Y) = \text{true},$$
$$\alpha \in \mathbb{R}, \quad x \in \mathbb{R}^n, \quad c \in \mathbb{R}^m,$$
$$Y \in \{\text{true}, \text{false}\}^m.$$

It should be noted that before applying the above master problem it is necessary to solve various subproblems (NLPD) so as to produce at least one linear approximation of each of the terms in the disjunctions. Selecting the smallest number of subproblems amounts to the solution of a set covering problem, which is of small size and easy to solve [9].

The above problem (MDP1) can be solved by the methods described in [1] and [7]. It is also interesting to note that for the case of process networks Turkay and Grossmann [9] have shown that if the convex hull representation of the disjunctions in (1) is used in (MDP1), then assuming $B^i = I$ and converting the logic relations $\Omega(Y)$ into the inequalities $Ay \leq a$, leads to the MILP problem,

$$\text{(MIPDF)} \quad \min Z = \sum_k c_k + f(x)$$

such that

$$\alpha \geq f(x^l) + \nabla f(x^l)^\top (x - x^l),$$
$$g(x^l) + \nabla g(x^l)^\top (x - x^l) \leq 0,$$
$$l = 1, \dots, L,$$
$$\nabla_{x_{z_i}} h_i(x^l)^\top x_{z_i} + \nabla_{x_{N_i}} h_i(x^l)^\top x^1_{N_i}$$
$$\leq \left[-h_i(x^\ell) + \nabla_x h_i(x^\ell)^\top x^\ell \right] y_i,$$
$$\ell \in K^i_L, \quad i \in I,$$
$$x_{N_i} = x^1_{N_i} + x^2_{N_i},$$
$$0 \leq x^1_{N_i} \leq x^U_{N_i} y_i,$$
$$0 \leq x^2_{N_i} \leq x^U_{N_i}(1 - y_i),$$
$$Ay \leq a,$$
$$x \in \mathbb{R}^n, \quad x^1_{N_i} \geq 0, x^2_{N_i} \geq 0,$$
$$y\{0, 1\}^m,$$

where the vector x is partitioned into the variables for each disjunction i according to the definition of the matrix B^i. The linearization set is given by $K^i_L = \{\ell\colon Y\ell_i = \text{true}, \ell = 1, \dots, L\}$ that denotes the fact that only a subset of inequalities were enforced for a given subproblem ℓ. It is interesting to note that the logic-based outer approximation algorithm represents a generalization of the modeling/decomposition strategy [5] for the synthesis of process flowsheets.

Turkay and Grossmann [9] have also shown that while a logic-based *generalized Benders method* [4] cannot be derived as in the case of the OA algorithm, one can exploit the property for MINLP problems that performing one Benders iteration [2] on the MILP master problem of the OA algorithm, is equivalent to generating a generalized Benders cut. Therefore, a logic-based version of the generalized Benders method consists of performing one Benders iteration on the MILP master problem (MIPDF). It should also be noted that slacks can be introduced to (MDP1) and to (MIPDF) to reduce the effect of nonconvexities as in the augmented-penalty MILP master problem [10].

Finally, it should be noted that S. Lee and Grossmann [6] have developed a new branch and bound method and a MINLP reformulation that is based on the convex hull of each of the disjunctions in (DP1) with nonlinear inequalities.

See also

- ▶ Chemical Process Planning
- ▶ Decomposition Principle of Linear Programming
- ▶ Disjunctive Programming
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming

- ▶ Reformulation-linearization Methods for Global Optimization
- ▶ Simplicial Decomposition
- ▶ Simplicial Decomposition Algorithms
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Successive Quadratic Programming: Decomposition Methods

References

1. Beaumont N (1991) An algorithm for disjunctive programs. Europ J Oper Res 48:362–371
2. Benders JF (1982) Partitioning procedures for solving mixed-variables programming problems. Numer Math 4:238–252
3. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307
4. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10(4):237–260
5. Kocis GR, Grossmann IE (1989) A modeling and decomposition strategy for the MINLP optimization of process flowsheets. Comput Chem Eng 13:797
6. Lee S, Grossmann IE (2000) New algorithms for nonlinear generalized disjunctive programming. Comput Chem Eng 24:2125
7. Raman R, Grossmann IE (1993) Symbolic integration of logic in mixed integer linear programming techniques for process synthesis. Comput Chem Eng 17:909
8. Raman R, Grossmann IE (1994) Modelling and computational techniques for logic based integer programming. Comput Chem Eng 18:563
9. Turkay M, Grossmann IE (1996) A logic based outer-approximation algorithm for MINLP optimization of process flowsheets. Comput Chem Eng 20:959–978
10. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer-approximation method for MINLP optimization. Comput Chem Eng 14:769

MINLP: Mass and Heat Exchanger Networks

MEN, MHEN

Katerina P. Papalexandri
bp Upstream Technol., Sunbury-on-Thames,
Middlesex, UK

MSC2000: 93A30, 93B50

Article Outline

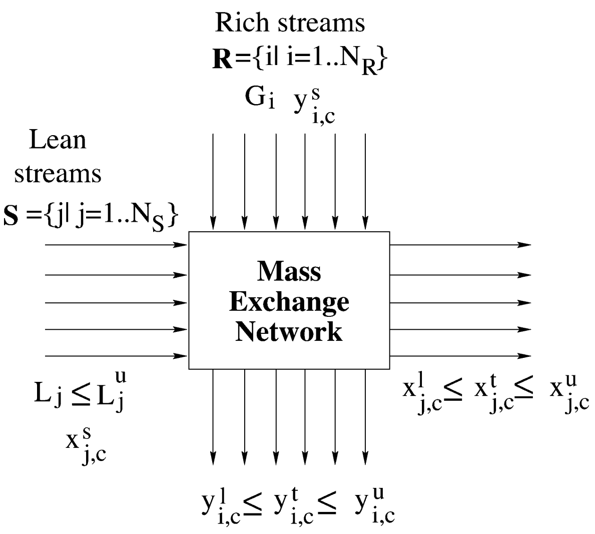
Keywords

MINLP; Mass and heat exchange; Separation

Mass integration in the form of *mass exchanger networks*, MEN, appears in the chemical industries as an economic alternative in waste treatment, feed preparation, product separation, recovery of valuable materials, etc. The MEN involves a set of rich streams, wherefrom one or more components are removed by means of lean streams (mass separating agents) in mass transfer operations that do not require energy (constant pressure and temperature).

The MEN synthesis/design problem is posed as a combinatorial problem, involving discrete and continuous decisions (e. g. the mass exchange operations/matches and the unit sizes, respectively), that both affect the overall mass integration cost.

When the mass transfer operations can take place at different temperatures, heat integration of the rich and


Rich streams
$\mathbf{R}=\{i| i=1..N_R\}$
$G_i \quad y^s_{i,c}$
Lean streams
$\mathbf{S}=\{j| j=1..N_S\}$
Mass Exchange Network
$L_j \leq L^u_j$
$x^s_{j,c}$
$x^l_{j,c} \leq x^t_{j,c} \leq x^u_{j,c}$
$y^l_{i,c} \leq y^t_{i,c} \leq y^u_{i,c}$

MINLP: Mass and Heat Exchanger Networks, Figure 1

lean streams is also considered within a combined *mass and heat exchanger network*, MHEN, synthesis problem.

In isothermal MEN synthesis, the simultaneous optimization of the mass exchange operations, the mass separating agent flows and the network configuration has been formulated by K.P. Papalexandri, E.N. Pistikopoulos and C.A. Floudas [9] as an MINLP problem based on:

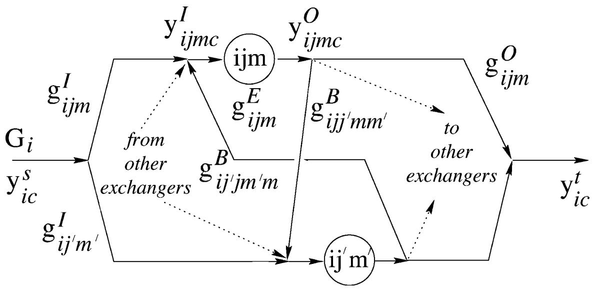
a) the MEN superstructure of synthesis/design alternatives;
b) modeling of mass exchange in each mass exchanger; and,
c) minimization of a total annualized network cost.

Details are given below

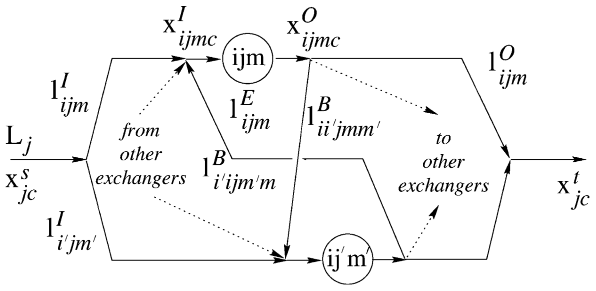
MEN Superstructure

The *MEN superstructure* for a given set of rich and lean streams includes all possible mass exchange operations (*mass exchange matches*) between the network streams in all possible network configurations. Its main features are:

- Each potential match between a rich and a lean stream corresponds to a potential mass exchanger (one-to-one correspondence).
  Multiple mass exchange matches between two streams may be considered (i. e. streams integrated at different points in the network), increasing thus the considered MEN structures and the combinatorial complexity of the synthesis problem. Note that, this is not similar to an a priori decomposition of the network into separable subnetworks.
- Each stream entering the network is split towards all its potential mass exchanger units.
- After each mass exchanger, a splitter is considered for each stream, where the stream is split towards its final mixer and all the other potential stream exchangers.
- Prior to each potential mass exchanger, a mixer is considered for each participating stream, where the flow from the initial splitter and connecting (bypass) flows from all the other exchangers of the stream are merged into the flow towards the exchanger.
- A mixer is considered at the network outlet of each stream, where flows from all the potential stream exchangers are merged into the outlet flow.



MINLP: Mass and Heat Exchanger Networks, Figure 2
Rich stream superstructure



MINLP: Mass and Heat Exchanger Networks, Figure 3
Lean stream superstructure

For example, for a rich stream i and its mth and m'th possible exchangers with lean streams j and j' respectively, we have Fig. 2

In Fig. 2, $c = 1, \ldots, C$ are the transferable components. All possible configurations for the two exchangers ((ijm) and $(ij'm')$ in series, or in parallel), result by 'deleting' appropriate connecting streams. Stream deletion corresponds to zero stream flows (e. g. $g^I_{ij'm'} = g^O_{ijm} = 0$ and $g^B_{ij'jm'm} = 0$ results in the exchangers in series).

For a lean stream j and its mth and m'th exchangers with rich streams i and i', we have Fig. 3.

The MEN superstructure is described by mass balances for the overall streams and each transferable component at the exchangers, splitters and mixers of the superstructure:

$$\begin{cases} g^E_{ijm}(y^I_{ijmc} - y^O_{ijmc}) = M_{ijmc}, \\ \quad i \in R,\ c = 1, \ldots, C, \\ l^E_{ijm}(x^O_{ijmc} - x^I_{ijmc}) = M_{ijmc}, \\ \quad i \in S,\ c = 1, \ldots, C, \end{cases} \tag{1}$$

$$\begin{cases} \sum_{j\in S,m} g^I_{ijm} - G_i = 0, & i \in R, \\ \sum_{i\in R,m} l^I_{ijm} - L_j = 0, & i \in S, \end{cases} \tag{2}$$

$$\begin{cases} g^O_{ijm} + \sum_{j'\in S,m'} g^B_{ijj'mm'} - g^E_{ijm} = 0, \\ l^O_{ijm} + \sum_{i'\in S,m'} l^B_{ii'jmm'} - l^E_{ijm} = 0, \\ \quad i \in R,\ j \in S,\ m = 1,\dots,M, \end{cases} \tag{3}$$

$$\begin{cases} g^I_{ijm} + \sum_{j'\in S,m'} g^B_{ij'jm'm} - g^E_{ijm} = 0, \\ l^I_{ijm} + \sum_{i'\in S,m'} l^B_{i'ijm'm} - l^E_{ijm} = 0, \\ \quad i \in R,\ j \in S,\ m = 1,\dots,M, \end{cases} \tag{4}$$

$$\begin{cases} g^I_{ijm} y^s_{ic} + \sum_{j'\in S,m'} g^B_{ij'jm'm} y^O_{ij'm'c} \\ \quad - g^E_{ijm} y^I_{ijmc} = 0, \\ l^I_{ijm} x^s_{jc} + \sum_{i'\in S,m'} l^B_{i'ijm'm} x^O_{i'jm'c} \\ \quad - l^E_{ijm} x^I_{ijmc} = 0, \\ \quad i \in R,\ j \in S, \\ \ c = 1,\dots,C,\ m = 1,\dots,M, \end{cases} \tag{5}$$

$$\begin{cases} \sum_{j\in S,m} g^O_{ijm} y^O_{ijmc} - G_i y^t_{ic} = 0, \\ \quad i \in R,\ c = 1,\dots,C, \\ \sum_{i\in R,m} l^O_{ijm} x^O_{ijmc} - L_j x^t_{jc} = 0, \\ \quad i \in S,\ c = 1,\dots,C, \end{cases} \tag{6}$$

where the inlet, outlet, exchanger and exchanger-connecting flows of the rich and lean streams (g^I, g^O, g^E, g^B and l^I, l^O, l^E, l^B, respectively) and the intermediate compositions of components (molar fractions x^I, x^O, y^I, y^O) are illustrated in the corresponding superstructure figures.

Modeling Mass Exchange

The existence of each potential mass exchanger in the network is denoted by a binary variable:

$$E_{ijm} = \begin{cases} 1, & \text{when the } m\text{th exchanger} \\ & \text{between streams } i \text{ and } j \text{ exists,} \\ 0, & \text{otherwise,} \end{cases}$$

and defined by

$$\begin{cases} g^E_{ijm} - E_{ijm} U \le 0, \\ l^E_{ijm} - E_{ijm} U \le 0, \\ M_{ijmc} - E_{ijm} U \le 0, \\ g^E_{ijm},\ l^E_{ijm},\ M_{ijmc} \ge 0, \end{cases} \tag{7}$$

where M_{ijmc} is the mass exchange load of component c in mass exchanger (ijm), and U a large positive number.

In each potential mass exchanger a component c is transferred from the rich to the lean stream when the rich composition is greater than the equilibrium composition with respect to the lean stream:

$$y_c \ge f(x_c),$$

where $f(x_c)$ is the mass transfer equilibrium relation, that may account for reactive mass transfer also.

Feasibility of mass transfer is ensured imposing the above constraint at the inlet and outlet of the streams, i. e. (for counter-current flows):

$$\begin{cases} -y^I_{ijmc} + f(x^O_{ijmc}) + \epsilon_{ijc} - (1 - E_{ijm})U \le 0, \\ -y^O_{ijmc} + f(x^I_{ijmc}) + \epsilon_{ijc} - (1 - E_{ijm})U \le 0, \end{cases} \tag{8}$$

where ϵ_{ijc} is a *minimum composition difference* that is required for feasible mass exchange in a unit of finite size (e. g. imposed from mechanical constraints). When $f(x_c)$ is not convex the constraints in (8) cannot guarantee feasible mass transfer throughout the exchanger. In this case $f(x_c)$ can be approximated by a set of convex functions and feasible mass transfer be ensured considering the constraints in (8) also for intermediate exchanger points, that define the convex parts. Note that, the mass-transfer feasibility or driving-force constraints in (8) are activated only when the corresponding exchanger exists ($E_{ijm} = 1$).

The size of each potential mass exchanger (number of mass transfer stages, N^{st}, etc.) is calculated as a function of the variable mass transfer, through appropriate design equations (e. g. for perforated-plate columns the *Kremser equation*):

$$N^{st}_{ijm} = N^{st}(g^E_{ijm}, l^E_{ijm}, x^I_{ijmc}, x^O_{ijmc}, y^I_{ijmc}, y^O_{ijmc}). \tag{9}$$

Minimizing Network Cost

The total network cost comprises

- the annualized capital cost of the mass exchangers, that may be discontinuous (involve a fixed charge cost factor), and
- the annualized operating cost, i. e. the cost of the mass separating agents.

Consequently, the MEN MINLP synthesis model is formulated as follows:

$$(\mathrm{P1})\quad \min$$

$$\sum_{ijm}\left(AC^{1}_{ijm}E_{ijm}+AC^{2}_{ijm}(N^{st}_{ijm})\right)+\sum_{j}AC^{3}_{j}L_{j}$$

such that

$$\begin{aligned}
&(2)-(9)\\
&g^{I}_{ijm}, g^{E}_{ijm}, g^{B}_{ijj'mm'}, g^{O}_{ijm}\geq 0,\\
&y^{I}_{ijmc}, y^{O}_{ijmc}\geq 0,\\
&i\in R,\ j,j'\in S,\\
&m,m'=1,\dots,M,\\
&c=1,\dots,C,\\
&l^{I}_{ijm}, l^{E}_{ijm}, l^{B}_{ii'jmm'}, l^{O}_{ijm}\geq 0,\\
&x^{I}_{ijmc}, x^{O}_{ijmc}\geq 0,\\
&i\in R,\ j,j'\in S,\\
&m,m'=1,\dots,M,\\
&c=1,\dots,C,\\
&E_{ijm}=0,1,\\
&i\in R,\ j,j'\in S,\\
&m,m'=1,\dots,M.
\end{aligned}$$

(P1) is a nonconvex MINLP problem and global optimization methods are required to guarantee global optimal solutions.

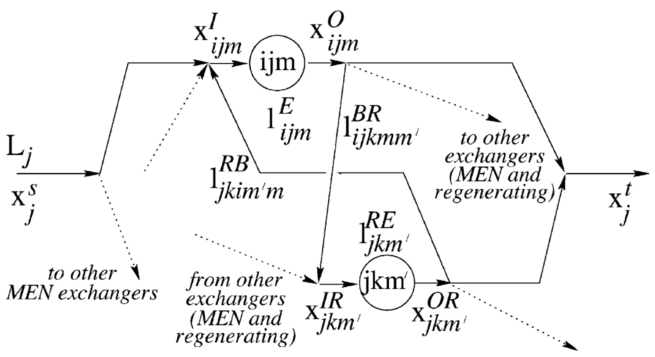
The main advantage of the simultaneous MEN synthesis model (P1), as opposed to the sequential MEN synthesis method, is that the trade-off between the capital and operating costs is systematically considered. Also,

- (P1) derives the optimal network with respect to all the transferable components, considering the mass transfer of each component separately within the calculated mass-transfer stages of each exchanger.
- Forbidden mass exchange matches, limited mass exchange and/or forbidden exchanger connections can be explicitly considered in (P1).
- Variable target compositions are straightforwardly handled.

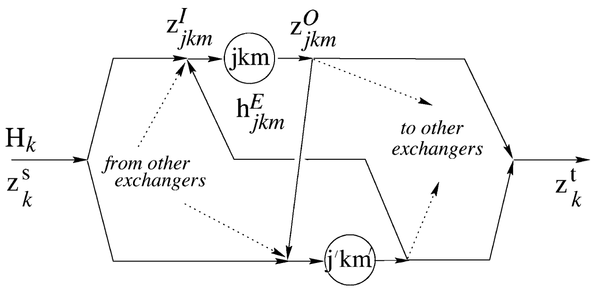
When the mass exchange matches and mass exchange loads are fixed (e. g. when these are determined within a sequential MEN synthesis framework), (P1) reduces to an NLP and can be solved to derive a network configuration and unit sizes with minimum capital cost.

Extending the concept of cost optimality of the mass exchanger network, two special cases have been studied:

- MEN and *regeneration networks*.
  When regenerating agents are available for some (or all) lean streams, the total mass integration cost involves also the regeneration cost. The regeneration network can be considered simultaneously within the MINLP MEN synthesis model [9], accounting for all the regeneration alternatives of the lean streams and employing binary variables to denote the existence of the regenerating exchangers. In this case, the mass separating agents behave as lean streams in the mass exchangers of the main MEN and as rich streams in the regenerating mass exchangers. The regeneration network is not necessarily separable from the main MEN, as a lean stream may be partly regenerated before being used as a separating agent in another mass exchanger. Thus, the lean stream superstructures involve all the possible interconnections between the exchangers of the main MEN and the regenerating exchangers. For example, for a lean stream j and its mth and m'th exchangers with rich stream i and regenerant k we have Fig. 4.
  The overall superstructure of mass exchange and regeneration alternatives involves also the superstructures of the regenerating agents, that have variable flows, while the overall network cost includes the main MEN and the regeneration cost (capital and operating cost).
- *Flexible mass exchange networks*.
  The ability of MEN to accommodate variations in the rich stream flows and inlet compositions in an efficient manner affects cost optimality. A *multiperiod MINLP MEN synthesis model* has been suggested in [7], to derive mass exchange networks, flexible to accommodate in an optimal manner different mass integration requirements. In the multiperiod MINLP model a weighted operating cost is optimized simultaneously with the capital cost



MINLP: Mass and Heat Exchanger Networks, Figure 4
Regenerable lean stream superstructure



MINLP: Mass and Heat Exchanger Networks, Figure 5
Regenerating stream superstructure

for mass exchangers that can operate feasibly under the different conditions. The MEN superstructure is extended to include control variables that enhance flexibility (as exchanger-bypassing streams and overall bypass streams that are accordingly penalized).

When the alternative mass transfer operations take place at different and/or variable temperatures, heat integration between the network streams can be simultaneously considered within a combined MEN and HEN synthesis problem [7]. The available rich and lean streams define hot, cold or hot-and-cold streams in the heat integration problem, depending on whether their supply and target compositions are above or below the mass exchange temperatures. Thus, their heat exchange alternatives include both hot- and cold-side matching. Inlet and outlet temperatures and compositions in mass and heat exchangers are variables. The combined mass and heat exchanger superstructure involves all the possible mass and heat exchangers of a stream and all the possible interconnections between them, Fig. 6.

The combined MEN and HEN superstructure is described by

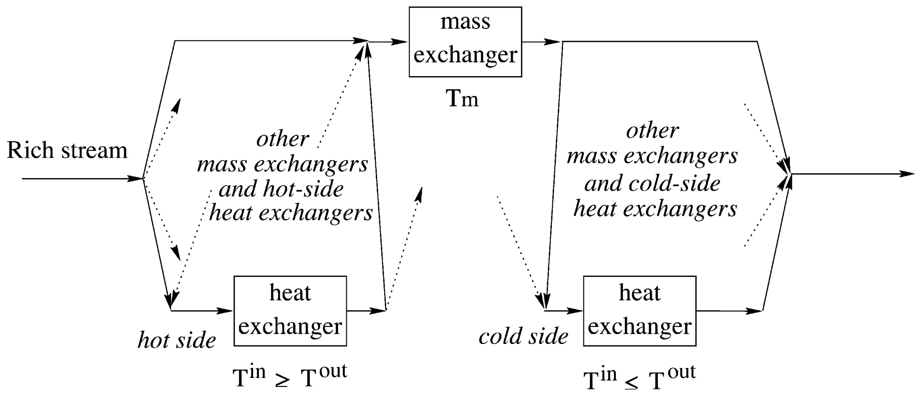
- mass balances at the superstructure splitters (i. e. the initial stream splitters and the splitters after each side of the possible mass and heat exchangers), similar to (2) and (3), and considering all the connecting flows;
- mass balances for overall flows and transferable components at the superstructure mixers (i. e. the final stream mixers and the mixers prior to each side of the potential mass and heat exchangers), similar to (4), (5) and (6), and considering all the connecting flows;
- energy balances at the superstructure mixers;
- mass balances at the mass exchangers, similar to (1), and
- energy balances at the heat exchangers.

The MHEN synthesis model also involves

- binary variables, to denote the existence of mass and heat exchangers, and their definition (mixed integer constraints),
- driving force constraints for mass exchange (8) at the potential mass exchangers, and for heat exchange at the potential heat exchangers (based on $\Delta T_{\min}$),
- design equations for the potential mass and heat exchangers, and
- a total annualized network cost.

and is formulated as a (nonconvex) MINLP.

The simultaneous MHEN synthesis model addresses systematically the trade-off between capital and operating cost of mass and heat integration. The MEN and HEN are not assumed separable. Thus, better integration can be achieved, as it is allowed for a stream to be partly heated for a particular mass exchange operation and then heated further for final purification.



MINLP: Mass and Heat Exchanger Networks, Figure 6
Combined MEN and HEN superstructure

In the simple case when the temperatures of the mass exchange operations are given or can be prepostulated, the rich and lean streams define hot (or cold) streams before participating to mass exchangers and cold (or hot) streams afterwards [11].

The final mass and heat exchanger network structure results from the flows of the superstructure substreams. Alternatively, the use of binary variables has been suggested in [7] to denote the existence of exchanger connections. This, although increasing the combinatorial complexity of the MINLP synthesis model, allows for:

i) explicit piping cost considerations,
ii) structural constraints to be easily modeled, and
iii) the solution of simple NLP subproblems within a decomposition-based MINLP solution method.

Mass exchange networks have been introduced as an end-of-pipe treatment alternative. However, the extent of mass recovery and the corresponding cost are closely related to the reactive and mixing operations in a process. A. Lakshmanan and L.T. Biegler [6] have suggested a MINLP model for the synthesis of optimal reactor networks, where the thermodynamic feasibility of mass integration and its implications are taken simultaneously into account, applying the first and second thermodynamic laws for mass exchange, i. e.

- Total mass balance for the mass integrated streams (resulting process and available rich and lean streams);

$$\sum_{i \in \mathbf{R}} G_i (y^s_{ic} - y^t_{ic}) = \sum_{j \in \mathbf{S}} L_j (x^t_{jc} - x^s_{jc}) \qquad (10)$$

  and

- feasibility of mass exchange above (and below) each candidate mass exchange pinch:

$$\left\{ \begin{array}{l} \text{Mass lost by all the rich} \\ \text{streams below each pinch} \\ \text{point candidate} \end{array} \right\} - \left\{ \begin{array}{l} \text{Mass gained by all the lean} \\ \text{streams below each pinch} \\ \text{point candidate} \end{array} \right\} \leq 0$$

  i. e.

$$\begin{aligned} &\sum_{i \in \mathbf{R}} G_i \\ &\times \left[\max(0, y_p - y^t_{ic}) - \max(0, y_p - y^S_{ic}) \right] \\ &- \sum_{j \in \mathbf{S}} L_j \\ &\times \left[\max(0, x_p - x^S_{jc}) - \max(0, x_p - x^t_{jc}) \right] \\ &\leq 0 \end{aligned} \qquad (11)$$

Note that the thermodynamic feasibility requirements in (11) involve nondifferentiable terms if inlet and outlet compositions are variables (position of streams with respect to candidate pinch points). These can be handled either employing differentiable approximation functions [6], or introducing binary variables [2,3,5].

The main assumption in MEN is that mass transfer operations are isothermal. In the general case these can be followed (or caused) by heat transfer, as in distillation. Assuming constant counter-current molar flows, M.J. Bagajewicz and V. Manousiouthakis showed in [1] that distillation columns can be handled as pure mass transfer operations and derived targets for energy consumption and separation of a 'key' component, employing the first and second thermodynamic laws in (10) and (11), within an MINLP-based MHEN sequential synthesis framework. The problem of energy-induced separations has been addressed by M.M. El-Halwagi, B.K. Srinivas and R.F. Dunn in [4], translating the energy-based separation tasks into simple energy-requiring operations (heating and cooling tasks) and deriving targets for energy consumption and the corresponding mass recovery, based on thermodynamic feasibility constraints.

Extending the concept of mass exchange to non-isothermal mass transfer operations Papalexandri and Pistikopoulos introduced a *mass/heat transfer module* [8], where mass is transferred between different phases or reacting species if that is thermodynamically feasible, i. e. if that decreases the total Gibbs free energy of the system. Mass and energy balances, taking into account possible reactions, and mass-transfer driving-force constraints based on total Gibbs free energy are employed to model the mass/heat transfer module as an aggregate of differential mass and energy transfer phenomena. Considering a superstructure of mass/heat and heat exchange modules in a process and all possible interconnections between them, process synthesis tasks can be formulated as mass/heat and heat exchange superstructure MINLP problems, where binary variables are employed to denote the existence of mass/heat and heat exchangers. Then, process operations (conventional and/or hybrid) and networks are derived as combinations of mass/heat and heat exchange phenomena [8,10].

See also

- ► Global Optimization of Heat Exchanger Networks
- ► MINLP: Global Optimization with αBB
- ► MINLP: Heat Exchanger Network Synthesis
- ► Mixed Integer Linear Programming: Heat Exchanger Network Synthesis
- ► Mixed Integer Linear Programming: Mass and Heat Exchanger Networks

References

1. Bagajewicz MJ, Manousiouthakis V (1992) Mass/heat exchange network representation of distillation networks. AIChE J 38:1769–1800
2. El-Halwagi MM, Manousiouthakis V (1990) Simultaneous synthesis of mass-exchange and regeneration networks. AIChE J 36:1209–1219
3. El-Halwagi MM, Srinivas BK (1992) Synthesis of reactive mass exchange networks. Chem Eng Sci 47:2113–2119
4. El-Halwagi MM, Srinivas BK, Dunn RF (1995) Synthesis of optimal heat-induced separation networks. Chem Eng Sci 50:81–97
5. Gupta A, Manousiouthakis V (1993) Minimum utility cost of mass exchange networks with variable single component supplies and targets. Industr Eng Chem Res 32:1937–1950
6. Lakshmanan A, Biegler LT (1996) Synthesis of optimal chemical reactor networks with simultaneous mass integration. Industr Eng Chem Res 35:4523–4536
7. Papalexandri KP, Pistikopoulos EN (1994) A multiperiod MINLP model for the synthesis of flexible heat and mass exchange networks. Comput Chem Eng 18:1125–1139
8. Papalexandri KP, Pistikopoulos EN (1996) A generalized modular representation framework for process synthesis. AIChE J 42:1010–1032
9. Papalexandri KP, Pistikopoulos EN, Floudas CA (1994) Mass exchange networks for waste minimization: A simultaneous approach. Chem Eng Res Des 72:279–294
10. Sebastian P, Nadeau JP, Puiggali JR (1996) Designing dryers using heat and mass exchange networks: An application to conveyor belt dryers. Chem Eng Res Des 74:934–943
11. Srinivas BK, El-Halwagi MM (1994) Synthesis of combined heat and reactive mass exchange networks. Chem Eng Sci 49:2059–2074

MINLP: Outer Approximation Algorithm

IGNACIO E. GROSSMANN
Carnegie Mellon University, Pittsburgh, USA

MSC2000: 90C11

Article Outline

Keywords

Mixed integer nonlinear programming; Outer approximation method; Generalized Benders decomposition; Extended cutting plane method

The *outer approximation algorithm* (OA algorithm) ([1,2,9]) addresses mixed integer nonlinear programs of the form:

$$(\mathrm{P})\quad \begin{cases} \min & Z = f(x,y) \\ \text{s.t.} & g_j(x,y) \le 0\,, \quad j \in J\,, \\ & x \in X\,, y \in Y\,, \end{cases}$$

where $f(\cdot)$, $g(\cdot)$ are convex, differentiable functions, J is the index set of inequalities, and x and y are the continuous and discrete variables, respectively. The set X is commonly assumed to be a convex compact set, e. g. $X = \{x: x \in \mathbf{R}^n, Dx \le d, x^L \le x \le x^U\}$; the discrete set Y corresponds to a polyhedral set of integer points, $Y = \{y: y \in \mathbf{Z}^m, Ay \le a\}$, and in most cases is restricted to $0-1$ values, $y \in \{0, 1\}^m$. In most applications of interest the objective and constraint functions $f(\cdot)$, $g(\cdot)$ are linear in y (e. g. fixed cost charges and logic constraints).

The OA algorithm is based on the following theorem [1]:

Theorem 1 *Problem (P) and the following mixed-integer linear program (MILP) master problem (M-OA) have the same optimal solution (x^*, y^*),*

$$(\mathrm{M-OA})\quad \min Z_L = \alpha$$

such that

$$\begin{aligned} &\alpha \ge f(x^k, y^k) + \nabla f(x^k, y^k)\begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix}\,, \\ &g_j(x^k, y^k) + \nabla g_j(x^k, y^k)\begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \le 0\,, \\ &j \in J\,, \quad k \in K^*\,, \\ &x \in X\,, \quad y \in Y\,, \end{aligned}$$

where

$$K^* = \left\{ k\colon \begin{array}{c} (x^k, y^k) \textit{ is the optimal} \\ \textit{solution to (NLP1)} \\ \textit{for all feasible } y^k \in Y \end{array} \right\}\,,$$

$$(\mathrm{NLP1})\quad \begin{cases} \min & Z_U^k = f(x, y^k) \\ \text{s.t.} & g_j(x, y^k) \le 0\,, \quad j \in J\,, \\ & x \in X\,, \end{cases}$$

where Z_U^k is an upper bound to the optimum of problem (P).

Note that since the functions $f(x, y)$ and $g(x, y)$ are convex, the linearizations in (M-OA) correspond to outer approximations of the nonlinear feasible region in problem (P). Also, since the master problem (M-OA) requires the solution of all feasible discrete variables y^k, the following MILP relaxation is considered, assuming that the solution of K NLP subproblems is available:

$$(\mathrm{RM-OA})\quad \min Z_L^K = \alpha$$

such that

$$\begin{aligned} &\alpha \ge f(x^k, y^k) + \nabla f(x^k, y^k)\begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix}\,, \\ &g_j(x^k, y^k) + \nabla g_j(x^k, y^k)\begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \le 0\,, \\ &j \in J\,, \quad k = 1, \dots, K\,, \\ &x \in X\,, \quad y \in Y\,. \end{aligned}$$

Given the assumption on convexity of the functions $f(x,y)$ and $g(x,y)$, the following property can be easily be established,

Property 2 The solution of problem (RM-OA), corresponds to a lower bound to the solution of problem (P).

Note that since function linearizations are accumulated as iterations proceed, the master problems (RM-OA) yield a nondecreasing sequence of lower bounds, $Z_L^1 \le \cdots \le Z_L^K$, since linearizations are accumulated as iterations k proceed.

The OA algorithm as proposed by M.A. Duran and I.E. Grossmann [1] consists of performing a cycle of major iterations, $k = 1, \dots, K$, in which (NLP1) is solved for the corresponding y^k, and the relaxed MILP master problem (RM-OA) is updated and solved with the corresponding function linearizations at the point (x^k, y^k). The (NLP1) subproblems yield an upper bound that is used to define the best current solution, $\mathrm{UB}^K = \min(Z_U^k)$. The cycle of iterations is continued until this upper bound and the lower bound of the relaxed master problem, are within a specified tolerance.

It should be noted that for the case when the problem (NLP1) has no feasible solution, there are two major ways to handle this problem. The more general option is to consider the solution of the feasibility problem,

$$\text{(NLFP)}\quad \begin{cases} \min & u \\ \text{s.t.} & g_j(x, y^k) \leq u\,, \quad j \in J\,, \\ & x \in X\,, \quad u \in \mathbb{R}^1\,. \end{cases}$$

R. Fletcher and S. Leyffer [2] have shown that for infeasible NLP subproblems, if the linearization at the solution of problem (NLFP) is included, this will guarantee convergence to the optimal solution.

For the case when the discrete set Y is given by 0–1 values in problem (P), the other option to ensure convergence of the OA algorithm without solving the feasibility subproblems (NLFP), is to introduce the following integer cut whose objective is to make infeasible the choice of the previous 0–1 values generated at the K previous iterations [1]:

$$\text{(ICUT)}\quad \begin{cases} \sum\limits_{i \in B^k} y_i - \sum\limits_{i \in N^k} y_i \leq \left| B^k \right| - 1\,, \\ k = 1, \dots, K\,, \end{cases}$$

where $B^k = \{i{:}y_i^k = 1\}$, $N^k = \{iy_i^k = 0\}$, $k = 1, \dots, K$. This cut becomes very weak as the dimensionality of the 0–1 variables increases. However, it has the useful feature of ensuring that new 0–1 values are generated at each major iteration. In this way the algorithm will not return to a previous integer point when convergence is achieved. Using the above integer cut the termination takes place as soon as $Z_L^K \geq \text{UB}^K$.

The OA method generally requires relatively few cycles or major iterations. One reason for this behavior is given by the following property:

Property 3 The OA algorithm trivially converges in one iteration if $f(x, y)$ and $g(x, y)$ are linear.

The proof simply follows from the fact that if $f(x, y)$ and $g(x, y)$ are linear in x and y the MILP master problem (RM-OA) is identical to the original problem (P).

It is also important to note that the MILP master problem need not be solved to optimality. In fact given the upper bound UB^K and a tolerance ε it is sufficient to generate the new (y^K, x^K) by solving,

$$(\text{M}-\text{OAF})\quad \min Z_L^K = 0\alpha$$

such that

$$\begin{aligned} &\alpha \geq \text{UB}^k - \varepsilon\,, \\ &\alpha \geq f(x^k, y^k) + \nabla f(x^k, y^k) \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix}\,, \\ &g_j(x^k, y^k) + \nabla g_j(x^k, y^k) \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} \leq 0\,, \\ &j \in J\,, \quad k = 1, \dots, K\,, \\ &x \in X\,, \quad y \in Y\,. \end{aligned}$$

While in (M-OA) the interpretation of the new point y^K is that it represents the best integer solution to the approximating master problem, in (M-OAF) it represents an integer solution whose lower bounding objective does not exceed the current upper bound UB^K; in other words it is a feasible solution to (M-OA) with an objective below the current estimate. Note that in this case the OA iterations are terminated when (M-OAF) is infeasible.

Another interesting point about the OA algorithm is the relationship of its master problem with the one of the *generalized Benders decomposition* method [3], which is given by:

$$(\text{RM}-\text{GBD})\quad \min Z_L^K = \alpha$$

such that

$$\begin{aligned} &\alpha \geq f(x^k, y^k) + \nabla_y f(x^k, y^k)^\top (y - y^k) \\ &\quad + (\mu^k)^\top \left[g(x^k, y^k) + \nabla g(x^k, y^k)(y - y^k) \right]\,, \\ &k \in KFS\,, \\ &(\lambda^k)^\top \left[g(x^k, y^k) + \nabla g(x^k, y^k)(y - y^k) \right]\,, \\ &k \in KIS\,, \\ &x \in X\,, \quad \alpha \in \mathbb{R}^1\,, \end{aligned}$$

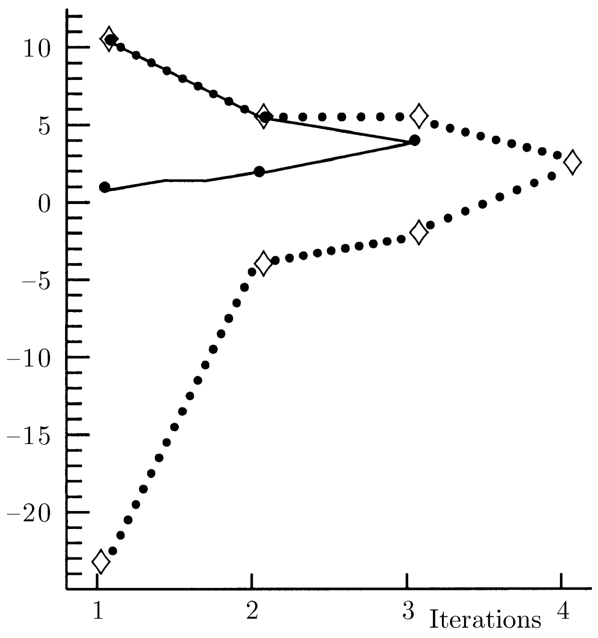
where *KFS* is the set of feasible subproblems (NLP1) and *KIS* the set of infeasible subproblems whose solution is given by (NLFP). Also |KFS ⊂ KIS | = K. The following property, holds between the two methods [1]:

Property 4 Given the same set of K subproblems, the lower bounds predicted by the relaxed master problem (RM-OA) are greater or equal to the ones predicted by the relaxed master problem (RM-GBD).

The above proof follows from the fact that the Lagrangian and feasibility cuts in (RM-GBD) are surrogates of the outer approximations in the master problem (M-OA). Given the fact that the lower bounds of

MINLP: Outer Approximation Algorithm, Table 1
Summary of computational results

Method	Subproblems	Master problems	LPs solved
BB	5 (NLP1)		
OA	3 (NLP2)	3 (M-PID)	19 LPs
GBD	4 (NLP2)	4 (M-GBD)	10 LPs
ECP	–	5 (M-MIP)	18LPs



MINLP: Outer Approximation Algorithm, Figure 1
Progress of iterations of OA and GBD for MINLP in MIP-EX

GBD are generally weaker, this method commonly requires a larger number of cycles or major iterations. As the number of 0–1 variables increases this difference becomes more pronounced. This is to be expected since only one new cut is generated per iteration. Therefore user-supplied constraints must often be added to the master problem to strengthen the bounds. As for the OA algorithm, the trade-off is that while it generally predicts stronger lower bounds than GBD, the computational cost for solving the master problem (M-OA) is greater since the number of constraints added per iteration is equal to the number of nonlinear constraints plus the nonlinear objective.

The OA algorithm is also closely related to the *extended cutting plane* (ECP) method by T. Westerlund and F. Peterssen [8]. The main difference lies that in the ECP method no NLP subproblem is solved, and that linerization simply takes place over the predicted continuous points from the MILP master problem, which in turn will normally only include linearizations of the most violated constraints.

Extension of the OA algorithm [4] include the LP/NLP based branch and bound [6], which avoids the complete solution of the MILP master problem (M-OA) at each major iteration. The method starts by solving an initial NLP subproblem which is linearized as in (M-OA). The basic idea consists then of performing an LP-based branch and bound method for (M-OA) in which NLP subproblems (NLP1) are solved at those nodes in which feasible integer solutions are found. By updating the representation of the master problem in the current open nodes of the tree with the addition of the corresponding linearizations, the need of restarting the tree search is avoided. Another important extension has been the method by Fletcher and Leyffer [2] who included a quadratic approximation based on the Hessian of the Lagrangian to the master problem (M-OAF) in order to capture nonlinearities in the 0–1 variables. Note that in this case the optimal solution of the mixed integer quadratic program (MIQP), Z^K, does not predict valid lower bounds in this case, and hence the constraint $\alpha \leq \text{UBK} - \varepsilon$ is added, with which the search is terminated when no feasible solution can be found in the MIQP master.

Finally, in order to handle equations in problem (P), G.R. Kocis and Grossmann [5] proposed the equality relaxation strategy, in which linearizations of equations are converted into inequalities for the MIP master problem according to the sign of the Lagrange multipliers of the corresponding NLP subproblem. J. Viswanathan and Grossmann [7], further proposed to add slack variables to this MILP master problem, and an augmented penalty function. Since in this generally nonconvex case the bounding properties do not apply, the algorithm was modified so as to start with the NLP relaxation of problem (P). If no integer solution is found, iterations between the MILP and NLP subproblems take place until there is no improvement in the objective function. This idea was precisely implemented in the commercial code DICOPT, which can also be modified to the original OA algorithm, if the user knows that the functions $f(x, y)$ and $g(x, y)$ are convex.

Example 5 In order to illustrate the performance of the OA algorithm, a simple numerical MINLP example is considered.

$$(\mathrm{MIP-EX})\quad \begin{cases} \min & Z = y_1 + 1.5y_2 + 0.5y_3 \\ & \quad + x_1^2 + x_2^2 \\ \text{s.t.} & (x_1 - 2)^2 - x_2 \le 0 \\ & x_1 - 2y_1 \ge 0 \\ & x_1 - x_2 - 4(1 - y_2) \le 0 \\ & x_1 - (1 - y_1) \ge 0 \\ & x_2 - y_2 \ge 0 \\ & x_1 + x_2 \ge 3y_3 \\ & y_1 + y_2 + y_3 \ge 1 \\ & 0 \le x_1 \le 4, \quad 0 \le x_2 \le 4 \\ & y_1, y_2, y_3 = 0, 1\ . \end{cases}$$

The optimum solution to this problem corresponds to $y_1 = 0$, $y_2 = 1$, $y_3 = 0$, $x_1 = 1$, $x_2 = 1$, $Z = 3.5$. Figure 1 shows the progress of the iterations of the OA and GBD algorithm with the starting point $y_1 = y_2 = y_3 = 1$. As can be seen the lower bounds predicted by the OA algorithm are considerably stronger than the ones predicted by GBD. In particular at iteration 1, the lower bound of OA is 1.0 while the one of GBD is −23.5. Nevertheless, since this is a very small problem GBD requires only one more iteration than OA (4 versus 3). It is interesting to note that the NLP relaxation of this problem is 2.53, which is significantly lower than the optimal mixed integer solution. Also, as can be seen in Table 1, an NLP-based branch and bound method requires the solution of 5 NLP subproblems, while the ECP method requires 5 successive MILP problems.

See also

- ▶ Chemical Process Planning
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming

References

1. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307
2. Fletcher R, Leyffer S (1994) Solving mixed integer nonlinear programs by outer approximation. Math Program 66:327
3. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10(4):237–260
4. Grossmann IE, Kravanja Z (1997) Mixed-integer nonlinear programming: A survey of algorithms and applications. In: Biegler LT, Coleman TF, Conn AR, Santosa FN (eds) Large-Scale Optimization with Applications. Part II: Optimal Design and Control. IMA vol Math Appl. Springer, Berlin, pp 73–100
5. Kocis GR, Grossmann IE (1987) Relaxation strategy for the structural optimization of process flow sheets. Industr Eng Chem Res 26(1869)
6. Quesada I, Grossmann IE (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Comput Chem Eng 16:937–947
7. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer-approximation method for MINLP optimization. Comput Chem Eng 14:769
8. Westerlund T, Pettersson F (1995) A cutting plane method for solving convex MINLP problems. Comput Chem Eng 19:S131–S136
9. Yuan X, Zhang S, Piboleau L, Domenech S (1988) Une methode d'optimisation nonlineare en variables pour la conception de procedes. Oper Res 22(331)

MINLP: Reactive Distillation Column Synthesis

AMY CIRIC, ZEYNEP GUMUS
Department Chemical Engineering,
University Cincinnati, Cincinnati, USA

MSC2000: 90C90

Article Outline

Keywords

Reactive distillation; Bilevel programming; MINLP

Reactive distillation (RD) occurs when a reaction takes place in the liquid holdup on the trays, in the reboiler, or in the condenser of a distillation column. Reactive distillation can increase the conversion of equilibrium limited reactions by continuously separating products and reactants, improve the selectivity in some kinetically limited reaction systems, and separate azeotropic and isomeric mixtures by converting one species into another that is easy to remove. It can also create a natural heat integration that uses an exothermic heat of reaction to create vapor boilup in a distillation column, and reduce capital costs by completing several processing steps in a single vessel. Reactive distillation is used commercially to produce methyl tert-butyl ether [13], esters including methyl acetate [1], and nylon 6, 6 [9]. It has also been proposed for hydrolysis reactions [7], ethyl- ene glycol synthesis [11], and cumene production [12]. See [7] for a review of the area.

As a result of increasing interest in the reactive distillation technique, systematic reactive distillation design methods have gained much importance. See [2,3,4,5] for residue curve maps, a powerful tool for visualizing distillation problems, to reactive distillation. In [7] this work was extended by including kinetic effects when the Damkohler number is fixed. In [14] synthesis of reactive distillation with multiple reactions is studied.

Reactive distillation poses a challenging problem for optimization based design techniques. Unlike in conventional distillation, holdup volume is an important design variable in reactive distillation, since the reaction generally takes place in the liquid body on the tray. The constant molar overflow assumption of conventional distillation design is not valid unless the reaction has thermal neutrality and is stoichiometrically balanced. For an optimal solution one should take into account that the feed to the column may be distributed. This, in addition to the holdup volume, liquid and vapor flows, composition and temperature profiles, number of trays and feed location(s) become major variables of an optimization problem which searches for a minimum of a cost function. The constraints of this optimization problem are material and energy balances, vapor-liquid equilibria, mole fraction summations, kinetic and thermodynamic relationships, and logical relationships between the variables. The resulting optimization model is a mixed integer nonlinear programming problem since it involves the optimum number of trays and feed tray locations which are integer variables. The cost function and the material and energy balances cause the nonlinearity of the problem.

There are two approaches to RD design via MINLP methods. One addresses reactive distillation through heat and mass exchanger networks [10], and the other addresses it through distillation column superstructures [6,8].

Problem Statement

The general problem of the reactive distillation column synthesis problem can be stated formally as follows.
Given:

- the chemical species, $i = 1, \ldots, I$, involved in the distillation; desired products, $i \in P$, and their production rates P_I;
- the set of chemical reactions, $j = 1, \ldots, J$;
- rate expressions r_j or an equilibrium constant K_j for each reaction j;
- heat of vaporization and vapor-liquid equilibrium data;
- cost of downstream separations;
- cost c_s and composition x_{is} of all feedstocks, $s = 1 \ldots S$;
- the cost of the column as a function of the number of trays and the internal vapor flow rate, $C(V, N)$;
- the form of the catalyst.

Determine:

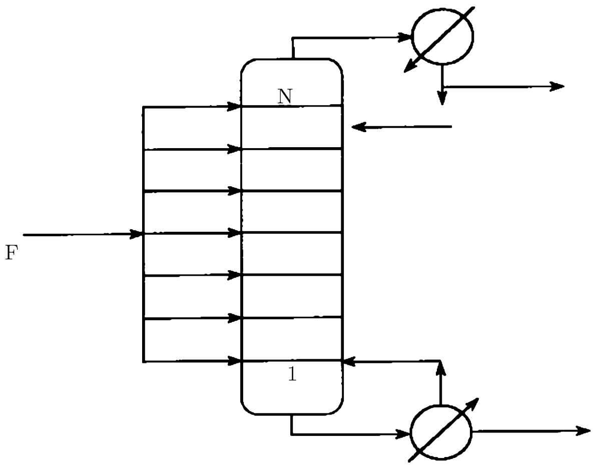
- the optimum number of trays;
- the trays where reactions take place;
- the holdup on each tray where a kinetically limited reaction takes place;

- the reflux ratio;
- the condenser and reboiler duties; and
- the feed location(s).

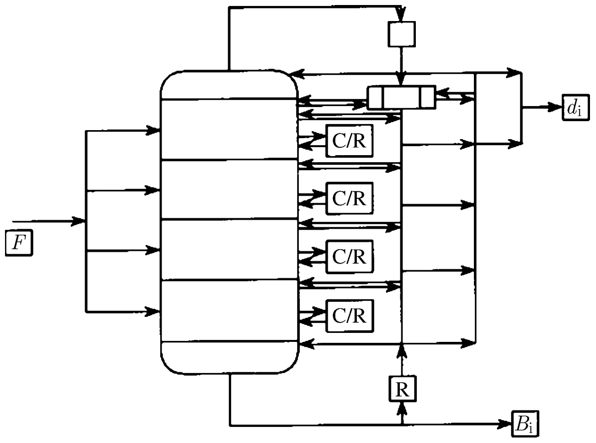
Such that the total cost is minimized while producing the correct amount of product.

Distillation Based Superstructure Approaches

One approach to MINLP based reactive distillation column design uses a superstructure that contains many different alternative designs embedded within it. Two different superstructures have been proposed; they differ in their treatment of the liquid reflux and vapor boilup, and in their heat management. See [6] for a structure that varies the number of trays and always recycles the liquid reflux to the top tray and the vapor boilup to the bottom tray (Fig. 1). More recently (1997), Z.H. Gumus and A.R. Ciric [8] modified the superstructure presented in [15] recycling vapor boilup and liquid reflux to each tray by adding a decanter to the distillate stream and side heaters and coolers to each tray (Fig. 2). In both of these superstructures, the number of trays may vary between 1 and some upper bound K. Each feed stream is split, and a portion is sent to each tray in the superstructure. In kinetically limited reactions, the hold-up volume may vary, and, in reactions systems with a solid catalyst, some trays will have reaction while others do not.



MINLP: Reactive Distillation Column Synthesis, Figure 1
Superstructure for optimum feed location(s) and number of trays [6]



MINLP: Reactive Distillation Column Synthesis, Figure 2
Tray-by-tray superstructure of [8]

The structure shown in Fig. 1 is appropriate for reactive distillation processes with a single liquid phase and kinetically limited reactions that are catalyzed with a solid catalyst. Representing the existence of each tray with an integer variable Y_k leads to a mixed integer nonlinear programming problem whose solution extracts a design with the number of trays, feed tray locations, reactive trays, holdup volumes, reflux ratio and boilup ratio that minimize the total cost. Assumed vapor liquid equilibrium on each tray, no reaction in the vapor phase, homogeneous liquid phase, negligible enthalpy of liquid streams, constant heat of vaporization leads to the MINLP shown below [6]:

$$\begin{aligned}\min Z = c_o + \sum_{sk} c_s F_{sk} + c_R Q_B + c_C Q_C \\ + c_T D^{1.55} \times \sum \left(2Y_k + 1.27\frac{W_k}{D^2}\right) \\ + c_{SH} D \sum_k \left(H_o + \sum_{k' \le k} 2 + 1.27\frac{W_{k'}}{D^2}\right)^{0.802} \\ \times (Y_k - Y_{k+1})\end{aligned}$$

subject to

$$\begin{aligned}\sum_s x_{is} F_{s1} - L_1 x_{i1}(1-\beta) \\ + L_2 x_{i,2} - V_1 K_{i1} x_{i1} + \sum_j \nu_{ij} \xi_{1j} = 0, \quad (1)\end{aligned}$$

$$\left[\sum_s x_{is}F_{sk} + V_{k-1}K_{i,k-1}x_{ik-1} + L_{k+1}x_{i,k+1} - L_k x_{ik} - V_k K_{ik} x_{ik} + \sum_j \nu_{ij}\xi_{ik}\right] Y_k = 0, \quad k = 2, \ldots, K, \tag{2}$$

$$\left[\lambda V_{k-1} - \lambda V_k - \sum_j \Delta H_j \xi_{jk}\right] Y_k = 0, \tag{3}$$

$$D_{ist} = \sum_k (V_k - L_{k+1})(Y_k - Y_{k+1}), \tag{4}$$

$$B_i = (1-\beta)L_1 x_{i1}, \tag{5}$$

$$B_i = P_i, \quad i \in P, \tag{6}$$

$$\left[\sum_i x_{ik} - 1\right](Y_k - Y_{k-1}) = 0, \tag{7}$$

$$\left[\sum_i K_{ik}x_{ik} - 1\right] Y_k = 0, \tag{8}$$

$$xd_i - K_{ik}x_{ik} - 1 + y_k - y_{k+1} \le 0, \tag{9}$$

$$x_{i,k+1} - xd_i - 1 + y_k - y_{k+1} \le 0, \tag{10}$$

$$\sum_i xd_i = 1, \tag{11}$$

$$\xi_{jk} = W_k f_j(x_{ik}, T_k), \tag{12}$$

$$K_{ik} = K_{ik}(x_{ik}, T_k), \tag{13}$$

$$V_k - F_{\max} Y_k \le 0, \tag{14}$$

$$\sum_s F_{sk} - F_{\max} Y_k \le 0, \tag{15}$$

$$L_{k+1} - F_{\max} Y_k \le 0, \tag{16}$$

$$W_k - W_{\max} Y_k \le 0, \tag{17}$$

$$Q_B = \beta\lambda L_1, \tag{18}$$

$$Q_C = \sum_k \lambda V_k (Y_k - Y_{k+1}), \tag{19}$$

$$D^4 \ge C_D \beta^2 L_1^2, \tag{20}$$

$$D \ge D_{\min}, \tag{21}$$

$$Y_{k+1} \le Y_k. \tag{22}$$

In this model, constraints (1) and (2) are the component balances of species i over the bottom tray and the remaining trays k; constraint (3) is the energy balance around tray k. The distillate flow is found with constraint (4). Distillate flow is calculated as the difference between the vapor flow leaving the top tray and the liquid flow entering it. Note that the term $Y_k - Y_{k+1}$ will be nonzero only for top tray, and zero for all others. Constraint (5) calculates the bottoms flow rate and constraint (6) specifies the production rate. Summation equations for the mole fractions are given in constraints (7) and (8). Constraints (9)–(11) identify the top tray and set the distillate and liquid reflux composition equal to the composition of the vapor leaving the top tray. Reaction rates are given in constraint (12), and the vapor liquid equilibrium constant is found by constraint (13). Constraints (14)–(17) ensure that when Y_k equals zero and tray k does not exist, the flows onto and off of the tray are zero. Constraints (18) and (19) calculate the reboiler and condenser duties, while constraints (20) and (21) find the column diameter. The last constraint ensures that tray $k+1$ does not exist if tray k does not exist.

In [6] this technique is demonstrated with the synthesis of a reactive distillation column that makes ethylene glycol from ethylene oxide and water. The main reaction is

$$C_2H_4O + H_2O \rightarrow C_2H_6O_2.$$

Further reaction of ethylene glycol gives the undesired byproduct diethylene glycol:

$$C_2H_4O + C_2H_6O_2 \rightarrow C_4H_{10}O_3$$

Ethylene glycol is produced using reactive distillation because the large volatility difference between the product and the reactants allows the continuous removal of EG from the reaction zone and absorption of the heat of reaction by the separation results in cost cuts.

The problem is solved using the reaction, physical property and cost data given in Table 1. The production rate is taken as 25 kg.mol/h of ethylene glycol. When the problem is solved without specifying the number of feed trays or their locations the solution obtained using GAMS is a 10-tray distillation column with a total

MINLP: Reactive Distillation Column Synthesis, Table 1
Ethylene glycol system; reaction, physical property and cost data

Reaction	Rate (mol/cm^3.s)	ΔH (kJ/mol)
1	$3.15\times10^9 \exp\left[\frac{-9.547}{T(K)}\right] x_{EO}x_{H_2O}$	-80
2	$6.3\times10^9 \exp\left[\frac{-9.547}{T(K)}\right] x_{EO}x_{EG}$	-13.1
Component	K for $P = 1$ atm	
EO	$71.9 \exp\left\{5.72\left[\frac{T-469}{T-35.9}\right]\right\}$	
H_2O	$221.2 \exp\left\{6.31\left[\frac{T-647}{T-52.9}\right]\right\}$	
EG	$77 \exp\left\{9.94\left[\frac{T-645}{T-71.4}\right]\right\}$	
DEG	$47 \exp\left\{10.42\left[\frac{T-681}{T-80.6}\right]\right\}$	
EO feedstock: \$43.7/kmol		
Water feedstock: \$21.9/kmol		
Downstream separation: \$0.15/kmol H_2O in effluent		
C_{sh} =	\$222/yr	
C_T =	\$15.7/yr	
C_R =	\$146.8/kW.yr	
C_C =	\$24.5/kW.yr	
C_O =	\$10,000/yr	

annualized cost of 15.69×10^6/yr. The reaction zone is above tray 4 and the feed is distributed to each tray in the reaction zone. When the problem is slightly modified by adding constraints on the feed tray number, the solution changes to a 10-tray column with a total annualized cost of 15.73×10^6/yr. The reaction zone is between trays 4 and 10 and water is fed to tray 10 while ethylene glycol enters the column at tray 4. The selectivity reached by both columns is the same. (Fig. 3) shows the solutions. The column specifications are given in Table 2.

Heat and Mass Exchange Networks

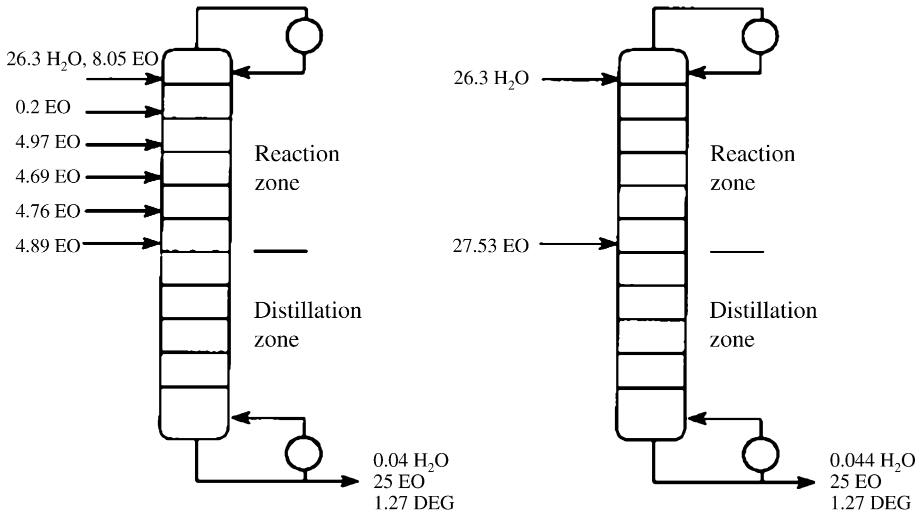
In this approach, process units are defined as combinations of heat and mass exchanger blocks, and the alternatives for the synthesis are explored simultaneously in a superstructure. A reactive distillation column can be described as a combination of mass/heat exchanger units with a condenser and a reboiler [10]. Heat and mass transfer takes place between the contacting vapor and liquid phases and from reactants to products. Multiple feeds and products and side heating and cooling tasks can be included in the description in the form of multiple mass and heat exchanger blocks between liquid and vapor streams. Its phase and quality define each stream. The quality indicator describes the leanness or richness of a stream in different components. Heat and mass transfer occurs between vapor and liquid streams of the same quality or between liquid and liquid (reactant and product) streams. For example, consider the reaction

$$A + B \rightarrow C + D.$$

Then there are liquid and vapor streams LABCD and VABCD in general notation. The streams lean in a component, for example in A, have that letter in parentheses, e. g. L(A)BCD or V(A)BCD. All possibilities of such streams, i. e. LAB(CD), VAB(CD), L(ABC)D, V(ABC)D, L(AB)C(D), V(AB)C(D), etc., and all the possible matches between them are considered within the structure. The possible matches are liquid-vapor matches of the same stream and all liquid-liquid matches.

This model describes exchangers with simple mass and energy balances and constraints defining phase and feasibility. Mass and heat generated or consumed by chemical reactions are included in the balances. Mass transfer is driven by a minimum concentration approach while a minimum temperature approach is the driving force for heat transfer. Concentration and temperature approach constraints are considered at each end of the exchanger. Equilibrium can be represented by a zero concentration approach, which means no driving force for mass transfer.

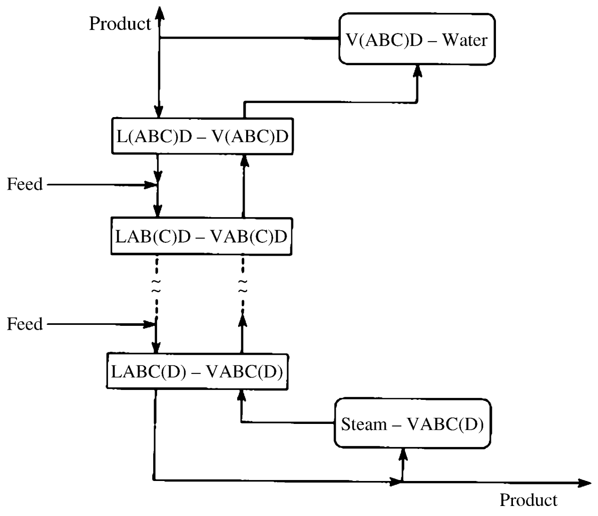
In the synthesis framework for an optimal process network, one should start with the construction of the stream sets containing all the initial, intermediate, and final process streams. The key is the availability of the physical and chemical property information on the streams. When the information is not enough to identify the individual streams, especially the intermediate streams, a general set of one vapor and one liquid stream is constructed, which contain all components involved in the process. The second step is to list all the possible stream matches. Engineering knowledge plays an important role in this step. One should be careful about not listing redundant or meaningless stream



MINLP: Reactive Distillation Column Synthesis, Figure 3
Optimal distributed and two-feed columns for ethylene glycol production

MINLP: Reactive Distillation Column Synthesis, Table 2
Column specifications for ethylene glycol production

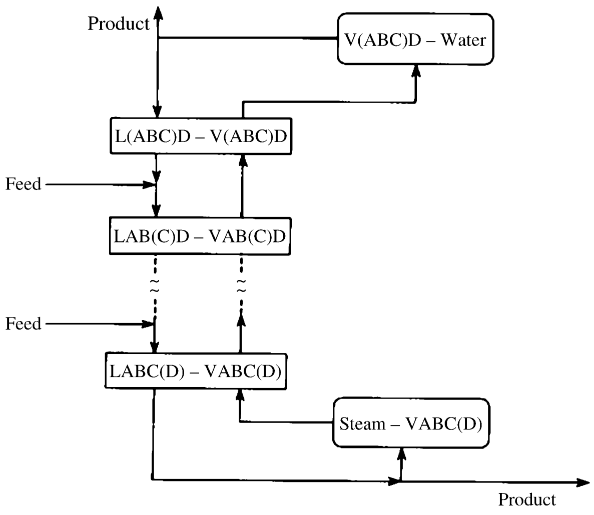
Feed type	Diam. (m)	Height (m)	Boilup ratio	Reboiler duty (MW)	Condenser duty (MW)
Distr.	1.3	12	0.958	6.7	7.31
Two-feed	1.3	12	0.96	6.9	7.5



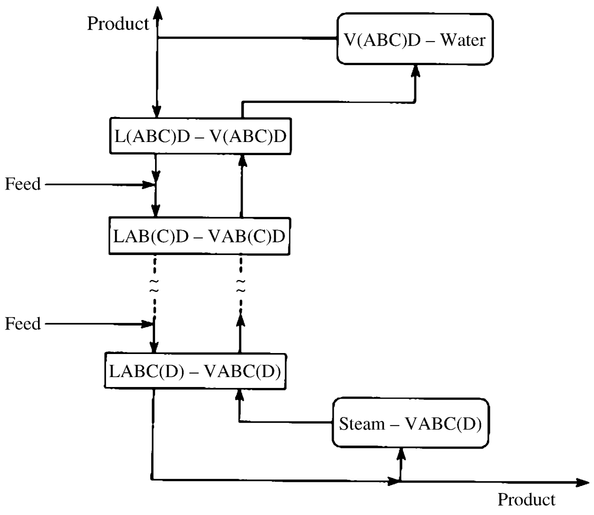
MINLP: Reactive Distillation Column Synthesis, Figure 4
Mass/heat exchange network representation of a multifeed reactive distillation column

matches since these will only make the problem more complex. Knowledge about the system is the key in this screening stage. Developing the mass/heat exchange network superstructure is the next step in the framework. All possible interconnections between the stream splitters and mixers should be taken into consideration. The last step is the optimization of the superstructure. Usually, the objective function of the optimization problem is a cost function. If the cost function includes only operating cost, which depends on the raw material and utility consumption, the objective function can be easily formulated from the superstructure. If, however, capital investment costs are involved in the objective cost function, the formulation is not straightforward from the superstructure, since process unit specifications are not considered in the superstructure. In this case, capital cost is to be approximated using cost functions that take operating conditions into account. Separation difficulty can be used in evaluating the capital cost of a distillation tray.

K.P. Papalexandri and E.N. Pistikopoulos [10] used the production of ethylene glycol from ethylene oxide and water to demonstrate this approach. The reactions involved in this production were given before. Physical properties, cost and reaction data are



MINLP: Reactive Distillation Column Synthesis, Figure 5
Steps of the synthesis framework



MINLP: Reactive Distillation Column Synthesis, Figure 6
Optimal reactive distillation column for ethylene glycol production

the same as given earlier in Table 1. The difference from the example problem studied in [6] is the objective, which is the minimization of operating cost only. The set of streams include the intermediate streams L{EO, H2O, EG, DEG} and V{EO, H2O, EG, DEG} and the product streams L(EG) and L(DEG). Five liquid-liquid mass/heat exchange matches and 15 liquid-vapor mass/heat exchange matches are considered. Representing each match with a binary variable, and considering all possible interactions between units, the problem is formulated as a mixed integer nonlinear programming problem with the objective of minimizing operating cost, which includes raw material cost, purification, and utility cost. The optimal reactive distillation column obtained is pictured in Fig. 6. The column has two reaction zones and multiple feeds, and the operating cost is 1.17×10^6 \$/yr.

Conclusions

This paper discussed the MINLP applications in reactive distillation design problems. Two main approaches are studied: distillation based superstructure approach that uses rigorous tray-by-tray method to model reactive distillation, and heat and mass exchanger network superstructure approach that realizes reactive distillation processes as combinations of several mass/heat exchangers with a condenser and a reboiler. Examples are included to demonstrate the approaches.

See also

- ▶ Chemical Process Planning
- ▶ Extended Cutting Plane Algorithm
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Mixed Integer Nonlinear Programming

References

1. Agreda VH, Partin LR, Heise WH (1990) High purity methyl acetate via reactive distillation. Chem Eng Prog 86(2):40–46

2. Barbosa D, Doherty MF (1988) Design and minimum reflux calculations for double-feed multicomponent reactive distillation columns. Chem Eng Sci 43:2377
3. Barbosa D, Doherty MF (1988) Design and minimum reflux calculations for single-feed multicomponent reactive distillation columns. Chem Eng Sci 43:1523
4. Barbosa D, Doherty MF (1988) The influence of equilibrium chemical reactions on vapor-liquid phase diagrams. Chem Eng Sci 43:529
5. Barbosa D, Doherty MF (1988) The simple distillation of homogeneous reactive mixtures. Chem Eng Sci 43:541
6. Ciric AR, Gu D (1994) A mixed integer nonlinear programming approach to nonequilibrium reactive distillation synthesis. AIChE J 40(9):1479
7. Doherty MF, Buzad G (1992) Reactive distillation by design. Trans Inst Chem Eng 70:448–458
8. Gumus ZH, Ciric AR (1997) Reactive distillation column design with vapor/liquid/liquid equilibria. Comput Chem Eng 21:S983–988
9. Jacobs DB, Zimmermann J (1977) Chap. 12. In: Schildknecht CE (ed) Polymerization Processes. Wiley, New York
10. Papalexandri KP, Pistikopoulos EN (1996) A generalized modular representation framework for process synthesis based on mass/heat transfer principles. AIChE J 42(4): 1010
11. Parker AS (1958) Preparation of alkylene glycol. US Patent 2, 839, 588
12. Shoemaker JD, Jones EM (1987) Cumene by catalytic distillation. Hydrocarbon Proc June:57–58
13. Smith LA (1984) Method for the preparation of methyl tertiary butyl ether. US Patent 4, 978, 807
14. Ung S, Doherty MF (1995) Synthesis of reactive distillation systems with multiple equilibrium chemical reactions. Industr Eng Chem Res 34:2555–2565
15. Viswanathan J, Grossmann IE (1993) Optimal feed locations and number of trays for distillation columns with multiple feeds. I–EC Res 32:2942–2949

MINLP: Trim-loss Problem

IIRO HARJUNKOSKI[1], RAY PÖRN[2], TAPIO WESTERLUND[3]

[1] Process Design Lab., Åbo Akad. University, Turku, Finland

[2] Department Math., Åbo Akad. University, Turku, Finland

[3] Process Design Lab., Åbo Akad. University, Åbo, Finland

MSC2000: 90C11, 90C90

Article Outline

Keywords

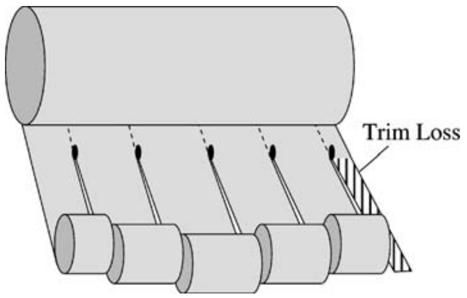
Scheduling; Paper converting; Trim-loss problem; Bilinear; Convex transformation

The trim-loss problem is one of the most demanding optimization problems in the paper-converting industry. It appears when an order specified by a customer is to be satisfied by cutting out a set of product reels from a wider raw paper reel.

The products in the order are characterized by width and quality. In a paper-converting mill the raw paper can be printed, coated and cut. In a typical paper-converting mill, there may be hundreds of different products to be produced. When considering the trim-loss problem, width is the most important property while the main problem is to determine such cutting patterns that minimize waste production, the *trim loss*.

In the optimization problem, beyond the number of cutting patterns needed, the appearance of each cutting pattern needs to be determined at the same time as having to decide how many times the cutting patterns ought to be repeated.

The customer widths and the raw paper widths are often more or less independent of each other. This makes it combinatorially very demanding to produce a cutting plan that minimizes the trim loss. Even if the trim-loss problem is in its basic form an integer problem, it has often been solved by linear programming (LP) methods [3] or some heuristic algorithms



MINLP: Trim-loss Problem, Figure 1
The cutting procedure

[4]. A good survey of widely used solution methods for trim-loss and assortment problems is given in [7].

When using an LP-approach to solve an integer problem the biggest difficulty is to convert the continuous solution such that the integer variables obtain integer values. The rounding methods are heuristic [8] and often fail to give the optimal integer solution even though the solution may be fairly good.

Problem Formulation

The trim-loss problem is a bilinear nonconvex integer nonlinear programming (INLP) problem. The appearance of a cutting pattern needs to be determined by integer variables and the bilinearity comes from the demand constraints.

A cutting pattern tells how many times a certain product is cut out from the raw paper. Let a cutting pattern have the index j and a product the index i. Assume a customer demand with I different products and further assume that the maximum allowed number of different cutting patterns is J. Further let m_j be the number of times a certain cutting pattern is repeated and n_{ij} be the number of times a product i appears in cutting pattern j. If the demand of a product i is expressed by $n_{i,\,\mathrm{order}}$, the demand constraints can be written as

$$\begin{aligned} & n_{i,\mathrm{order}} - \sum_{j=1}^{J} m_j \cdot n_{ij} \leq 0, \\ & i = 1, \dots, I, \\ & m_j, n_{ij} \in \mathbb{Z}^+. \end{aligned} \tag{1}$$

The negative bilinear terms make the problem nonconvex. Both of the variables in the term are integer variables and consequently the problem is a bilinear integer optimization problem. It is not possible to replace one of the variables n_{ij} with a continuous variable because this would violate the product specification. In theory it is possible to replace the m_j with a continuous variable but this may easily dissatisfy the desired product reel length and diameter requirements. Therefore, in the following study it is preferable to keep both m_j and n_{ij} as integers.

While raw paper reels of the same width are often glued together to form a continuous raw paper reel the problem can be simplified by omitting the raw paper length and assuming that the pattern lengths are equal.

Besides the demand constraint, certain constraints are needed to keep the problem feasible. Let the width of a product i be expressed by b_i and the width of the raw paper used for cutting pattern j by $B_{j,\,\max}$. The trim-loss width cannot exceed, for instance, 200mm owing to the machinery. This limit is represented by Δ_j. Furthermore, the maximum number of products that can be cut out from a pattern often has a physical restriction. The outcoming product reels have to form an angle big enough so that the reels do not attach together, yet with too big an angle between the outermost reels the paper may be torn off. Let this upper limit be $N_{j,\,\max}$.

Besides the total number of patterns, the pattern changes are also of interest when doing the optimization. This is due to the fact that the machinery normally needs to be stopped for a knife change which causes a production stop. Let therefore the variable y_j be 1 if the cutting pattern j exists and 0 if not. The sum of y_j variables then indicates how many different cutting patterns are needed to satisfy the production and the sum of m_j indicates the total number of all patterns which are related to the running metres of the raw material.

Now the basic formulation can be written in mathematical form. The objective is to minimize the total number of patterns and the number of pattern changes.

$$\min_{m_j, n_{ij}, y_j} \left\{ \sum_{j=1}^{J} c_j \cdot m_j + C_j \cdot y_j \right\} \tag{2}$$

subject to

$$\sum_{i=1}^{I} b_i \cdot n_{ij} - B_{j,\max} \leq 0, \tag{3}$$

$$-\sum_{i=1}^{I} b_i \cdot n_{ij} + B_{j,\max} - \Delta_j \le 0, \tag{4}$$

$$\sum_{i=1}^{I} n_{ij} - N_{j,\max} \le 0, \tag{5}$$

$$y_j - m_j \le 0, \tag{6}$$

$$\begin{aligned} & m_j - M_j \cdot y_j \le 0, \\ & j = 1, \dots, J, \end{aligned} \tag{7}$$

$$\begin{aligned} & n_{i,\text{order}} - \sum_{j=1}^{J} m_j \cdot n_{ij} \le 0, \\ & i = 1, \dots, I, \\ & m_j,\ n_{ij} \in \mathbb{Z}, \quad y_j \in \{0,1\}. \end{aligned} \tag{8}$$

The M_j gives the upper bound for corresponding m_j variables. When using an objective as in (2) the constraint (6) becomes irrelevant. The width constraints are given in (3)–(4) and the constraint (5) restricts the number of cuts in a pattern. The binary variables, y_j, are defined in (6)–(7).

The functionality of the variables are demonstrated in the following figure where the raw-paper width is $B_{j,\max}$. Note that the pattern length may typically be e. g. 6500m.

The last constraint, the demand constraint (8), is an integer bilinear constraint where both variables in bilinear terms are pure integers. This makes the problem a nonconvex MINLP problem where the nonconvexity appears in the integer variables.

There are very few methods available that are capable of solving similar nonconvex MINLP problems. Some heuristic methods such as simulated annealing [9] may find the global optimal solution within infinite time but algorithmic methods have not been proven to converge with such types of problems. Only recently (1999) some advancements have been reported in [1] and [11].

However, it is fully possible to transform the trim-loss problem into convex or linear form and use some established MINLP or MILP solver to solve the resulting problem to global optimality. Some linear transformations are presented in [6] and methods to transform the nonconvex problem into a convex form can be found in [10] and [5].

Linear Transformations

As can be seen from (2)–(8), all constraints but the last demand constraint are linear. This means that the problem should be fairly well bounded already by the linear part of the problem and thus a linear formulation strategy seems to be fully possible.

However, this linear transformation requires new variables and constraints that may complicate the problem. Using a standard approach, by rewriting one of the integer variables in the bilinear term by binary variables, the following is obtained.

$$\begin{aligned} & m_j = \sum_{k=1}^{K} 2^{k-1} \cdot \beta_{jk}, \\ & m_j \in \mathbb{R}, \quad \beta_{jk} \in \{0,1\}. \end{aligned} \tag{9}$$

K is the number of binary variables needed. By defining L_{ij} to be the upper bound for respective n_{ij} variables and introducing a new slack-variable s_{ijk} the following constraints will create a necessary link between the n_{ij} and s_{ijk} variables:

$$s_{ijk} - n_{ij} \le 0, \tag{10}$$

$$-s_{ijk} + n_{ij} - L_{ij} \cdot (1 - \beta_{jk}) \le 0, \tag{11}$$

$$s_{ijk} - L_{ij} \cdot \beta_{jk} \le 0. \tag{12}$$

Using the above constraints the bilinear demand constraint can be written in linear form

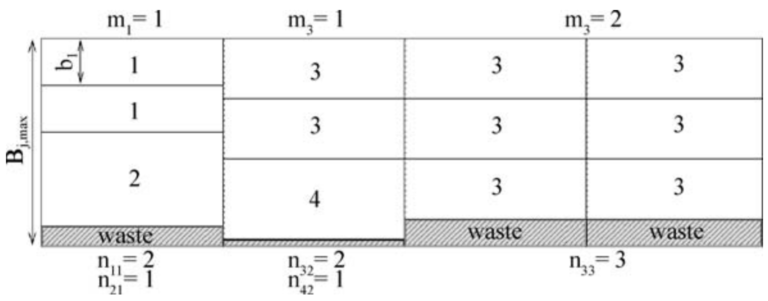
$$n_{i,\text{order}} - \sum_{j=1}^{J} \sum_{k=1}^{K} 2^{k-1} \cdot s_{ijk} \le 0. \tag{13}$$

The m_j could also be represented by special ordered sets (SOS) where at most, one of the binary variables are allowed to be nonzero.

$$m_j = \sum_{k=1}^{K} k \cdot \beta_{jk}\,, \qquad \sum_{k=1}^{K} \beta_{jk} \le 1\,. \tag{14}$$

It should be noted that the usage of this kind of transformation may enlarge the integrality gap unless for instance the n_{ij} variables in equations (3)–(5) are replaced with corresponding variables s_{ijk}.

The same transformation can be modified such that n_{ij} is replaced by a binary representation and m_j is defined through the slack-variables s_{ijk}.



MINLP: Trim-loss Problem, Figure 2
The integer variables

Parameterization Methods

Beyond the linear transformation, the problem can be written in linear form by simply parameterizing one of the variables in the bilinear term. This method though may lead to global optimality only in such cases where all the possible combinations have been considered. This strategy may be good for smaller problems but it may also generate far too many integer variables in solving larger trim-loss problems.

It is quite easy to generate all the possible combinations of n_{ij} variables satisfying the constraints (3)–(5). This strategy results in a problem where these constraints can be removed and where the n_{ij} variables in the resulting linear demand constraint are parameters:

$$n_{i,\text{order}} - m_j \cdot n'_{ij} \le 0\,, \qquad m_j \in \mathbb{Z}\,. \tag{15}$$

The same type of parameterization strategy may also be applied to the other variable m_j but in this case it may be more difficult to define the exact values of the parameters. One strategy is to use the upper bounds M_j or define all the m_j variables to be equal to one and make sure that a sufficient amount of the variables m_j are considered.

Another alternative is to combine the parameterization and transformation methods so that a proper amount of parameterized variables are combined with original variables. This strategy may be very efficient but often requires such information that may be difficult to obtain from a larger problem without any knowledge of the solution.

Convex Transformations

In the previous sections a number of methods were presented where the nonconvex problem can be transformed or parameterized into linear form. The main drawback for this linear transformation strategy is the large number of extra constraints and continuous variables. The parameterization strategy results in a formulation with a few constraints but many extra integer variables.

In the following a number of convexification methods are presented. Generally, the convex formulations need fewer extra constraints and continuous variables as the linear strategies and no extra integer variables as is the case with the parameterization methods. Thus, the convex transformation could be expected to result in formulations which are easier to solve especially for larger-scale orders. This creates an interesting problem, where the integer search space is reduced at the expense of more complex nonlinear functions, which could, in principle, be used as benchmarks for the performance of MINLP algorithms.

The basic principle for the convex transformation is to first expand the bilinearity in the demand constraint

$$m_j \cdot n_{ij} = (m_j + \tau)(n_{ij} + \tau) - \tau \cdot (m_j + n_{ij}) - \tau^2\,. \tag{16}$$

In the following text, the translation constant $\tau = 1$ is used for simplicity. The second step is to substitute the bilinear term in the original demand constraint

$$n_{i,\text{order}} - \sum_{j=1}^{J} (m_j + 1)(n_{ij} + 1) + \sum_{j=1}^{J} (m_j + n_{ij}) + J \le 0\,. \tag{17}$$

It should be noted that the transformations that follow need to consider the whole problem not only individual functions, which makes the transformation techniques more demanding. A transformation of a single function may cause linear constraints to become nonlinear if one is unaware of this fact.

Exponential Transformation

The demand constraint is originally a negative bilinear constraint. The exponential transformation can only be applied to a positive bilinear constraint. Therefore, one of the variables in the bilinear term needs to be substituted with its reversed value.

$$r_{ij} = N_{j,\max} - n_{ij} \tag{18}$$

and the demand constraint is modified to

$$n_{i,\text{order}} - \sum_{j=1}^{J} m_j \cdot N_{j,\max} + \sum_{j=1}^{J} m_j \cdot r_{ij} . \tag{19}$$

Now the exponential transformation can be applied. The transformation is of the form

$$m_j + 1 = e^{M_j}, \quad r_{ij} + 1 = e^{R_{ij}} \tag{20}$$

and the variables are defined as

$$m_j = \sum_{l=1}^{L_j} \beta_{jl} \cdot l, \tag{21}$$

$$M_j = \sum_{l=1}^{L_j} \beta_{jl} \cdot \ln(l+1), \tag{22}$$

$$r_{ij} = \sum_{k=1}^{K_i} \beta_{ijk} \cdot k, \tag{23}$$

$$R_{ij} = \sum_{k=1}^{K_i} \beta_{ijk} \cdot \ln(k+1), \tag{24}$$

$$\sum_{l=1}^{L_j} \beta_{jl} \le 1, \quad \sum_{k=1}^{K_i} \beta_{ijk} \le 1, \tag{25}$$

$$\beta_{jl}, \beta_{ijk} \in \{0,1\}, \quad M_j, R_{ij} \in \mathbb{R}.$$

When combining these definitions, the demand constraint can be written in convex form

$$n_{i,\text{order}} - J + \sum_{j=1}^{J} e^{M_j + R_{ij}} - \sum_{j=1}^{J} \left((N_{j,\max} + 1) \cdot \sum_{l=1}^{L_j} \beta_{jl} \cdot l + \sum_{k=1}^{K_i} \beta_{ijk} \cdot k \right) \le 0. \tag{26}$$

This transformation can also be achieved in slightly another way but using this strategy also requires updating some of the constraints in (3)–(7).

Square-Root Transformation

This transformation is almost equivalent to the previous one. A main difference is that it can be applied straight to the negative bilinear constraint and thus no r_{ij} variables need to be defined. The constraint (21) is valid but the constraint (23) needs to be modified to

$$n_{ij} = \sum_{k=1}^{K_i} \beta_{ijk} \cdot k . \tag{27}$$

Note that the equations in (25) are valid. The transformation is of the form

$$m_j + 1 = \sqrt{M_j}, \quad n_{ij} + 1 = \sqrt{N_{ij}} . \tag{28}$$

The transformation variables M_j and N_{ij} are defined as

$$M_j = 1 + \sum_{l=1}^{L_j} \beta_{jl} \cdot l(l+2), \tag{29}$$

$$N_{ij} = 1 + \sum_{k=1}^{K_i} \beta_{ijk} \cdot k(k+2), \tag{30}$$

$$\beta_{jk}, \beta_{ijk} \in \{0,1\}, \quad M_j, N_{ij} \in \mathbb{R}.$$

and the resulting convex demand constraint is

$$n_{i,\text{order}} + J - \sum_{j=1}^{J} \sqrt{M_j \cdot N_{ij}} + \sum_{j=1}^{J} \left(\sum_{l=1}^{L_j} \beta_{jl} \cdot l + \sum_{k=1}^{K_i} \beta_{ijk} \cdot k \right) \le 0. \tag{31}$$

Logarithmic and Square-Root Transformation

The square-root and the logarithmic functions can be combined, resulting in a third convex transformation. It is directly applicable to the negative bilinear function and the transformation can be written as

$$m_j + 1 = \sqrt{M_j}, \qquad n_{ij} + 1 = \ln N_{ij}. \tag{32}$$

The m_j, n_{ij} and M_j variables are defined as in the square-root transformation and the N_{ij} is defined as

$$N_{ij} = e + \sum_{k=1}^{K_i} \beta_{ijk} \cdot (e^{k+1} - e) \tag{33}$$

and the following convex demand constraint is obtained

$$n_{i,\text{order}} + J - \sum_{j=1}^{J} \sqrt{M_j} \cdot \ln N_{ij} + \sum_{j=1}^{J} \left(\sum_{l=1}^{L_j} \beta_{jl} \cdot l + \sum_{k=1}^{K_i} \beta_{ijk} \cdot k \right) \leq 0. \tag{34}$$

It can be noted in equation (34) that the only difference to the former transformation is the third term of the demand constraint.

Inverted Transformation

The following transformation can be applied to a positive bilinear constraint. Thus the same definition of r_{ij} has to be done as for the exponential transformation. The transformation has the form

$$m_j + 1 = \frac{1}{M_j}, \quad r_{ij} + 1 = \frac{1}{R_{ij}}. \tag{35}$$

The definitions of the transformation variables follow:

$$M_j = 1 + \sum_{l=1}^{L_j} \beta_{jl} \cdot \left(\frac{1}{l+1} - 1 \right), \tag{36}$$

$$R_{ij} = 1 + \sum_{k=1}^{K_i} \beta_{ijk} \cdot \left(\frac{1}{k+1} - 1 \right). \tag{37}$$

The demand constraint is obtained exactly in the same way as before

$$n_{i,\text{order}} - J + \sum_{j=1}^{J} \frac{1}{M_j \cdot R_{ij}} - \sum_{j=1}^{J} \left((N_{\max} + 1) \cdot \sum_{l=1}^{L_j} \beta_{jl} \cdot l + \sum_{k=1}^{K_i} \beta_{ijk} \cdot k \right) \leq 0. \tag{38}$$

Modified Square-Root Transformation

As the last transformation, a modification to the previously presented square-root transformation is introduced. In such cases where the variable m_j may take large values, it may be more efficient to use another type of binary representation.

$$m_j = \sum_{l=1}^{L'_j} 2^{l-1} \cdot \beta_{jl}, \tag{39}$$

where $L_j' = \lfloor \log_2(m_{j,\max}) \rfloor + 1$ if $m_{j,\max}$ is the upper bound for the respective m_j variable. This modification reduces the required number of binary variables and the transformation variable M_j needs to be redefined. The definition also requires additional slack-variables and constraints. In the following, the square-root transformation is used:

$$M_j = 1 + \sum_{l=1}^{L'_j} (s^{2l-2} + 2^l) \cdot \beta_{jl} + \sum_{l,m=1;m<l}^{L'_j} 2^{l+m-1} \cdot s_{jlm}, \tag{40}$$

$$-s_{jlm} - 1 + \beta_{jl} + \beta_{jm} \leq 0, \tag{41}$$

$$2 \cdot s_{jlm} - \beta_{jl} - \beta_{jm} \leq 0, \quad l, m = 1, \ldots, L'_j; \quad m < l. \tag{42}$$

By adding the extra constraints and defining N_{ij} as in the square-root strategy, the demand constraint can be written in convex form as follows

$$n_{i,\text{order}} + J - \sum_{j=1}^{J} \sqrt{M_j \cdot N_{ij}} + \sum_{j=1}^{J} \left(\sum_{l=1}^{L_j} \beta_{jl} \cdot 2^{l-1} + \sum_{k=1}^{K_i} \beta_{ijk} \cdot k \right) \leq 0. \quad (43)$$

Five methods for transforming the originally nonconvex trim-loss problem into convex form have been discussed. Three of them were directly applicable to a negative bilinear function but for two methods some operations were needed to change the demand constraint into a positive bilinear constraint.

Example: A Numerical Problem

In this last section a numerical example is solved with all of the presented methods. To improve the performance of the solution procedure some extra linear constraints need to be defined. They are, however, not specified here.

In the following example order an upper limit for products $n_{i,\max}$ that are allowed to be produced also has been defined. Here, the maximal possible overproduction of any product is 2. This limit is somewhat unnatural and is therefore not used as a constraint. However, the use of this type of upper bounds makes it possible to efficiently reduce the combinatorial space.

i	b_i(mm)	$n_{i,\text{order}}$	$n_{i,\max}$
1	330	8	10
2	360	16	18
3	380	12	14
4	430	7	9
5	490	14	16
6	530	16	18

Example order

The example demand is a mid-size customer order with a total weight of 27.5tons. Some important parameters need to be defined before optimization. The raw paper width of 2200mm is chosen and a maximal trim loss of 100mm is tolerated. At most 5 products may be cut out from a cutting pattern. Among the following parameters, the parameter M_j refers to the upper bound of the respective m_j variable and the parameter N_i to the n_{ij} variables. Note, that since the raw paper width is equal for every pattern the latter upper bound is independent of the index j.

$J = I = 6$	$N_{j,\max} = 5$
$c_j = 1$	$M_j = \{14, 12, 8, 7, 4, 2\}$
$C_j = 0.1$	$N_i = \{2, 3, 3, 5, 3, 4\}$
$B_{j,\max} = 2200\text{mm}$	$M_{\min} = 15$
$\Delta_j = 100\text{mm}$	

The problem parameters

The parameter $M_{\min}$ is the lower bound for the sum of the variables m_j. This sum can easily be calculated in advance and significantly enhances the optimization performance.

Before doing the actual optimization it should be pointed out that the results are not comparable. The main purpose for showing the numerical results is to demonstrate that the above presented strategies are fully usable and result in quite efficient solvable formulations. The transformation strategies can be directly applied to any problem where the bilinear terms contain integer variables.

The methods are divided into three groups of which the linear transformation and the parameterization strategies result into MILP formulations. The third group, the convex transformation strategy produces MINLP formulations that have in this case been solved using the extended cutting plane (ECP) algorithm by T. Westerlund and F. Peterssen [12].

In the parameterization strategies the problem is redefined by parameterizing certain variables which means that the resulting problem has already been partly solved. This may, however, not always be a benefit, especially in such problems where a huge number of parameters increases the integer search space for other variables.

The strategies are numbered as follows:

1.	binary representation of m_j
2.	binary representation of n_{ij}
3.	parameterization of n_{ij}
4.	parameterization of m_j
5.	exponential transformation
6.	square-root transformation
7.	logarithmic and square-root transformation
8.	inverted transformation
9.	modified square-root transformation

The strategies enlarge the problem both in terms of variables and constraints. In the following the number of variables and constraints are given. All the constraints are linear except in the convex transformation strategies where six of the constraints are nonlinear.

The strategies 1–4 are linear formulations of which 3–4 use the parameterization strategy to overcome the bilinearity. Strategies 5–9 are convex transformations. The field with combinations gives simply the number of unconstrained discrete variable combinations as a function of number of binary variables. This information is more informative than just the number of variables.

Strategy	Constraints	Variables (I/B/C)	Comb. 2^n
1.	408	36/23/120	2^{98}
2.	366	6/88/144	2^{105}
3.	59	51/51/—	2^{140}
4.	201	282/47/—	2^{634}
5.	199	—/169/84	2^{96}
6.	199	—/169/84	2^{96}
7.	185	—/169/84	2^{96}
8.	185	—/169/84	2^{96}
9.	225	—/208/84	2^{118}

The MILP problems 1–4 were solved with CPLEX-5.0 using default settings and the MINLP problems 5–9 were solved by 'mittlp', an ECP application written by H. Skrifvars. The optimization was done on a Pentium Pro 200MHz running the Linux operating system.

The optimization results can be seen in the following table.

Strategy	Nodes (MILP)	ECP-iter. (MINLP)	CPU-time (s)
1.	265	-	7.6
2.	51	-	0.51
3.	2174	-	3.2
4.	265	-	7.7
5.	-	4	8.6
6.	-	7	66.6
7.	-	9	138.6
8.	-	10	736.4
9.	-	6	49.9

The optimal result has two cutting patterns with the widths $B_1 = 2110$ mm and $B_2 = 2170$ mm and multiples $m_1 = 8$, $m_2 = 7$. The appearances of the patterns are given by the following variables: $n_{1,1} = 1$, $n_{2,1} = 2$, $n_{6,1} = 2$, $n_{3,2} = 2$, $n_{4,2} = 1$, $n_{5,2} = 2$

Conclusions

The study above is not a fair comparison. Experience has shown that the performance order is highly dependent on the specific problem. In order to get an idea of which of the methods is, in average the most efficient one, tens of problems of different sizes need to be solved. However, the study illustrates that it is fully possible to apply the transformation methods to a well explored real industrial problem.

In the present study the trim-loss problem was used as an example case but the transformation methods are general and can be applied to any problem with similar type of bilinear constraints.

Notation

i	product index
j	cutting pattern index
I	number of products in the order
J	number of possible cutting patterns
m_j	number of times the pattern j is used
n_{ij}	number of product i in pattern j
r_{ij}	reversed value of n_{ij}
$n_{i,\text{order}}$	number of product i ordered
b_i	width of product i
$B_{j,\max}$	width of raw paper of pattern j
Δ_j	max. trim-loss width
$N_{j,\max}$	max. number of products in pattern j
y_j	binary variable that is one if $m_j > 0$
c_j, C_j	cost coefficients
M_j	upper bound / transformation variable
β_{jl}, β_{jk}	binary variables for defining m_j
L_{ij}	upper bound
s_{ijk}	slack-variable for linear transformations
β_{ijk}	binary variables for defining n_{ij} or r_{ij}
n_{ij}'	fixed n_{ij} values
τ	translation constant
N_{ij}	transformation variable
R_{ij}	transformation variable
l, k, m	indices of binary variables
L_j, K_i	number of binary variables needed

See also

- ▶ Branch and Price: Integer Programming with Column Generation
- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-dependent Traveling Salesman Problem

References

1. Adjiman CS, Androulakis IP, Floudas CA (1997) Global optimization of MINLP problems in process synthesis and design. Comput Chem Eng 21:S445–S450
2. Dakin RJ (1965) A tree search algorithm for mixed integer programming problems. Comput J 8:250–255
3. Gilmore PC, Gomory RE (1961) A linear programming approach to the cutting-stock problem. Oper Res 9:849–859
4. Haessler RW (1971) A heuristic programming solution to a non-linear cutting stock problem. Managem Sci 17:B793–B802
5. Harjunkoski I, Pörn R, Westerlund T, Skrifvars H (1997) Different strategies for solving bilinear integer problems with convex transformations. Comput Chem Eng 21:S487–S492
6. Harjunkoski I, Westerlund T, Isaksson J, Skrifvars H (1996) Different formulations for solving trim-loss problems in a paper converting mill with ILP. Comput Chem Eng 20:S121–S126
7. Hinxman AI (1980) The trim-loss and assortment problems: A survey. Europ J Oper Res 5:8–18
8. Johnston RE (1986) Rounding algorithms for cutting stock problems. Asia–Pacific J Oper Res 3:166–171
9. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
10. Skrifvars H, Harjunkoski I, Westerlund T, Kravanja Z, Pörn R (1996) Comparison of different MINLP methods applied on certain chemical engineering problems. Comput Chem Eng 20:S333–S338
11. Smith EMB, Pantelides CC (1997) Global optimization of nonconvex MINLPs. Comput Chem Eng 21:S791–S796
12. Westerlund T, Pettersson F (1995) An extended cutting plane method for solving convex MINLP problems. Comput Chem Eng 19:S131–S136

Mixed 0-1 Linear Programming Approach for DNA Transcription Element Identification

HAN-LIN LI[1], CHANG-JUI FU[2]
[1] Institute of Information Management, National Chiao Tung University, Hsinchu, Taiwan
[2] Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan

MSC2000: 90C08

Article Outline

Introduction/Background

In the past two decades, biologists have sequenced more and more complete genome sets for various species. To reveal secrets of life hiding in enormous genome data, the mechanism which conducts gene expression is continuously researched and discussed. Gene transcription, a primary gateway to gene function, is controlled by a complex regulatory mechanism. In this mechanism, many specific regulatory proteins bind to local regions of a gene upstream, called transcription factor binding sites (TFBS) or motifs, to control the gene expression. Therefore, the discrimination of TFBSs becomes an essential task for genome function analysis. Finding TFBSs is a challenging issue because motifs are mostly orientation- and position- independent to transcription starting points, and usually with some degree of ambiguity. Experimental methods like DNA microarray (DeRisi et al., 1997; Lockhart et al., 1996) and SAGE (Velculescu et al., 2000) are capable of precisely elucidating motifs, but too laborious and time consuming to analyze enormous genome data. More and more computer based methods - such as enumeration methods, probability models and heuristics - are being developed to help motif finding. The modeling of in silico motif finding has two parts: scoring function and algorithm. The simplest scoring function is given by summing up the number of base matches in a regulatory region. Generally it needs a predefined shared pattern for accuracy. Another scoring criterion is position-specific scoring matrices (PSSM) or its variant, information content (IC, Schneider et al., 1986), [44]. Though more computing is required, PSSM and IC are the most popular scoring functions, owing to their pattern-free property.

Current motif finding algorithms can generally be categorized as the probabilistic approaches and the deterministic approaches. Popular probabilistic algorithms are the expectation maximization [22], Gibbs sampling [21] and hidden Markov model (HMM). These are used to develop various sample-driven tools like MEME [3], CONSENSUS [17], AlignACE [19], ANN-spec [54], BioProspector [24], MotifSampler [48], GLAM [13], The Improbizer [1], QuickScore [38], SesiMCMC [11] and TFBSfinder [51].

There are many discrepancies among deterministic methods. A representative one is the consensus-based approach, [45] which tests all 4^m m-wide patterns and promises an optimal solution, but is very time consuming and impractical for large m [33,49]. Many heuristics are developed to prune the huge searching space, including testing only the substrings in the sequences [15,26], specifying a shared pattern to restrict the locations of mismatches [5,7,38,41], constructing suffix tree with fixed mismatches [30,31] and clustering approaches [6,23,34].

The methods for determining a consensus pattern can be split into two parts. The first part is the model for describing the shared pattern, and the second part is the algorithm for identifying the optimal consensus sequence according to its shared pattern. This study belongs to the second part. A consensus based motif finding problem is, given a set of sequences known to contain binding sites for a common factor but not knowing where the site are, to discover the location of the sites in each sequence [45].

Ecker et al. (2002) utilized optimization techniques to reformulate the maximum likelihood approach for motif finding problems. They adopted a probabilistic model and formulated a well-designed nonlinear model with reference to the expectation maximization algorithm of Lawrence and Reilly [22]. Their method, however, occasionally only finds a feasible solution or a local optimum, which means the best solution may not be found. Additionally, no further structural feature in the target motif can be embedded conveniently in their model.

Definitions

This study introduces a linear programming method for solving a motif finding problem to reach the globally optimal consensus sequence. Two examples of searching for CRP-binding sites and for FNR-binding sites in the *Escherichia coli* genome are used to illustrate the proposed method. The motif finding problem is firstly formulated as a nonlinear mixed 0-1 program for the alignment of DNA sequences; each of the four bases are coded with two binary variables and a matching score is designed. This nonlinear mixed 0-1 program is then converted into a linear mixed 0-1 program by linearization techniques. Owing to some special features of the binary relationships, this linear 0-1 program includes $2m$ binary variables where m is the number of active letters in the consensus. This method makes the number of binary variables independent of the number of sequences and the size of each sequence. That means the proposed method is computationally efficient in solving a motif finding problem with a large data size. Secondly, the proposed method is guaranteed to find the global optimum instead of a local optimum. Thirdly, many kinds of specific features accompanied with the target motif can be formulated as logical constraints and embedded into the linear program.

An example of searching CRP-binding sites, as discussed in Stormo et al. [44] and Ecker et al. (Ecker et al., 2002), is described as follows. Given eighteen letter sequences, each 105 positions long, where each position contains a letter from the set {A, T, C, G}, find a consensus sequence of length16 with the pattern

$$L_1L_2L_3L_4L_5 * * * * * * L_6L_7L_8L_9L_{10}$$

where $L_i \in${A, T, C, G} and $*$'s mean the positions of ignored letters.

Restated, the problem is to specify

(i) the L_i's of the consensus sequence pattern, and
(ii) the location of the site in each given sequence which can fit most closely the consensus sequence.

Formulation

This study firstly formulates a motif finding problem as a nonlinear mixed 0-1 program. This nonlinear mixed 0-1 program is then converted into a linear mixed 0-1 program using linearization techniques. To reduce the computational burden, many 0-1 variables in this linear mixed 0-1 program can actually be solved as continuous variables by an all or nothing assignment technique which greatly improves the computational efficiency of this program.

Here we use the example data in [44], as listed in Appendix, to describe the proposed method. First, we represent the data in Appendix as an 18*105 data matrix D:

$$D = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,105} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,105} \\ \vdots & \vdots & \ddots & \vdots \\ b_{18,1} & b_{18,2} & \cdots & b_{18,105} \end{bmatrix} \tag{1}$$

where $b_{l,p}$ is the letter in the position p of the sequence l.

Recall the example discussed in previous section: the consensus sequence we want to find has 16 positions (ten L_i's and six ignored letters). A sequence has 90 corresponding sites, so an 18*900 data matrix D' is generated from D.

$$D' = \begin{bmatrix} d^1_{1,1} & \cdots & d^{10}_{1,1} & d^1_{1,2} & \cdots & d^{10}_{1,2} & \cdots & d^1_{1,90} & \cdots & d^{10}_{1,90} \\ d^1_{2,1} & \cdots & d^{10}_{2,1} & d^1_{2,2} & \cdots & d^{10}_{2,2} & \cdots & d^1_{2,90} & \cdots & d^{10}_{2,90} \\ \vdots & & & & \vdots & & \ddots & & & \vdots \\ d^1_{18,1} & \cdots & d^{10}_{18,1} & d^1_{18,2} & \cdots & d^{10}_{18,2} & \cdots & d^1_{18,90} & \cdots & d^{10}_{18,90} \end{bmatrix} \tag{2}$$

where

$$d^i_{l,s} = \begin{cases} b_{l,i+s-1} & (\text{for } i = 1, 2, \ldots, 5) \\ b_{l,i+s+5} & (\text{for } i = 6, 7, \ldots, 10)\,, \end{cases}$$

and $s = 1 \ldots 90$ is the starting position of each candidate site.

Mixed 0-1 Linear Programming Approach for DNA Transcription Element Identification, Table 1
Base code in the determined consensus sequence

Base	u_i	v_i	a_i	t_i	c_i	g_i
A	0	0	1	0	0	0
T	1	1	0	1	0	0
C	0	1	0	0	1	0
G	1	0	0	0	0	1

For $L_i \in${A, T, C, G}, two binary variables u_i and v_i can be used to express L_i, an element of the consensus sequence, as shown in Table 1.

Table 1 indicates that if L_i is A, T, C, or G respectively, then $a_i = 1$, $t_i = 1$, $c_i = 1$ or $g_i = 1$, which implies following conditions.

$$\begin{aligned} a_i &= (1-u_i)(1-v_i) \\ t_i &= u_i v_i \\ c_i &= (1-u_i)v_i \\ g_i &= u_i(1-v_i) \end{aligned} \tag{3}$$

Now we let Score_l be the degree of fitting to the found consensus sequence, specified as

$$\text{Score}_l = \sum_{s=1}^{90} z_{l,s}\left(\theta^1_{l,s} + \theta^2_{l,s} + \ldots\cdots + \theta^{10}_{l,s}\right) \tag{4}$$

where $\theta^i_{l,s}$ is the element of candidate sites extracted from D'. The constraints associated with (4) are below:
(i)

$$\sum_{s=1}^{90} z_{l,s} = 1, \quad z_{l,s} \in \{0,1\} \text{ for all } l \text{ and } s\,. \tag{5}$$

(ii)

$$\theta^i_{l,s} = \begin{cases} a_i & \text{if } d^i_{l,s} = \text{A} \\ t_i & \text{if } d^i_{l,s} = \text{T} \\ c_i & \text{if } d^i_{l,s} = \text{C} \\ g_i & \text{if } d^i_{l,s} = \text{G}\,. \end{cases} \tag{6}$$

Clearly, $0 \le \text{Score}_l \le 10$, and the objective is to maximize the total sum of Score_l.

Methods/Applications

Consider the sample data in Fig. 1 for instance:

$$\begin{aligned} &\text{Score}_1 = \\ &z_{1,1}(a_1 + a_2 + g_3 + a_4 + c_5 + t_6 + t_7 + t_8 \\ &\quad + g_9 + a_{10}) \\ &+ z_{1,2}(a_1 + g_2 + a_3 + c_4 + t_5 + t_6 + t_7 + g_8 \\ &\quad + a_9 + t_{10}) \\ &z_{1,3}(g_1 + a_2 + c_3 + t_4 + g_5 + t_6 + g_7 + a_8 \\ &\quad + t_9 + c_{10}) \end{aligned} \tag{7}$$

$$\begin{aligned} &\text{Score}_2 = \\ &z_{2,1}(g_1 + a_2 + t_3 + t_4 + a_5 + c_6 + g_7 + g_8 \\ &\quad + c_9 + g_{10}) \\ &+ z_{2,2}(a_1 + t_2 + t_3 + a_4 + t_5 + g_6 + g_7 + c_8 \\ &\quad + g_9 + t_{10}) \\ &+ z_{2,3}(t_1 + t_2 + a_3 + t_4 + t_5 + g_6 + c_7 + g_8 \\ &\quad + t_9 + c_{10}) \end{aligned} \tag{8}$$

All $z_{l,s}$ in (4) are binary variables. Equation (5) implies that for a sequence l, only one site is chosen to contribute to Score_l. Suppose the kth site is selected, then $z_{l,k} = 1$ and $z_{l,s} = 0$ for all $s \in \{1, 2, \ldots, 90\}$, $s \neq k$. Since a huge amount of $z_{l,s}$ (i. e., $|l| * |s|$) are involved, to treat $z_{l,s}$ as binary variables would cause a heavy computational burden. Therefore $z_{l,s}$ should be resolved as continuous variables rather than binary variables. An important proposition is introduced below:

Proposition 1 (All or nothing assignment) *Let $z_{l,s} \ge 0$ be continuous variables instead of binary variables. If there is a k, $k \in \{1, 2, \ldots, 90\}$, such that $\sum_{i=1}^{10} \theta^i_{l,k} = \max\left\{\sum_{i=1}^{10} \theta^i_{l,s} \text{ for } s = 1, 2, \ldots, 90\right\}$, then assigning $z_{l,k} = 1$ and $z_{l,s} = 0$ for all $s \neq k$, $s \in \{1, 2, \ldots, 90\}$, can maximize the value of Score_l.*

Proof Since $\sum_s z_{l,s} = 1$ and $z_{l,s} \ge 0$, it is true that $\max\left\{\sum_s (z_{l,s} \sum_i \theta^i_{l,s})\right\} \le \max\left\{\sum_i \theta^i_{l,s} \text{ for } s = 1, 2, \ldots, 90\right\} = \sum_i \theta^i_{l,k}$. □

Remark 1 The objective function $\sum_l Score_l$ can be rewritten as

$$f(x) = \sum_{i=1}^{10} \left\{ a_i \sum_{(l,s)\in SA_i} z_{l,s} + t_i \sum_{(l,s)\in ST_i} z_{l,s} \right. \\ \left. + c_i \sum_{(l,s)\in SC_i} z_{l,s} + g_i \sum_{(l,s)\in SG_i} z_{l,s} \right\} \quad (9)$$

where $SA_i = \{(l,s)|d^i_{l,s} = A\}$, $ST_i = \{(l,s)|d^i_{l,s} = T\}$, $SC_i = \{(l,s)|d^i_{l,s} = C\}$, and $SG_i = \{(l,s)|d^i_{l,s} = G\}$ for $i = 1, 2, \dots, 10$.

This result implies that SA_i (or ST_i, SC_i, SG_i) is a set composed of (l,s) in which the product term $z_{l,s}a_i$ (or $z_{l,s}t_i$, $z_{l,s}c_i$, $z_{l,s}g_i$ respectively) appears on the right hand side of (4) because $\theta^i_{l,s} = a_i$.

For instance, the sum of Score_1 and Score_2 in (7) and (8) becomes

$$\text{Score}_1 + \text{Score}_2 = a_1(z_{1,1} + z_{1,2} + z_{2,2}) + \dots \\ + a_{10}z_{1,1} + \dots + g_1(z_{1,3} + z_{2,1}) + \dots + g_{10}z_{2,1}\,. \quad (10)$$

Some logical constraints can be conveniently expressed by binary variables. For instance, the constraint that a CRP dimer binds a symmetrical site requires that

$$\text{if } L_i = \begin{cases} \text{A} & \text{then } L_{11-i} = \text{T}\,, \\ \text{C} & \text{then } L_{11-i} = \text{G}\,. \end{cases}$$

Such a logical structure can be conveniently formulated with the following constraints:

$$\left.\begin{aligned} u_i + u_{11-i} &= 1 \\ v_i + v_{11-i} &= 1 \end{aligned}\right\} \text{ for } i = 1,2,3,4,5 \quad (11)$$

where $u_i, v_i, u_{11-i}, v_{11-i} \in \{0,1\}$.

With reference to Table 1, clearly if $L_i = A$ (i. e., $u_i = 0$ and $v_i = 0$) then $L_{11-i} = T$ (i. e., $u_{11-i} = 1$ and$v_{11-i} = 1$) and vice versa; (ii) if $L_i = C$ (i. e., $u_i = 0$ and $v_i = 1$) then $L_{11-i} = G$ (i. e., $u_{11-i} = 1$ and $v_{11-i} = 0$) and vice versa.

Models

A motif finding problem can be formulated as a nonlinear mixed 0-1 program based on these constraints:

Program 1 (Nonlinear Mixed 0-1 Program)

Maximize

$$\sum_{l=1}^{18} \text{Score}_l = \sum_{i=1}^{10} \left\{ a_i \sum_{(l,s)\in SA_i} z_{l,s} \right. \\ + t_i \sum_{(l,s)\in ST_i} z_{l,s} + c_i \sum_{(l,s)\in SC_i} z_{l,s} \\ \left. + g_i \sum_{(l,s)\in SG_i} z_{l,s} \right\} \quad (12)$$

subject to $\sum_{s=1}^{90} z_{l,s} = 1\,, \quad z_{l,s} \ge 0$ for all l, s

$$\left.\begin{aligned} a_i &= (1-u_i)(1-v_i) \\ t_i &= u_i v_i \\ c_i &= (1-u_i)v_i \\ g_i &= u_i(1-v_i) \end{aligned}\right\} \begin{array}{l}\text{Conservative} \\ \text{constraints for} \\ i = 1,2,\dots,10\end{array}$$

$$\left.\begin{aligned} u_i + u_{11-i} &= 1 \\ v_i + v_{11-i} &= 1 \end{aligned}\right\} \begin{array}{l}\text{Logical constraints} \\ \text{for } i = 1,2,\dots,5\end{array}$$

$u_i, v_i \in \{0,1\}$ for $i = 1,2,\dots,5$

$0 \le u_i, v_i \le 1$ for $i = 6,7,\dots,10$

$0 \le a_i, t_i, c_i, g_i \le 1$ for $i = 1,2,\dots,10$.

This program intends to solve $\{a_i, t_i, c_i, g_i\}$ for $i = 1, 2, \dots, 10$ thus to maximize the total degree of fitting to the consensus sequence for the given 18 sequences, subjected to a possible logical constraint. A very important feature of Program 1 is that we can treat $z_{l,s}$ as continuous variables rather than binary variables, which can improve the computational efficiency dramatically. We can ensure all found $z_{l,s}$ still have binary values as discussed in the next section.

Linearization of Program 1 Program 1 is a mixed nonlinear 0-1 program where $q_i \sum z_{l,s}$ for $q_i \in \{a_i, t_i, g_i, c_i\}$ and $u_i v_i$ are product terms. These product terms can be linearized directly by the following propositions:

Proposition 2 *The product term $\lambda_i = q_i \sum z_{l,s}$, where λ_i is to be maximized and $q_i \in \{0,1\}$, can be linearized* *as follows:*

$$\begin{aligned} &\lambda_i \geq \sum z_{l,s} + M(q_i - 1) \\ &\lambda_i \geq 0 \\ &\lambda_i \leq \sum z_{l,s} \\ &\lambda_i \leq M\, q_i \end{aligned} \tag{13}$$

where M is a big constant larger than or equal to the number of sequences.

Proof If $q_i = 1$ then $\lambda_i = \sum z_{l,s}$; and otherwise $\lambda_i = 0$.

□

Proposition 3 *The product term $w_i = u_i v_i$, where $u_i, v_i \in \{0, 1\}$, can be linearized as follows:*

$$\begin{aligned} &w_i \leq u_i \\ &w_i \leq v_i \\ &w_i \geq 0 \\ &w_i \geq u_i + v_i - 1\,. \end{aligned} \tag{14}$$

Denote $Z(a_i) = a_i \sum_{(l,s)\in SA_i} z_{l,s}$, $Z(t_i) = t_i \sum_{(l,s)\in ST_i} z_{l,s}$, $Z(c_i) = c_i \sum_{(l,s)\in SC_i} z_{l,s}$, and $Z(g_i) = g_i \sum_{(l,s)\in SG_i} z_{l,s}$. Program 1 is then linearized into Program 2 based on Proposition 2 and Proposition 3.

Program 2 (Linear Mixed 0-1 Program)

$$\text{Maximize} \quad \sum_{l=1}^{18} \text{Score}_l = \sum_{i=1}^{10} \big(Z(a_i) + Z(t_i) + Z(c_i) + Z(g_i)\big) \tag{15}$$

subject to

$$\sum_{s=1}^{90} z_{l,s} = 1, \qquad z_{l,s} \geq 0 \quad \text{for all } l, s$$

$$\left.\begin{aligned} &a_i = 1 - u_i - v_i + w_i \\ &t_i = w_i \\ &c_i = v_i - w_i \\ &g_i = u_i - w_i \\ &w_i \leq u_i \\ &w_i \leq v_i \\ &w_i \geq 0 \\ &w_i \geq u_i + v_i - 1 \end{aligned}\right\} \begin{array}{c}\text{Conservative constraints}\\ \text{for } i = 1, 2, \ldots, 10\end{array}$$

$$\left.\begin{aligned} &u_i + u_{11-i} = 1 \\ &v_i + v_{11-i} = 1 \end{aligned}\right\} \text{Logical constraints}$$

for $i = 1, 2, \ldots, 5$

$$\left.\begin{aligned} &\sum_{(l,s)\in SA_i} z_{l,s} + M(a_i - 1) \leq Z(a_i) \leq \sum_{(l,s)\in SA_i} z_{l,s} \\ &0 \leq Z(a_i) \leq M\, a_i \\ &\sum_{(l,s)\in ST_i} z_{l,s} + M(t_i - 1) \leq Z(t_i) \leq \sum_{(l,s)\in ST_i} z_{l,s} \\ &0 \leq Z(t_i) \leq M\, t_i \\ &\sum_{(l,s)\in SC_i} z_{l,s} + M(c_i - 1) \leq Z(c_i) \leq \sum_{(l,s)\in SC_i} z_{l,s} \\ &0 \leq Z(c_i) \leq M\, c_i \\ &\sum_{(l,s)\in SG_i} z_{l,s} + M(g_i - 1) \leq Z(g_i) \leq \sum_{(l,s)\in SG_i} z_{l,s} \\ &0 \leq Z(g_i) \leq M\, g_i \end{aligned}\right\} \text{Constraints for linearizing product terms}$$

$u_i, v_i \in \{0, 1\}$ for $i = 1, 2, \ldots, 5$

$0 \leq u_i, v_i \leq 1$ for $i = 6, 7, \ldots, 10$

$0 \leq a_i, t_i, c_i, g_i \leq 1$ for $= 1, 2, \ldots, 10$

$z_{l,s}$'s are treated as non-negative continuous variables for $l = 1, 2, \ldots, 18$ and $s = 1, 2, \ldots, 90$ where M can be any value greater than or equal to 18 .

In Program 2, since u_i and v_i are binary variables, a_i, t_i, c_i, and g_i should have binary values following (3). Although $z_{l,s}$ are treated as continuous variables, the values of $z_{l,s}$ should be 0 or 1. This is because the optimal solution of a linear program should be a vertex point satisfying $\sum_s z_{l,s} = 1$ for all l.

Consider the following proposition.

Proposition 4 *Let the optimal solution of Program 2 be $x^* = (Z^*,\ u^*, v^*)$ and $\sum_s z_{l,s} = 1$. Assume that a sequence l contains sites $s_1, s_2, \ldots, s_k$ such that*

$0 < z^*_{l,s_j} < 1$ *for* $j = 1, 2, \ldots, k$, *then,*

$$\sum\nolimits_i \theta^i_{l,s_1} = \sum\nolimits_i \theta^i_{l,s_2} = \cdots = \sum\nolimits_i \theta^i_{l,s_k} = \max\left\{\sum\nolimits_i \theta^i_{l,s}\right\}\,,$$

where θ^i_{l,s_j} *are specified in (6) .*

Proof For $\sum_s z_{l,s} = 1$, if $s_p, s_q \in \{s_1, s_2, \ldots s_k\}$ where $\sum_i \theta^i_{l,s_p} > \sum_i \theta^i_{l,s_q}$, then to maximize $\text{Score}_l = \sum_{l,j} z_{l,s_j} \sum_i \theta^i_{l,s_j}$ requires $z_{l,s_q} = 0$. This conflicts with the observation that $0 < z_{l,s_q} < 1$, therefore $\sum_i \theta^i_{l,s_1} = \sum_i \theta^i_{l,s_2} = \cdots = \sum_i \theta^i_{l,s_k}$. □

After solving Program 2 we can obtain the globally optimum solution "TGTGA******TCACA" with objective value 147. The related nonzero $z_{l,s}$ values indicate the starting positions of the binding sites in the 18 sequences, as listed below:

$$\begin{aligned} z_{1,64} &= z_{2,58} = z_{3,79} = z_{4,66} = z_{5,53} = z_{6,63} = z_{7,27} \\ &= z_{8,42} = z_{9,12} = z_{10,17} = z_{11,64} = z_{12,44} = z_{13,51} \\ &= z_{14,74} = z_{15,20} = z_{16,56} = z_{17,87} = z_{18,81} = 1 \end{aligned}$$

All other $z_{l,s}$'s have value 0.

In Program 2 the total number of 0-1 variables is $2m$ and the total number of the continuous variables is $20m + |l| * |s|$. Since the number of 0-1 variables is independent of the lengths of l and s, a motif finding problem with many long sequences can be solved effectively.

Suboptimal Consensus Sequences Program 2 can find the exact global optimum solution. Sometimes the second best and the third best solution may also be useful. It is very convenient for the proposed method to find a complete set of consensus sequences by adding some extra constraints. For instance, the second best solution of Program 2 can be obtained conveniently by solving the following program:

$$\begin{aligned} &\text{Maximize} \quad \sum_{l=1}^{18} \text{Score}_l && (16) \\ &\text{subject to} \quad \text{(i) The same constraints in Model 1} \\ &\qquad\qquad\quad \text{(ii) } t_1 + g_2 + t_3 + g_4 + a_5 + t_6 + c_7 + \\ &\qquad\qquad\qquad a_8 + c_9 + a_{10} \le 9 \ \ \text{(new constraint)} \end{aligned}$$

AAGACTGTTTTTTTGATC
GATTATTTGCACGGCGTC

a

$l = 1, s = 1$	$\boxed{\texttt{AAGAC}}\texttt{TGTTTT}\boxed{\texttt{TTTGA}}\texttt{TC}$
$l = 1, s = 2$	$\texttt{A}\boxed{\texttt{AGACT}}\texttt{GTTTTT}\boxed{\texttt{TTGAT}}\texttt{C}$
$l = 1, s = 3$	$\texttt{AA}\boxed{\texttt{GACTG}}\texttt{TTTTTT}\boxed{\texttt{TGATC}}$
$l = 2, s = 1$	$\boxed{\texttt{GATTA}}\texttt{TTTGCA}\boxed{\texttt{CGGCG}}\texttt{TC}$
$l = 2, s = 2$	$\texttt{G}\boxed{\texttt{ATTAT}}\texttt{TTGCAC}\boxed{\texttt{GGCGT}}\texttt{C}$
$l = 2, s = 3$	$\texttt{GA}\boxed{\texttt{TTATT}}\texttt{TGCACG}\boxed{\texttt{GCGTC}}$

b

$$\begin{bmatrix} \mathit{AAGACTTTGA} & \mathit{AGACTTTGAT} & \mathit{GACTGTGATC} \\ \mathit{GATTACGGCG} & \mathit{ATTATGGCGT} & \mathit{TTATTGCGTC} \end{bmatrix}$$

c

Mixed 0-1 Linear Programming Approach for DNA Transcription Element Identification, Figure 1
A small example of finding consensus sequence: a two sequences to be compared; b Schematic representation of the candidate sites; c The associated D' matrix

The new constraint is used to force the program to find a new solution different from the solution of Program 2. The found second best consensus sequence is "TTTGA******TCAAA" with score 129. Similarly we can find another solution by adding following constraint into (16).

$$t_1 + t_2 + t_3 + g_4 + a_5 + t_6 + c_7 + a_8 + a_9 + a_{10} \le 9$$

The found third best consensus sequence is "AAATT******AATTT" with score 129.

Extend to Find Unknown Binding Sites A more complicated motif finding problem is to search for the consensus sequence with an uncertain pattern format where the number of ignored letters between the two half sites is unknown. An example is to find a consensus sequence of length $2 * 5 + k$ with the pattern

$$L_1 L_2 L_3 L_4 L_5 * \cdots * L_6 L_7 L_8 L_9 L_{10}$$

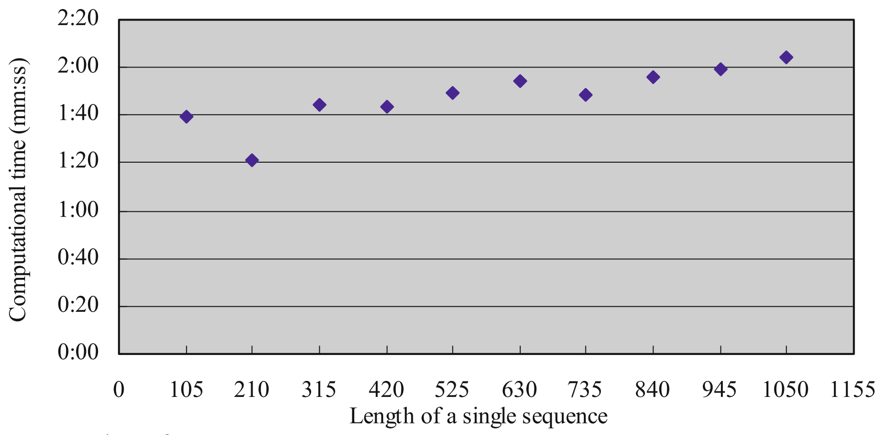
where k, the number of $*$'s, is an unknown integer between 0 and 10.

Program 2 can be modified slightly to treat this type of motif finding problem. Firstly we expand D in (1) as D' below:

$$D' = [D'(0) D'(1) D'(2) \ldots\ldots D'(10)]$$

Computational time versus sequence length

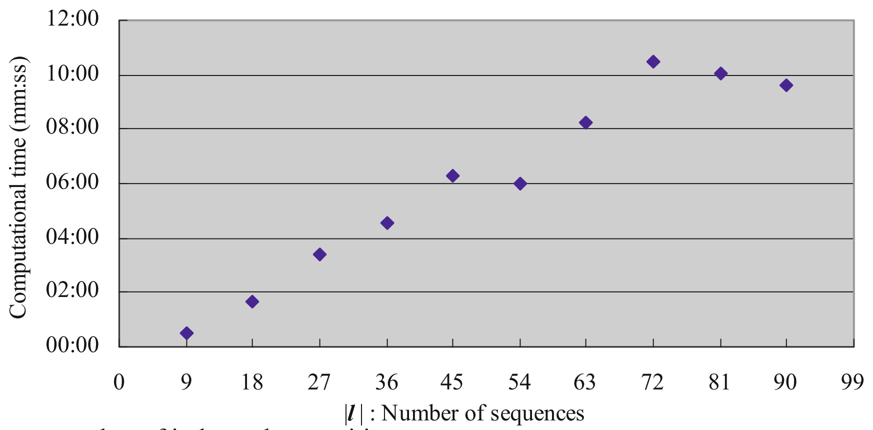
Sequence Length	Solving Time (mm:ss)
105	1:39
210	1:21
315	1:44
420	1:43
525	1:48
630	1:54
735	1:48
840	1:56
945	1:59
1050	2:04



a

Computational time versus number of sequences

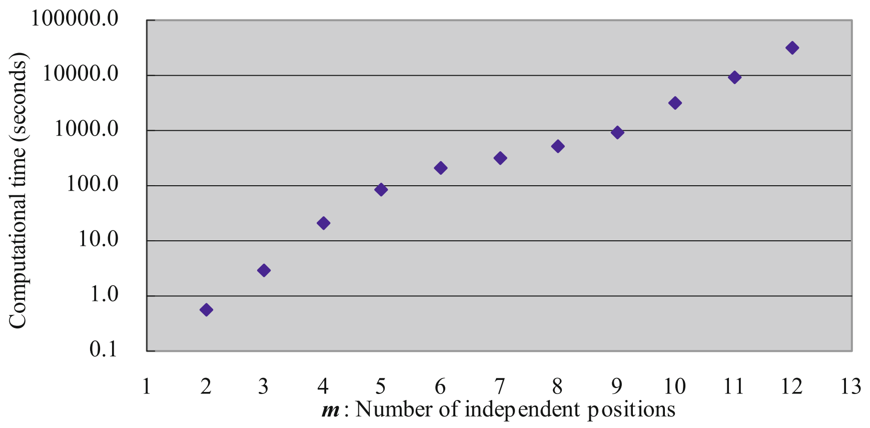
Number of Sequences	Solving Time (mm:ss)
9	0:30
18	1:39
27	3:21
36	4:32
45	6:15
54	6:01
63	8:16
72	10:29
81	10:01
90	9:37



b

Computational time versus number of independent positions

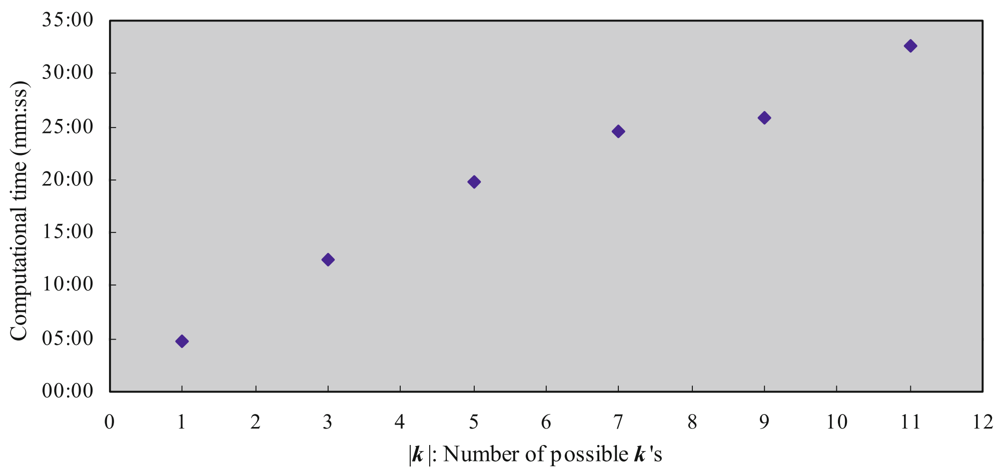
Number of Indep Pos	Solving Time (h:mm:ss)
2	0:00:01
3	0:00:03
4	0:00:21
5	0:01:23
6	0:03:38
7	0:05:18
8	0:08:25
9	0:15:52
10	0:53:27
11	2:33:20



c

Mixed 0-1 Linear Programming Approach for DNA Transcription Element Identification, Figure 2
The relationship between computational time and various factors involved in a consensus based motif finding problem. This figure illustrates the computational time of solving Program 2 with a various sequences sizes; b various number of sequences and c various independent positions

$\bar{k}$	$\lvert k\rvert$	Common Site	Score	Computational Time
0	1	`TGTTT(0)AAACA`	126	4:51
2	3	`TGAAA(2)TTTCA`	129	12:32
4	5	`GTGAA(4)TTCAC`	134	19:46
6	7	`TGTGA(6)TCACA`	147	24:28
8	9	`TGTGA(6)TCACA`	147	25:49
10	11	`TGTGA(6)TCACA`	147	32:35



Mixed 0-1 Linear Programming Approach for DNA Transcription Element Identification, Figure 3
Computational time of Program 3 with various numbers of possible k's. The number enclosed in the common site is the solution of k

in which

$$D'(k) = \begin{bmatrix} d^1_{1,1,k} \cdots d^{10}_{1,1,k} & d^1_{1,2,k} \cdots d^{10}_{1,2,k} & \cdots & d^1_{1,90,k} \cdots d^{10}_{1,90,k} \\ d^1_{2,1,k} \cdots d^{10}_{2,1,k} & d^1_{2,2,k} \cdots d^{10}_{2,2,k} & \cdots & d^1_{2,90,k} \cdots d^{10}_{2,90,k} \\ \vdots & \vdots & \ddots & \vdots \\ d^1_{18,1,k} \cdots d^{10}_{18,1,k} & d^1_{18,2,k} \cdots d^{10}_{18,2,k} & \cdots & d^1_{18,90,k} \cdots d^{10}_{18,90,k} \end{bmatrix}$$

where $k \in \{0, 1, \ldots, 10\}$.

$$d^i_{l,s,k} = \begin{cases} b_{l,i+s-1} & (\text{for } i = 1, 2, 3, 4, 5) \\ b_{l,i+s+k-1} & (\text{for } i = 6, 7, 8, 9, 10) \end{cases}$$

$$\theta^i_{l,s.k} = a_i,\ t_i,\ c_i \text{ or } g_i \text{ when } d^i_{l,s.k} = \text{'A', 'T', 'C', or 'G' respectively}.$$

The cases with k larger than 10 are not considered since they are relatively rare. A linear mixed 0-1 program for solving this example is formulated below:

Program 3

$$\text{Maximize} \quad \sum_{i=1}^{2m} \left(Z(a_i) + Z(t_i) + Z(c_i) + Z(g_i)\right) \quad (15)$$

subject to (i) $\displaystyle\sum_{k=0}^{10} \sum_{s=1}^{96-k} z_{l,s,k} = 1\,,$

$z_{l,s,k} \geq 0$ for all l, s, k

(ii) $\displaystyle\sum_s z_{1,s,k} = \sum_s z_{2,s,k} = \ldots = \sum_s z_{18,s,k}$ for $k \in \{0, 1, \ldots, 10\}$

(iii) the same conservative and logical constraints in Program 2

(iv) the same constraints for linearizing product terms in Program 2 but replace $z_{l,s}$ by $z_{l,s.k}$.

Constraints (i) and (ii) are used to ensure that when a specific k is chosen then $\sum_s z_{l,s,k} = 1$ and $\sum_s z_{l,s,k'} = 0$ for $k' \neq k$.

Mixed 0-1 Linear Programming Approach for DNA Transcription Element Identification, Table 2
FNR binding sites found by Program 3

Operon	Seq. length	Site seq. found by Program 3	Predicted Position	Score	Site seq. listed in RegulonDB*	Center Position
Common site:		**TTGAT----ATCAA**				
narK	338	ATGAT----ATCAA	-86	9	actatgGGTA$\boxed{\texttt{ATGAT}}$AA**A**T$\boxed{\texttt{ATCAA}}$TGATagataa	-79.5
		TTGAT----ATCAA	-48	10	atcttaTCGT$\boxed{\texttt{TTGAT}}$TT**A**C$\boxed{\texttt{ATCAA}}$ATTGccttta	-41.5
ansB	345	TTGTT----GTCAA	-48	8	acgttgTAAA$\boxed{\texttt{TTGTT}}$TA**A**C$\boxed{\texttt{GTCAA}}$ATTTcccata	-41.5
		TTGTA----TCCAA	-81	6	gcctctAACT$\boxed{\texttt{TTGTA}}$GA**T**C$\boxed{\texttt{TCCAA}}$AATAtattca	-74.5
		TTTAT----TTTAA	-123	7		
narG	525	TTGAT----ATCAA	-55	10	ctc$\boxed{\texttt{ttgAT}}$CGTT$\boxed{\texttt{ATCAA}}$**T**TCCCACGCTGtttcag	-41.5
dmsA	325	TTGAT----AACAA	-48	9	ct$\boxed{\texttt{ttgaT}}$ACCG$\boxed{\texttt{AACAA}}$T**A**ATTACTCCTCacttac	-33
frd	781	TTCAG----ATCCA	-37	7	AAAAATCGATC**T**CGTCAAAT$\boxed{\texttt{TTcag}}$actt$\boxed{\texttt{atcca}}$	-47
		TTAAT----TTCAG	-98	7		
nirB	262	TTGAT----ATCAA	-48	10	aaaggtGAAT$\boxed{\texttt{TTGAT}}$TT**A**C$\boxed{\texttt{ATCAA}}$TAAGcggggt	-41.5
sodA	284	TTGAT----ATTTT	-42	7	agtacgGCA$\boxed{\texttt{TTGAT}}$AAT**C**$\boxed{\texttt{ATTTT}}$CAATAtcattt	-34
fnr**	96	TTGAC----ATCAA	-7	9	atgttaAAA$\boxed{\texttt{TTGAC}}$AAA**T**$\boxed{\texttt{ATCAA}}$TTACGgcttga	1
					ccttaaCAACTTAAGGG**T**TTTCAAATAGatagac	-103.5
(cyoA)	599	CTTCT----ATCAA	-113	7	N/A	N/A
		TTGTT----TTCAC	-198	7		
(icdA)	290	ATGAC----AACAA	16	7	N/A	N/A
		TTGCT----AGCAT	73	7		
(sdhC)	708	TTGAT----AATAA	-330	8	N/A	N/A
(ulaA)	346	TCAAT----ATCAA	-278	8	N/A	N/A
		TTGGT----ATTAA	-257	8		

* For visualizing the comparison, the letters in uppercase represent the binding site listed in RegulonDB; the letter in bold face is the center of the site sequence; and the encompassed letters represent the exact binding site obtained by Program 3.
** The second site listed in RegulonDB is not contained in the sequence data, which is only 96 bases long, from GenBank.

Cases

Finding CRP Binding Sites with a Given Pattern

Several experiments are tested here, using the example in the Appendix, to analyze the effect of sequence length and number of sequences on the computational time. All examples are solved by LINGO [40], a widely used optimization software, on a personal computer with a Pentium 4 2.0G CPU. A software package named "Global Site Seer" is developed based on Program 2 for finding DNA motifs. This software is available from http://www.iim.nctu.edu.tw/~cjfu/gss.htm.

Figure 2 illustrates the experimental results for analyzing the time complexity. Figure 2a is the computational time given various sequence lengths, where the number of sequences is fixed at 18. The results show that the computational time changes only slightly even if the sequence length is increased from 105 to 1050. Figure 2b is the computational time with various numbers of sequences. It shows that the solving time is roughly proportional to the number of sequences. The proposed model is quite promising for finding DNA motifs in a dataset with a large sequence length and a large number of sequences. Figure 2c shows that the computational time rises exponentially as the number of independent positions increases.

Using Program 3 to search CRP binding sites, we obtain the globally optimal solution "TGTGA******TCACA" with score 147, which is exactly the same solution found in Program 2. The second best solution is "GTGAA****TTCAC" with score 134. The relationship between the computational time and the number of possible k's (i. e. $|k|$) is linear, as shown in the experiment result listed in Fig. 3. The number of ignored letters k is between 0 and $\bar{k}$, the upper bound of k, and thus we have $|k| = \bar{k} + 1$ in this experiment.

Finding FNR-binding Sites with an Ambiguous Shared Pattern Program 3 is also applied to solve an example of searching for binding sites of fumarate and nitrate reduction regulatory protein (FNR) in *E. coli*. Both CRP and FNR belong to the CRP/FNR helix-turn-helix transcription factor superfamily [47]. The sequence data, which is taken from GenBank, contains 12 DNA sequences with lengths varied from 96 to 781. Owing to the dimer structure of the binding protein, the consensus sequence in this example also has a constraint of inverse symmetry. The RegulonDB database [18] lists the found regulatory binding sites for eight of these twelve sequences while the exact positions of the other four sequences are not listed yet. Solving this example by Program 3 we obtained the global optimal consensus sequence as "TTGAT****ATCAA" with score 107, which is the same consensus sequence as indicated by [47]. Table 2 illustrates the result including the consensus sequence and the predicted binding sites for all of the 12 sequences. Some sites downstream of the transcription start (i. e. with positive indices) are also listed because there are a few known cases in which regulatory sites appear within transcription units [47]. The proposed method has found some sites not listed in RegulonDB, but which have scores higher than those listed in RegulonDB (e. g. the third solution in the Operon *ansB* row of Table 2). The best predicted sites in the four undetermined sequences are also listed in Table 2.

Conclusions

This study proposes a linear mixed 0-1 programming approach for finding DNA motifs. Compared to the widely used maximum likelihood methods, the proposed method can reach a global optimum rather than finding a local optimum or a feasible solution. Additionally, by utilizing binary variables, some logical constraints can be embedded into the models. It is also convenient to find the complete set of the second, third, etc. best consensus sequences. Since the number of binary variables is fully independent of the number of sequences and the length of a sequence, the proposed method can treat motif finding problems with many long sequences. For finding motifs with many independent positions in an acceptable time, this study also proposes a method for distributed computing.

The proposed method can also be conveniently extended to treat more complicated motif finding problems. In this study an extension of the linear program is designed to find DNA motifs with an unknown number of ignored letters between the two half sites. The result of searching for FNR-binding sites shows that the extended model can find not only the locations of known binding sites listed in the RegulonDB database but also those not yet delimitated.

References

1. Ao W, Gaudet J, Kent WJ, Muttumu S, Mango SE (2004) Environmentally induced foregut remodeling by PHA-4/FoxA and DAF-12/NHR. Science 305(5691):1743–1746
2. Bailey T, Elkan C (1995) Unsupervised learning of multiple motifs in biopolymers using expectation maximization. Mach Learn 21(1–2):51–80
3. Bailey T, Elkan C (1995) The value of prior knowledge in discovering motifs with MEME. In: Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology, AAAI Press, Menlo Park, pp 21–29
4. Blanchette M, Schwikowski B, Tompa M (2000) An exact algorithm to identify motifs in orthologous sequences from multiple species. In: Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, San Diego, pp 37–45
5. Brāzma A, Jonassen I, Eidhammer I, Gilbert D (1998) Approaches to the automatic discovery of patterns in biosequences. J Comput Biol 5(2):279–305
6. Buhler J, Tompa M (2002) Finding Motifs Using Random Projections. J Comput Biol 9(2):225–242
7. Califano A (2000) SPLASH: structural pattern localization analysis by sequential histograms. Bioinformatics 16(4):341–357
8. DeRisi J, Iyer V, Brown P (1997) Exploring the metabolic and genetic control of gene expression on a genomic scale. Science 278(5338):680–686
9. Ecker JG, Kupferschmid M, Lawrence CE, Reilly AA, Scott ACH (2002) An application of nonlinear optimization in molecular biology. Eur J Oper Res 138(2):452–458
10. Eskin E, Pevzner P (2002) Finding composite regulatory patterns in DNA sequences. Bioinformatics (Supplement 1) 18(1):S354–S363
11. Favorov AV, Gelfand MS, Gerasimova AV, Mironov AA, Makeev VJ (2004) Gibbs sampler for identification of symmetrically structured, spaced DNA motifs with improved estimation of the signal length and its validation on the ArcA binding sites. In: Proceedings of BGRS 2004, BGRS, Novosibirsk
12. Fratkin E, Naughton BT, Brutlag DL, Batzoglou S (2006) MotifCut: Regulatory motifs finding with maximum density subgraphs. Bioinformatics 22(14):e150–157

13. Frith MC, Hansen U, Spouge JL, Weng Z (2004) Finding functional sequence elements by multiple local alignment. Nucl Acids Res 32(1):189–200
14. Galas D, Eggert M, Waterman M (1985) Rigorous pattern-recognition methods for DNA sequences: analysis of promoter sequences from Escherichia coli. J Mol Biol 186(1):117–128
15. Gelfand M, Koonin E, Mironov A (2000) Prediction of transcription regulatory sites in archaea by a comparative genomic approach. Nucl Acids Res 28(3):695–705
16. Hertz GZ, Hartzell GW, Stormo GD (1990) Identification of consensus patterns in unaligned DNA sequences known to be functionally related. Comput Appl Biosci 6(2):81–92
17. Hertz GZ, Stormo GD (1999) Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. Bioinformatics 15(7–8):563–577
18. Huerta AM, Salgado H, Thieffry D, Collado-Vides J (1998) RegulonDB: a database on transcriptional regulation in Escherichia coli. Nucl Acids Res 26(1):55–59
19. Hughes JD, Estep PW, Tavazoie S, Church GM (2000) Computational identification of cis-regulatory elements associated with functionally coherent groups of genes in Saccharomyces cerevisiae. J Mol Biol 296(5):1205–1214
20. Krause M, Park M, Zhang JM, Yuan J, Harfe B, Xu SQ, Greenwald I, Cole M, Paterson B, Fire A (1997) A C. elegans E/Daughterless bHLH protein marks neuronal but not striated muscle development. Development 124(11):2179–2189
21. Lawrence CE, Altschul S, Boguski M, Liu J, Neuwald A, Wootton J (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. Science 262(5131):208–214
22. Lawrence CE, Reilly AA (1990) An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. PROTEINS: Struct, Funct Genet 7(1):41–51
23. Liang S, Samanta M, Biegel B (2004) cWINNOWER algorithm for finding fuzzy DNA motifs. J Bioinform Comput Biol 2(1):47–60
24. Liu XS, Brutlag DL, Liu JS (2001) BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. Pacific Symposium on Biocomputing, pp 127–138
25. Li HL, Fu CJ (2005) A linear programming approach for identifying a consensus sequence on DNA sequences. Bioinformatics 21(9):1838–1845
26. Li M, Ma B, Wang L (1999) Finding similar regions in many strings. In: Proceedings of the 31st ACM Annual Symposium on Theory of Computing, pp 473–482
27. Lockhart DJ, Dong H, Byme MC, Follettie MT, Gallo MV, Chee MS, Mittmann M, Wang C, Kobayashi M, Horton H, Brown EL (1996) Expression monitoring by hybridization to high-density oligonucleotide arrays. Nat Biotechnol 14(13):1675–1680
28. Matys V, Fricke E, Geffers R, Gößling E, Haubrock M, Hehl R, Hornischer K, Karas D, Kel AE, Kel-Margoulis OV, Kloos DU, Land S, Lewicki-Potapov B, Michael H, Münch R, Reuter I, Rotert S, Saxel H, Scheer M, Thiele S, Wingender E (2003) TRANSFAC: transcriptional regulation, from patterns to profiles. Nucl Acids Res 31(1):374–378
29. Needleman S, Wunsch C (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48(3):443–453
30. Pavesi G, Mauri G, Pesole G (2001) An algorithm for finding signals of unknown length in DNA sequences. ISMB 2001 Bioinformatics 17(Suppl 1):S207–214
31. Pavesi G, Mereghetti P, Mauri G, Pesole G (2004) Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. Nucl Acids Res 32(Web Server Issue):W199–W203
32. Peng CH, Hsu JT, Chung YS, Lin YJ, Chow WY, Hsu DF, Tang CY (2006) Identification of Degenerate Motifs Using Position Restricted Selection and Hybrid Ranking Combination. Nucl Acids Res 34:6379–6391
33. Pesole G, Prunella N, Liuni S, Attimonelli M, Saccone C (1992) WORDUP: an efficient algorithm for discovering statistically significant patterns in DNA sequences. Nucl Acids Res 20(11):2871–2875
34. Pevzner P, Sze H (2000) Combinatorial approaches to finding subtle signals in DNA sequences. In: Proceedings International Conference on Intelligent Systems for Molecular Biology, La Jolla, 20–23 August 2000, pp 269–278
35. Portman DS, Emmons SW (2000) The basic helix-loop-helix transcription factors LIN-32 and HLH-2 function together in multiple steps of a C. elegans neuronal sublineage. Development 127(24):5415–5426
36. Rajasekaran S, Hu Y, Luo J, Nick H, Sahni S, Shaw S (2001) Efficient algorithms for similarity search. J Comb Optim 5(1):125–132
37. Rajasekaran S, Sahni S, Shaw S (2001) Efficient algorithms for local alignment search. J Comb Optim 5(1):117–124
38. Régnier M, Denise A (2004) Rare events and conditional events on random strings. Discret Math Theor Comput Sci 6(2):191–214
39. Schneider TD, Stormo GD, Gold L, Ehrenfeucht A (1986) Information content of binding sites on nucleotide sequences. J Mol Biol 188(3):415–431
40. Schrage L (1999) Optimization Modeling With Lingo. LINDO Systems Inc., Chicago
41. Sinha S, Tompa M (2003) Performance comparison of algorithms for finding transcription factor binding sites. In: Bourbakis NG (ed) 3rd IEEE Symposium on Bioinformatics and Bioengineering, IEEE Computer Society, New York, 2003, pp 214–220
42. Sinha S, Tompa M (2003) YMF: a program for discovery of novel transcription factor binding sites by statistical overrepresentation. Nucl Acids Res 31(13):3586–3588
43. Stein L, Sternberg P, Durbin R, Thierry-Mieg J, Spieth J (2001) WormBase: network access to the genome and bi-

ology of Caenorhabditis elegans. Nucl Acids Res 29(1): 82–86
44. Stormo GD, Hartzell GW (1989) Identifying protein-binding sites from unaligned DNA fragments. In: Proceedings of the National Academy of Sciences of the USA, 86(4), pp 1183–1187
45. Stormo GD (2000) DNA binding sites: representation and discovery. Bioinformatics 16(1):16–23
46. Swoboda P, Adler HT, Thomas JH (2000) The RFX-type transcription factor DAF-19 regulates sensory neuron cilium formation in C. elegans. Mol Cell 5(3):411–421
47. Tan K, Moreno-Hagelsieb G, Collado-Vides J, Stormo GD (2001) A comparative genomics approach to prediction of new members of regulons. Genome Res 11(4):566–584
48. Thijs G, Lescot M, Marchal K, Rombauts S, De Moor B, Rouzé P, Moreau Y (2001) A higher-order background model improves the detection of promoter regulatory elements by Gibbs sampling. Bioinformatics 17(12):1113–1122
49. Tompa M (1999) An exact method for finding short motifs in sequences with application to the Ribosome Binding Site problem. In: Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology, Heidelberg, 6–August 2000, pp 262–271
50. Tompa M, Li N, Bailey TL, Church GM, De Moor B, Eskin E, Favorov AV, Frith MC, Fu Y, Kent WJ, Makeev VJ, Mironov AA, Noble WS, Pavesi G, Pesole G, Regnier M, Simonis N, Sinha S, Thijs G, van Helden J, Vandenbogaert M, Weng Z, Workman C, Ye C, Zhu Z (2005) Assessing Computational Tools for the Discovery of Transcription Factor Binding Sites. Nat Biotechnol 23(1):137–144
51. Tsai HK, Huang GTW, Chou MY, Lu HHS, Li WH (2006) Method for identifying transcription factor binding sites in yeast. Bioinformatics 22(14):1675–1681
52. Velculescu VE, Zhang L, Vogelstein B, Kinzler KW (1995) Serial analysis of gene expression. Science 270(5235): 484–487
53. Waterman M, Galas D, Arratia R (1984) Pattern recognition in several sequences: consensus and alignment. Bull Math Biol 46(4):512–527
54. Workman CT, Stormo GD (2000) ANN-Spec: a method for discovering transcription factor binding sites with improved specificity. In: Altman R, Dunker AK, Hunter L, Klein TE (eds) Pacific Symposium on Biocomputing. Stanford University, Stanford, pp 467–478

Mixed Integer Classification Problems

Paul A. Rubin
The Eli Broad Graduate School of Management, Michigan State University, East Lansing, USA

MSC2000: 62H30,68T10,90C11

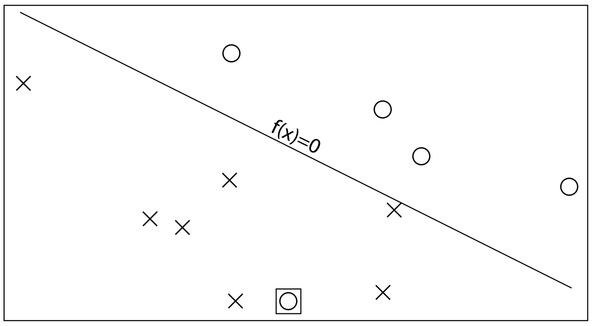
Article Outline

Keywords and Phrases

Classification; Discriminant analysis; Integer programming

Introduction

The *G-group classification problem*, also known as the *G-group discriminant problem*, involves a population partitioned into G distinct (and predefined) groups. The object is to construct a scalar- or vector- valued *scoring function* $f : \Re^p \to \Re$ so that the group to which a population member with observed attributes $\mathbf{x} \in \Re^p$ belongs can be determined, with best possible accuracy, from its score $f(\mathbf{x})$. The scoring function f is usually restricted to a particular class (most commonly, linear). By a wide margin, the majority of studies have focused on the two-group case. Construction of f is based on *training samples* from the various groups. The most reasonable criterion for choosing f may be expected misclassification cost, but many studies make the simplifying assumptions that all misclassifications are equally expensive and that groups are represented in the training samples in proportion to their prior probability of being encountered, in which case the criterion reduces



Mixed Integer Classification Problems, Figure 1
Optimal linear classifier (□ = misclassified)

to minimizing the number of misclassifications in the combined training samples. Figure 1 illustrates an optimal choice of linear classifier f in a two-group problem.

Classical discriminant analysis relies on distributional assumptions. In the two-group case with normally distributed attributes, the scalar-valued discriminant function that minimizes expected misclassification cost is known to be linear if the two groups have identical covariance structures and quadratic if not. In both cases, direct estimation of f is straightforward. Nonparametric approaches, making no distributional assumptions, have utilized an eclectic assortment of techniques, among them neural networks, metaheuristics, and mathematical programming. Although some consideration has been given to nonlinear programming methods, the bulk of the work involving mathematical programming has utilized either linear or mixed integer linear programming models, or support vector machines (quadratic programs) [8]. See [10] and ► linear programming models for classification (elsewhere in this volume) for an overview of the subject.

When f is linear, the problem of minimizing the number of misclassifications is a special case of the slightly more general problem of dropping the smallest (or least costly) set of constraints necessary to render an inconsistent set of linear inequalities consistent. This problem crops up in a variety of contexts, including pattern recognition [18], machine learning/data mining [5] and the analysis of infeasible linear programs [6]. Thus methods from those areas may be applicable to discriminant problems. For instance, Soltysik and Yarnold [15] applied the algorithm of Warmack and Gonzalez [18] to the two-group linear discriminant problem.

Formulation

The following is a typical mixed integer programming model for the two-group case, using a scalar linear discriminant function:

$$\begin{aligned} \min \quad & \sum_{g=1}^{2} \frac{\pi_g C_g}{N_g} \sum_{n=1}^{N_g} z_{gn} \\ \text{s.t.} \quad & \mathbf{X}_1\mathbf{w} + w_0 \cdot \mathbf{1} - M \cdot \mathbf{z}_1 \leq \mathbf{0} \\ & \mathbf{X}_2\mathbf{w} + w_0 \cdot \mathbf{1} + M \cdot \mathbf{z}_2 \geq \mathbf{0} \\ & \mathbf{w}, w_0 \text{ free}; \mathbf{z}_g \in \{0,1\}^{N_g} . \end{aligned} \tag{1}$$

Matrix $\mathbf{X}_g$ is an $N_g \times p$ training sample from group g, while π_g and C_g are respectively the prior probability of group g and the cost of misclassifying a member of that group. M is a sufficiently large positive constant, and $\mathbf{0}$ and $\mathbf{1}$ denote vectors, all of whose entries are respectively 0 or 1. The discriminant function $f(\mathbf{x}) = \mathbf{w}'\mathbf{x} + w_0$ is intended to produce negative scores for members of the first group and positive scores for members of the second group. Bivalent indicator variable z_{gn} takes value 1 if the nth training observation from group g is classified incorrectly and 0 if it is classified correctly.

The discriminant function is linear as written, but various nonlinear functions can be generated by embedding the attribute space $\Re^p$ in a higher-dimensional space. Support vector machines are particularly adept at this. Polynomial functions, for instance, are easily accommodated in (1) by expanding the sample matrices to include powers and products of attributes.

A score of zero results in an ambiguous classification. Some authors deal with this by changing the first two constraints of (1) to

$$\begin{aligned} \mathbf{X}_1\mathbf{w} + w_0 \cdot \mathbf{1} - M \cdot \mathbf{z}_1 &\leq -\varepsilon \cdot \mathbf{1} \\ \mathbf{X}_2\mathbf{w} + w_0 \cdot \mathbf{1} + M \cdot \mathbf{z}_2 &\geq +\varepsilon \cdot \mathbf{1} , \end{aligned}$$

where ε is a small positive constant. This formulation is nearly as general, although it is mathematically possible that infelicitous choices of ε and M could rule out an otherwise desirable solution.

Problem (1) is known to be *NP*-hard [1]. At the same time, using the finite VC-dimension of linear classifiers [16,17], it can be shown that the error rate of the solution to (1) converges in probability to the optimal error rate as sample size grows [2]. Assuming availability of sufficient data, a key question is whether the problem remains tractable when the training sample is large enough to provide a suitably accurate solution. There is grounds for (cautious) optimism, in that progress in hardware, software and algorithms advances the boundaries of what is tractable, while for a given problem instance the sample size needed for accuracy is static.

While there will often be a unique best choice of training observations to misclassify (i. e., unique optimal values of $\mathbf{z}_1$ and $\mathbf{z}_2$), there commonly will be infinitely many choices for the coefficients $\mathbf{w}$, w_0 of a discriminant function that misclassifies those observations only. To select from among those coefficient solutions, authors often introduce additional terms in the objective function. As an example, Bajgier and Hill [4] used a formulation similar to the following:

$$\begin{aligned}\min \quad & \sum_{g=1}^{2} \frac{\pi_g}{N_g} \sum_{n=1}^{N_g} \left[C_g z_{gn} + \varepsilon_1 d_{gn}^{-} - \varepsilon_2 d_{gn}^{+} \right] \\ \text{s.t.} \quad & \mathbf{X}_1\mathbf{w} + w_0 \cdot \mathbf{1} + \mathbf{d}_1^{+} - \mathbf{d}_1^{-} - M \cdot \mathbf{z}_1 \le \mathbf{0} \\ & \mathbf{X}_2\mathbf{w} + w_0 \cdot \mathbf{1} - \mathbf{d}_2^{+} + \mathbf{d}_2^{-} + M \cdot \mathbf{z}_2 \ge \mathbf{0} \\ & \mathbf{w}, w_0 \text{ free}; \quad \mathbf{d}_g^{+}, \mathbf{d}_g^{-} \ge \mathbf{0}; \quad \mathbf{z}_g \in \{0,1\}^{N_g} .\end{aligned}$$

The deviation variables $\mathbf{d}_g^{+}$ and $\mathbf{d}_g^{-}$ measure the amount by which each score falls on the correct and incorrect side of the zero cutoff, respectively. The objective functions rewards the former and penalizes the latter, using small positive objective coefficients ε_1 and ε_2 to prevent improvements in these terms from inducing unnecessary misclassifications.

The motivation for formulation (1) is simple: if the training samples are representative of the overall population, the discriminant function that minimizes misclassification costs on the training samples should come close to minimizing expected misclassification cost on the overall population. Models like (1) tend to be computationally expensive, however. As is typical with mixed integer programming models, computation time increases modestly with the number of attributes (p) but more dramatically with the number of zero-one variables ($N_1 + N_2$, the combined sample size). Moreover, the constant M must be chosen large enough that the best choice of $\boldsymbol{w}$ and w_0 is not rendered infeasible by a misclassified observation with score larger than M in magnitude; but the larger M is, the weaker the bounds in a branch-and-bound solution of the problem, and thus the longer the solution time. Codato and Fischetti [7] reported success using a form of Benders cut to eliminate M.

In the special case where all attribute variables are discrete, it is likely that some observation vectors will appear more than once in the training samples. When that occurs, the number of zero-one variables can be reduced from one per observation to one per *distinguishable* observation, yielding a variation of (1) in which the objective function is replaced with

$$\min \quad \sum\nolimits_{g=1}^{2} \frac{\pi_g C_g}{N_g} \sum\nolimits_{k=1}^{K_g} N_{gk} z_{gk} .$$

In this formulation [3], K_g is the number of distinct attribute vectors $\mathbf{x}$ in the training sample from group g, N_{gk} is the number of repetitions of the kth distinct observation from group g, and the matrices $\mathbf{X}_g$ contain only one copy of each such observation.

Multiple Groups

When $G > 2$ groups are involved, the problem becomes considerably more complicated. In a practical application with multiple groups, it is plausible that misclassification costs would depend not only on the group to which a misclassified point belonged but also the one into which it was classified. Thus an appropriate objective function might look like

$$\sum_{g=1}^{G} \frac{\pi_g}{N_g} \sum_{\substack{h=1 \\ h \ne g}}^{G} C_{gh} \sum_{n=1}^{N_g} z_{ghn} ,$$

where C_{gh} is the cost of classifying a point from group g into group h and z_{ghn} is 1 if the nth observation of group g is classified into group h and 0 otherwise. This represents a substantial escalation of the number of indicator variables. As a consequence, most research on the multiple group problem assumes that misclassification costs depend only on the correct group.

Few models, and fewer computational results, have been published for the multiple group problem. Gehrlein [9] presented one of the earliest scalar-valued mixed integer models for the case $G > 2$. The range of his discriminant function is partitioned into separate intervals corresponding to the groups. His model, adapted to the preceding notation, is

$$\begin{aligned}\min \quad & \sum_{g=1}^{G} \frac{\pi_g C_g}{N_g} \sum_{n=1}^{N_g} z_{gn} \\ \text{s.t.} \quad & \mathbf{X}_g\mathbf{w} + w_0 \cdot \mathbf{1} - M \cdot \mathbf{z}_g - U_g \cdot \mathbf{1} \le \mathbf{0} \\ & \mathbf{X}_g\mathbf{w} + w_0 \cdot \mathbf{1} + M \cdot \mathbf{z}_g - L_g \cdot \mathbf{1} \ge \mathbf{0} \\ & U_g - L_g \ge 0 \\ & L_h - U_g + M y_{hg} \ge \varepsilon \\ & y_{gh} + y_{hg} = 1 \\ & \mathbf{w}, w_0, \mathbf{L}, \mathbf{U} \text{ free}; \\ & \mathbf{z}_g \in \{0,1\}^{N_g}; \quad \mathbf{y} \in \{0,1\}^{G(G-1)} .\end{aligned}$$

The first three constraints are repeated for $g = 1, \ldots, G$ while the next two are repeated for all pairs g,

$h = 1, \ldots, G$ such that $g \neq h$. Observations are classified into group g if their scores fall in the interval $[L_g, U_g]$. Variable $y_{gh} = 1$ if the scoring interval for group g precedes that for group h. Parameter $\varepsilon > 0$ dictates a minimum separation between intervals.

Using a single scalar-valued discriminant function with $G > 2$ groups is restrictive; it assumes that the groups project onto some line in an orderly manner. In [9], Gehrlein also suggested a model using a vector-valued discriminant function $\boldsymbol{f}()$ of dimension G. Observation $\mathbf{x}$ would be classified into the group corresponding to the largest component of $\mathbf{f}(\mathbf{x})$. The model increases the number of coefficient variables and the number of constraints but not the number of 0-1 variables, the primary determinant of execution time. The model is:

$$\begin{aligned} \min \quad & \sum_{g=1}^{G} \frac{\pi_g C_g}{N_g} \sum_{n=1}^{N_g} z_{gn} \\ \text{s.t.} \quad & \mathbf{X}_g \mathbf{w}_g + w_{g0} \cdot \mathbf{1} - \mathbf{X}_g \mathbf{w}_h - w_{h0} \cdot \mathbf{1} + M \cdot \mathbf{z}_g \\ & \quad \geq \ \varepsilon \cdot \mathbf{1} \\ & \mathbf{w}_g, w_{g0} \text{ free}; \ \ \mathbf{z}_g \in \{0,1\}^{N_g} . \end{aligned}$$

Here $\mathbf{w}'_g \mathbf{x} + w_{g0}$ is the gth component of $\mathbf{f}(\mathbf{x})$ and $\varepsilon > 0$ is the minimum acceptable difference between the correct component of the scoring function and the largest incorrect component. The sole constraint is repeated once for each pair $g, h = 1, \ldots, G$ such that $g \neq h$.

Methods

Advances in computer hardware, optimization software and algorithms for the mixed integer classification problem have allowed progressively larger training samples to be employed: where Koehler and Erenguc [11] were restricted to combined training samples of 100 in 1990 (on a mainframe), Rubin [13] was able to handle over 600 observations in 1997 (on a personal computer). Nonetheless, a variety of heuristics have been developed to find near optimal solutions to the problem. Several revolve around this property of the problem: if the training samples can be classified with perfect accuracy by a linear function, then problem (1) can be solved as a linear program, with the z_{gn} deleted, to obtain a discriminant function. Deletion of the z_{gn} reduces the objective function to a constant 0. Although this is perfectly acceptable, heuristics may substitute an objective function from one of the linear programming classification models, to encourage the chosen discriminant function to separate scores of the two groups as much as possible. This often also necessitates inclusion of a normalization constraint, to keep the resulting linear program from being unbounded. Alternatively, (1) may be solved heuristically to determine which training observations to misclassify, and then a linear programming model using the remaining observations may be employed to select the final discriminant function.

The BPMM heuristic of [11] solves the linear program dual to a relaxation of the mixed integer problem, notes which observations would be misclassified by the resulting discriminant function, and then solves the dual of each linear relaxation obtainable by deleting one of those observations. Solving the dual problem tends to be more efficient than solving the primal, since there will typically be more observations than attributes ($N_1 + N_2 \gg p$). The heuristics presented in [14] also operate on the dual of the linear relaxation of the mixed integer problem, restricting basis entry to force certain dual variables to take value zero (equivalent to relaxing the corresponding primal constraints, thus allowing the associated observations to be misclassified).

As noted earlier, comparatively few computational studies involve mixed integer models for multiple groups. Pavur proposed a sequential mixed integer method to handle multiple groups [12], constructing a vector-valued scoring function from a sequence of scalar functions. An initial mixed integer model similar to Gehrlein's is solved to obtain the first scalar function. Thereafter, a sequence of similar mixed integer models is solved, with each model bearing additional constraints compelling the scores produced by the next scoring function to have sample covariance zero with the scores of each of the preceding functions. The covariance constraints impose a sort of probabilistic "orthogonality" on the dimensions of the composite (vector-valued) scoring function.

See also

- ▶ Deterministic and Probabilistic Optimization Models for Data Classification
- ▶ Integer Programming

- ► Linear Programming Models for Classification
- ► Optimization in Boolean Classification Problems
- ► Statistical Classification: Optimization Approaches

References

1. Amaldi E, Kann V (1995) The complexity and approximability of finding maximum feasible subsystems of linear relations. Theor Comput Sci 147:181–210
2. Asparouhov O, Rubin PA (2004) Oscillation heuristics for the two-group classification problem. J Classif 21:255–277
3. Asparoukhov OK, Stam A (1997) Mathematical programming formulations for two-group classification with binary variables. Ann Oper Res 74:89–112
4. Bajgier SM, Hill AV (1982) An experimental comparison of statistical and linear programming approaches to the discriminant problem. Decis Sci 13:604–618
5. Bennett KP, Bredensteiner EJ (1997) A parameteric optimization method for machine learning. INFORMS J Comput 9(3):311–318
6. Chinneck JW (2001) Fast heuristics for the maximum feasible subsystem problem. INFORMS J Comput 13(3):210–223
7. Codato G, Fischetti M (2006) Combinatorial Benders' cuts for mixed-integer linear programming. Oper Res 54(4): 756–766
8. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
9. Gehrlein WV (1986) General mathematical programming formulations for the statistical classification problem. Oper Res Lett 5:299–304
10. Hand DJ (1997) Construction and assessment of classification rules. Wiley, Chichester
11. Koehler GJ, Erenguc SS (1990) Minimizing misclassifications in linear discriminant analysis. Decis Sci 21:63–85
12. Pavur R (1997) Dimensionality representation of linear discriminant function space for the multiple-group problem: An MIP approach. Ann Oper Res 74:37–50
13. Rubin PA (1990) Heuristic solution procedures for a mixed-integer programming discriminant model. Manag Decis Econ 11:255–266
14. Rubin PA (1997) Solving mixed integer classification problems by decomposition. Ann Oper Res 74:51–64
15. Soltysik R, Yarnold P (1994) The Warmack–Gonzalez algorithm for linear two-category multivariable optimal discriminant analysis. Comput Oper Res 21:735–745
16. Vapnik V (1999) An overview of statistical learning theory. IEEE Trans Neural Netw 10:988–999
17. Vapnik V, Chervonenkis A (1971) On the uniform convergence of relative frequencies of events to their probabilities. Theor Probab Appl 16:264–280
18. Warmack R, Gonzalez R (1973) An algorithm for the optimal solution of linear inequalities and its application to pattern recognition. IEEE Trans Comput C22:1065–1075

Mixed Integer Linear Programming: Heat Exchanger Network Synthesis

KEMAL SAHIN, KORHAN GÜRSOY, AMY CIRIC
Department Chemical Engineering,
University Cincinnati, Cincinnati, USA

MSC2000: 90C90

Article Outline

Keywords

MILP; HEN synthesis; Transshipment model

Heat exchanger networks use the waste heat released by hot process streams to heat the cold process streams of a chemical manufacturing plant, reducing utility costs by as much as 80%. *Heat exchanger network synthesis* has been an active area of process research ever since the energy crisis of the 1970s, and over 400 research papers have been published in the area. See [1,2,4,5,6], for recent reviews.

In 1979, T. Umeda et al. [8] discovered a thermodynamic pinch point that limits the energy savings of a heat exchanger network, establishes minimum utility levels, and partitions the heat exchanger network into two independent subnetworks. This discovery revolutionized heat exchanger network synthesis: with it, designers could compute utility levels a priori, then seek the heat exchanger network structure that uses the minimum utility consumption while also minimizing the total investment cost. This remaining problem requires matching the hot utilities and process streams that release heat with the cold process streams and utilities that require heat, choosing the network structure of each stream, and designing the individual heat exchanger networks. In general, this is a *mixed integer nonlinear programming problem* (MINLP), but can be decomposed into two smaller problems by first selecting the matches between hot and cold process streams and utilities by minimizing the total number of units, then optimizing the network structure. The first problem is a mixed integer linear programming problem that will be discussed in detail here.

Using MILP Models to Find the Minimum Number of Units

Stated formally, the *minimum-units problem* is:
Given

1) A set of hot process streams and utilities $i \in H$, and for each hot stream *i*:
   a) the inlet and outlet temperatures $\mathbf{T}_i^I$ and $\mathbf{T}_i^O$;
   b) either the heat capacity flow rate $\mathbf{FC_{Pi}}$ or the heat duty $\mathbf{Q_i}$.
2) A set of cold process streams $j \in C$, and for each cold stream *j*:
   a) the inlet and outlet temperatures $\mathbf{T}_j^I$ and $\mathbf{T}_j^O$;
   b) either the heat capacity flow rate $\mathbf{FC}_{pj}$ or the heat duty $\mathbf{Q}_j$,
3) The minimum temperature difference between hot and cold streams exchanging heat, $\Delta\ T_{\min}$.

Identify a set of stream matches (*ij*) and their heat duties Q_{ij} that

a) meets the heating and cooling needs of each stream; and
b) minimizes the total number of matches.

S.A. Papoulias and I.E. Grossmann [7] formulated this as a mixed integer programming problem using a *transshipment model*, by making an analogy between heat exchanger networks and transportation networks. In the transshipment analogy, hot process streams, the sources of heat, are similar to manufacturing plants, the sources of goods, while cold process streams, the heat sinks, are akin to stores and shopping malls, the sinks of manufactured goods.

The analogy is not perfect, as heat only flows from a high temperature to a lower one, in obedience to the second law of thermodynamics. Partitioning the temperature range of the heat exchanger network into intervals can capture this heat flow pattern. Each interval sends excess, or residual, heat to the interval below it, just as excess manufactured goods are sent to a discount warehouse.

The hot side of this *temperature cascade* is created by ordering $\mathbf{T}_i^I$ and $\mathbf{T}_j^I + \Delta\ T_{\min}$ from the highest to the lowest value, creating $t = 1, \ldots, TI$ temperature intervals. Temperatures on the cold side of the cascade equal the temperature on the hot side minus $\Delta T_{\min}$. Hot stream *i* releases $\mathbf{Q}_{it}^H$ units of heat to temperature interval *t*. $\mathbf{Q}_{it}^H$ is equal to

Mixed Integer Linear Programming: Heat Exchanger Network Synthesis, Table 1
Stream data. $Q_{CW} = 8395.2\ kW$, $\Delta T_{\min} = 10°C$

Stream	$T^{\text{in}}(°)$	$T^{\text{out}}(°)$	$FC_p(kW/K)$
H1	159	77	228.5
H2	159	88	20.4
H3	159	90	53.8
C1	26	127	93.3
C2	118	149	196.1

$$Q_{it}^H = \begin{cases} FCP_i(T_{t-1} - T_t) \\ \quad \text{if } T_i^I \geq T_{t-1} \text{ and } T_i^O \leq T_t, \\ FCP_i(T_{t-1} - T_i^O) \\ \quad \text{if } T_i^I \geq T_{t-1} \text{ and } T_i^O \geq T_t, \\ Q \\ \quad \text{if } T_i^I = T_i^O \text{ and } T_{t-1} = T_i^I. \end{cases}$$

Cold stream *j* absorbs $\mathbf{Q}_{jt}^C$ units of heat from temperature interval *t*. $\mathbf{Q}_{jt}^C$ equals

$$Q_{jt}^C = \begin{cases} FCP_j(T_{t-1} - T_t) \\ \quad \text{if } T_j^I \leq T_t - \Delta T_{\min} \text{ and} \\ \qquad T_j^O \geq T_{t-1} - \Delta T_{\min}, \\ FCP_j(T_j^0 - T_{t-1}) \\ \quad \text{if } T_j^I \leq T_t - \Delta T_{\min} \text{ and} \\ \qquad T_j^O \leq T_{t-1} - \Delta T_{\min}, \\ Q_j \\ \quad \text{if } T_j^I = T_j^O \text{ and } T_j^I = T_{t-1} - \Delta T_{\min}. \end{cases}$$

Any excess heat sent to interval *t* from hot stream *i* cascades down to interval *t*+1 through the residual flow R_{it}. Process utilities may be treated as process streams, or may be placed at the top or bottom of the cascade.

This transshipment model of heat flow leads to the following mixed integer linear programming problem:

$$\min \sum_{i,j} y_{ij}$$

subject to

$$R_{i,t} - R_{i,t-1} + \sum_j q_{ijt} = Q^H_{it}, \quad i = 1, \dots, H, \quad t = 1, \dots, TI, \tag{1}$$

$$\sum_i q_{ijt} = Q^C_{jt}, \quad j = 1, \dots, C, \ t = 1, \dots, TI, \tag{2}$$

$$Q_{ij} = \sum_{t \in TI} q_{ijt}, \quad i = 1, \dots, H, \ j = 1, \dots, C, \tag{3}$$

$$Q_{ij} \le U_{ij} y_{ij}, \quad i = 1, \dots, H, \ j = 1, \dots, C, \tag{4}$$

$$\begin{cases} q_{ijt} \ge 0, \\ R_t \ge 0, \end{cases} \quad i = 1, \dots, H, \ j = 1, \dots, C, \ t = 1, \dots, TI, \tag{5}$$

$$R_0 = R_T = 0, \tag{6}$$

$$y_{ij} = \{0, 1\}, \quad i = 1, \dots, H, \ j = 1, \dots, C. \tag{7}$$

In this formulation, y_{ij} is a binary variable which is one if a match between hot process stream i and cold process stream j occurs, and zero otherwise; q_{ijt} is the amount of heat exchanged between hot stream i and cold stream j in temperature interval t, R_{it} is the residual heat flow associated with hot stream i that cascades down from temperature interval t to temperature interval t+1, and Q_{ij} is the heat duty of match (i, j). The overall objective function minimizes the total number of units. Constraint (1) is the energy balance for hot stream i around temperature interval t and constraint (2) is the energy balance for cold stream j around temperature interval t. Constraint (3) finds the overall heat duty of match (ij). Constraint (4) sets this heat duty to zero when match (ij) does not exist. The nonnegativity constraints prevents heat flow from a low temperature to a higher one. Note that the residual heat flows into the first temperature interval and out of the last temperature interval are zero when there are no utilities above or below the cascade. The objective function and the constraints are linear, and the formulation involves both continuous and integer variables, making this a mixed integer linear programming problem.

Lower bounds on the solution of this problem are given by linear programming problems where some integer variables are fixed to either zero or one and the remainder are treated as continuous variables. The accuracy of these bounds depends upon the parameters U_{ij} is the fourth constraint. When these parameters are very large, the lower bounds will be quite far from the solution of the MILP.

The smallest acceptable value of U_{ij} is the minimum of the cooling requirements of stream i and the heating requirements of stream j:

$$U_{ij} = \min \left\{ \sum_{t \in TI} Q^H_{it}, \sum_{t \in TI} QC_{jt} \right\} .$$

Example 1 This example is from [3] and features three hot streams, two cold streams, and a cold utility. Table 1 gives the inlet and outlet stream temperatures and the flowrate heat capacities of each process stream and the cooling water duty.

Temperatures on the hot side of the cascade are 159°C, 128°C, and 36°C, while temperatures on the cold side are 149°C, 118°C and 26°C. There are two temperature intervals. Table 2 gives the heat released from hot streams to the temperature intervals, while Table 3 gives the heat absorbed by the cold streams from the temperature intervals.

Mixed Integer Linear Programming: Heat Exchanger Network Synthesis, Table 2
Q^H_{it}, heat released from hot stream *i* to temperature interval *t*

Stream	Temperature	Interval
	TI-1	TI-2
H1	7083.5	11635.5
H2	632.4	816.0
H3	1667.8	2044.4

Mixed Integer Linear Programming: Heat Exchanger Network Synthesis, Table 3
Q^C_{it}, heat absorbed by cold stream *i* from temperature interval *t*

Stream	Temperature	Interval
	TI-1	TI-2
C1	839.7	8583.6
C2	6079.1	
CW		8395.2

Mixed Integer Linear Programming: Heat Exchanger Network Synthesis, Table 4
Four solutions which satisfy minimum number of matches

Solution 1 Match Duty (Q_{ij})		Solution 2 Match Duty (Q_{ij})		Solution 3 Match Duty (Q_{ij})		Solution 4 Match Duty (Q_{ij})	
H1–C1	9423.3	H1–C1	7974.9	H1–C1	5711.1	H1–C1	4262.7
H1–C2	6079.1	H1–C2	6079.1	H1–C2	6079.1	H1–C2	6079.1
H1–CW	3234.6	H1–CW	4683.0	H1–CW	6946.8	H1–CW	8395.2
H2–CW	1448.4	H2–C1	1448.4	H2–CW	1448.4	H2–C1	1448.4
H3–W	3712.2	H3–CW	3712.2	H3–C1	3712.2	H3–C1	3712.2

In this example, the minimum number of units is 5, and there are four solutions to this MILP that meet this minimum (cf. Table 4).

Conclusions

Mixed integer linear programs are used in heat exchanger network synthesis to identify the minimum number of units, and a set of matches and their heat loads meeting the minimum. These MILPs are based upon a transshipment model of heat flow.

See also

- ► Chemical Process Planning
- ► Extended Cutting Plane Algorithm
- ► Generalized Benders Decomposition
- ► Generalized Outer Approximation
- ► Global Optimization of Heat Exchanger Networks
- ► MINLP: Application in Facility Location-allocation
- ► MINLP: Applications in Blending and Pooling Problems
- ► MINLP: Applications in the Interaction of Design and Control
- ► MINLP: Branch and Bound Global Optimization Algorithm
- ► MINLP: Branch and Bound Methods
- ► MINLP: Design and Scheduling of Batch Processes
- ► MINLP: Generalized Cross Decomposition
- ► MINLP: Global Optimization with αBB
- ► MINLP: Heat Exchanger Network Synthesis
- ► MINLP: Logic-based Methods
- ► MINLP: Outer Approximation Algorithm
- ► MINLP: Reactive Distillation Column Synthesis
- ► Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ► Mixed Integer Nonlinear Programming

References

1. Biegler LT, Grossmann IE, Westerberg AW (1997) Systematic methods of chemical process design. Prentice-Hall, Englewood Cliffs
2. Floudas CA (1995) Nonlinear and mixed-integer optimization. Oxford Univ. Press, Oxford
3. Gundersen T, Grossmann IE (1990) Improved optimization strategies for automated heat exchanger synthesis through physical insights. Comput Chem Eng 14(9):925
4. Gundersen T, Naess L (1988) The synthesis of cost optimal heat exchanger networks: An industrial review of the state-of-the-art. Comput Chem Eng 12(6):503
5. Jezowski J (1994) Heat exchanger network grassroot and retrofit design: The review of the state-of-the-art: Part I. Hungarian J Industr Chem 22:279–294
6. Jezowski J (1994) Heat exchanger network grassroot and retrofit design: The review of the state-of-the-art: Part II. Hungarian J Industr Chem 22:295–308
7. Papoulias SA, Grossmann IE (1983) A structural optimization approach in process synthesis - II: Heat recovery networks. Comput Chem Eng 7:707
8. Umeda T, Harada T, Shiroko K (1979) A thermodynamic approach to the structure in chemical processes. Comput Chem Eng 3:373

Mixed Integer Linear Programming: Mass and Heat Exchanger Networks

MEN, MHEN

KATERINA P. PAPALEXANDRI
bp Upstream Technol., Middlesex, UK

MSC2000: 93A30, 93B50

Article Outline

Keywords

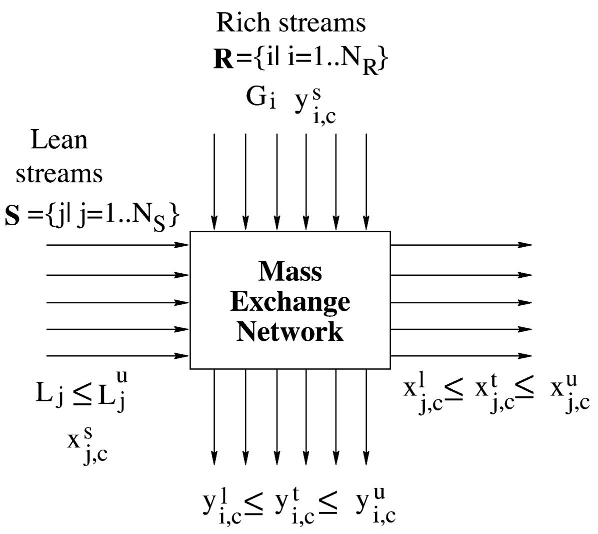
MILP; Mass and heat exchange; Separation

Separation networks involving mass transfer operations that do not require energy (e. g. absorption, liquid-liquid extraction, ion-exchange etc.) are characterized as mass exchange networks (MEN). These appear in the chemical industries mostly in waste treatment, but also, in feed preparation, product separation, recovery of valuable materials, etc. A *mass exchanger*, in this context, is any counter-current, direct-contact mass transfer unit, where one or more components are transferred at constant temperature and pressure from one process stream, which is characterized as *rich stream*, to another process or utility stream, characterized as *lean stream*. Mass integration aims to the purification of the rich streams and the recovery of valuable or hazardous materials at the minimum total cost (investment and operating cost of auxiliary streams). In the specific case, when the mass transfer operations take place at the same temperature, or heating/cooling requirements are negligible, the integration problem is limited to the synthesis of a mass exchanger network (MEN) only. When mass exchange operations at different temperature levels are encountered, mass and heat exchanger networks (MHEN) may be considered simultaneously.

MEN synthesis involves a set of rich streams, in terms of one or more components, $\mathbf{R} = \{i: i = 1, \dots, N_R\}$, with known flowrates, G_i, inlet and outlet compositions for the components of interest, y^s_{ic}, y^t_{ic} (exact values or bounds) respectively, and a set of process or auxiliary lean streams (*mass separating agents*, MSAs), $\mathbf{S} = \{j: j = 1, \dots, N_S\}$ with known cost, inlet and outlet compositions for the same components, x^s_{jc}, x^t_{jc} (exact values or bounds), as shown in Fig. 1.

The *synthesis problem* refers to the selection of the appropriate lean streams and their flowrates, L_j, the mass exchange operations (*mass exchange matches*), the mass transfer load for each separator and its required size, and the configuration of the overall network.

Mass transfer in each mass exchanger is governed by the first and second thermodynamic laws, as is heat transfer in heat exchangers. Mass transfer of a component c from a rich to a lean stream is feasible if the composition of c in the rich phase is greater than the equi-


Rich streams
$\mathbf{R}$={i| i=1..N_R}
G_i $y^s_{i,c}$
Lean streams
$\mathbf{S}$ ={j| j=1..N_S}
Mass Exchange Network
$L_j \leq L^u_j$
$x^s_{j,c}$
$x^l_{j,c} \leq x^t_{j,c} \leq x^u_{j,c}$
$y^l_{i,c} \leq y^t_{i,c} \leq y^u_{i,c}$

Mixed Integer Linear Programming: Mass and Heat Exchanger Networks, Figure 1

librium composition with respect to the lean phase:

$$y_c \geq f(x_c) + \epsilon, \qquad (1)$$

where $f(x_c)$ is the equilibrium relation and ϵ is a *minimum composition difference* that ensures feasible mass transfer in a separator of finite size, in analogy to $\Delta T_{\min}$ in heat exchangers. This analogy led to the development of synthesis methods for mass exchanger networks employing mixed integer optimization techniques, similar to heat exchanger networks (cf. Mixed Integer Linear Programming: Mass and Heat Exchanger Networks; ► MINLP: Mass and Heat Exchanger Networks), that are categorized into the *sequential synthesis* and the *simultaneous synthesis* methods.

The *sequential MEN synthesis method*, introduced in [3] and [4] involves the following steps:

1) Minimum cost of mass separating agents (minimum utility problem), to determine the optimal flows of the mass separating agents.
2) Minimum number of mass exchanger units, for fixed MSA flows, to determine the mass exchange matches.
3) Network configuration and separator sizes for fixed mass exchange operations.

The first two synthesis steps involve the solution of linear and mixed integer linear problems.

A useful tool of the sequential MEN synthesis method is the *composition interval diagram*, CID, where thermodynamic feasibility of mass transfer is explored mapping the rich and the lean streams on equivalent composition scales, that are derived from the mass transfer feasibility requirements in (1). In general, the composition equivalent scales and the minimum composition difference, ϵ, are defined for each component of interest and each pair of rich and lean streams. In the simple case of a single component, where mass transfer is independent of the presence of other components in the rich streams, the CID is constructed as illustrated in Fig. 2.

Feasible rich-to-lean mass transfer is guaranteed within a composition interval when the equilibrium relation $f(x_c)$ is convex within the interval. When $f(x_c)$ is convex in the whole composition range, only inlet compositions are required to construct the CID [8].

The minimum cost of mass separating agents is found employing a *transshipment model*, where the components of interest are the transferred commodities, the rich and the lean streams are considered as sources and sinks respectively, and the composition intervals define the intermediate nodes [4]. The model involves energy balances around the temperature intervals (intermediate nodes):

$$(\text{TP1})\quad\begin{cases}\min & \sum_j c_j L_j \\ \text{s.t.} & \delta_{k-1} + \sum_{i\in R_k} WR_k^i \\ & \quad = \delta_k + \sum_{j\in S_k} WS_k^j \\ & 0 \le L_j \le L_j^{\text{up}},\ j \in S, \\ & \delta_0 = \delta_{N_{\text{int}}} = 0; \\ & \delta_k \ge 0,\ k = 1,\dots,N_{\text{int}}-1,\end{cases}$$

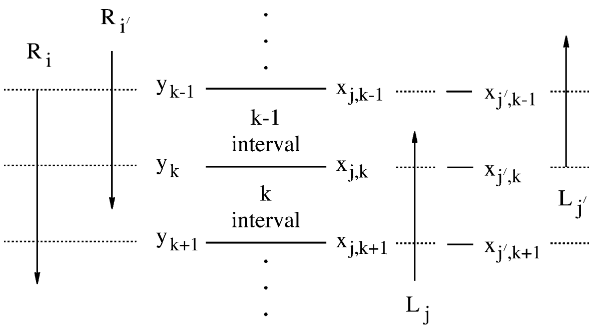
where

- R_k is the set of rich streams, present in interval k;
- S_k is the set of lean streams, present in interval k;
- N_{int} is the number of composition intervals;
- WR_k^i is the mass exchange load of rich stream i in interval k,

$$WR_k^i = G_i(y_k - \max(y_{k+1}, y_i^t));$$

- WS_k^j is the mass exchange load of lean stream j in interval k,

$$WS_k^j = L_j(\min(x_j^t, x_{jk}) - x_{jk+1});$$



Mixed Integer Linear Programming: Mass and Heat Exchanger Networks, Figure 2
Composition interval diagram

- δ_k is the residual mass exchange load in interval k.

Problem (TP1) results in the optimal flows of the mass separating agents and the identification of the *pinch points*, i. e. the thermodynamic bottlenecks in mass transfer. The pinch points are defined by zero residual flows and divide the mass exchange network into subnetworks. Mass transfer between different subnetworks (i. e. across the pinch) increases the cost of mass separating agents.

An assumption in (TP1) is that molar flows of the rich and the lean streams are constant. If significant flowrate variations take place, compositions and mass exchange loads are calculated based on nontransferable components.

The following cases are distinguished:

- Fixed inlet and outlet compositions.
  Then, (TP1) is an LP problem.
  When multiple components are considered, the CID is defined for all the components of interest and (TP1) corresponds to the multicommodity transshipment model. The pinch points are then determined by the component that requires the greater MSA flows.
- Variable outlet compositions.
  Then, the mass exchange loads of the rich and lean streams in their final intervals (defined by the upper and lower bounds on their outlet compositions) are variables. Problem (TP1) can still be solved as an LP [9], considering the variable mass exchange loads explicitly in the model.

Variable inlet compositions usually require flexible mass exchange networks to accommodate the variations and define a different problem. For a single component it has been shown that the minimum MSA cost corresponds to the lower bounds of the inlet compositions [8].

For nonconvex equilibrium relations, (TP1) cannot guarantee feasible mass transfer throughout the composition range, while the predicted MSA cost is a lower bound to the actual minimum one. B.K. Srinivas and M.M. El-Halwagi suggested in [14] an iterative procedure to calculate the minimum required MSA cost, that involves two major steps:

i) a 'feasibility problem', where 'critical' composition levels are identified and included in the CID (nonconvex NLP step, that requires global optimization methods), and
ii) (TP1) with updated intervals, which calculates increasing lower bounds to the minimum MSA cost.

Instead of target outlet compositions for the rich streams, it may be of interest to remove a certain total mass load of pollutants. Then, (TP1) is solved with variable rich outlets and a fixed total mass exchange load [10]:

$$M_c = \sum_i G_i(y_i^s - y_i^t)$$

The minimum-utility-cost problem has been alternatively formulated as an LP or MINLP problem, based on total mass balances and the following property:

$$\left\{\begin{array}{l}\text{Mass lost by all the rich}\\ \text{streams below each}\\ \text{pinch point candidate}\end{array}\right\} - \left\{\begin{array}{l}\text{Mass gained by all the lean}\\ \text{streams below each}\\ \text{pinch point candidate}\end{array}\right\} \le 0 \quad (2)$$

and employing binary variables to denote the relative position of variable outlet compositions with respect to each pinch point candidate in the CID [5,6,8,9].

The minimum number of mass exchange operations (units) for fixed MSA cost is determined in each subnetwork in a second step, in an attempt to minimize the fixed cost of the separators. The minimum number of mass exchangers is found employing the *expanded transshipment model*, where the existence of a mass exchange match-separator in a subnetwork is denoted by a binary variable:

$$E_{ijm} = \begin{cases} 1, & \text{when streams } i, j \\ & \text{exchange mass} \\ & \text{in subnetwork } m \\ 0, & \text{otherwise.} \end{cases}$$

For a single component, the minimum number of mass exchanger units is given by the following MILP problem [4]:

$$(\text{TP2}) \quad \begin{cases} \min & \displaystyle\sum_m \sum_{i \in R_m} \sum_{j \in S_m} E_{ijm} \\ \text{s.t.} & \delta_{ik} - \delta_{ik-1} + \displaystyle\sum_{j \in S_{mk}} M_{ijk} = WR_k^i, \\ & k \in I_m,\ i \in R_{mk},\ m \in M, \\ & \displaystyle\sum_{i \in R_{mk}} M_{ijk} = WS_k^j, \\ & k \in I_m,\ j \in S_{mk},\ m \in M \\ & \displaystyle\sum_{k \in I_m} M_{ijk} - E_{ijm} U_{ijm} \le 0 \\ & \delta_{ik} \ge 0,\ k \in I_m,\ i \in R_m, \\ & M_{ijk} \ge 0,\ k \in I_m, \\ & i \in R_{km},\ j \in S_{km} \\ & E_{ijk} = 0, 1,\ k \in I_m, \\ & i \in R_{km},\ j \in S_{km}, \end{cases}$$

where

- R_m is the set of rich streams, present in subnetwork m,
- S_m is the set of lean streams, present in subnetwork m,
- I_m is the set of intervals in subnetwork m,
- R_{km} is the set of rich streams, present in interval k of subnetwork m, or above,
- S_{km} is the set of lean streams, present in interval k of subnetwork m,
- WR_k^i is the mass exchange load of rich stream i in interval k,
- δ_{ik} is the residual mass exchange load of rich stream i in interval k,
- WS_k^j is the mass exchange load of lean stream j in interval k, as determined by (TP1),
- M_{ijk} is the mass exchange load between i and j in interval k,

- U_{ijm} is an upper bound to the possible mass exchange load between i and j in subnetwork m,

$$U_{ijm} = \min\left(\sum_{k\in I_m} WR_k^i, \sum_{k\in I_m} WS_k^j\right).$$

Srinivas and El-Halwagi have shown [14] that, when the equilibrium relations around a pinch point are not convex, a mass exchanger can straddle the pinch and still be thermodynamically feasible. To account for such cases, exchangers across the pinch points can be considered introducing extra binary variables:

$$I_{ijp} \leq M_{ijp},$$
$$I_{ijp+1} \leq M_{ijp+1},$$
$$I_{ijp} + I_{ijp+1} \geq 2B_{ijp},$$
$$I_{ijp}, I_{ijp+1} \in \{0,1\},$$
$$B_{ijp} \in \{0,1\},$$

where

- I_{ijp} denotes that streams i and j exchange mass at the interval directly above pinch point p,
- I_{ijp+1} denotes that streams i and j exchange mass at the interval directly below pinch point p,
- B_{ijp} denotes the existence of an exchanger between streams i and j, across the pinch p.

Then, the number of required units to minimize is given by:

$$\sum_m \left(\sum_{i\in R_m}\sum_{j\in S_m} E_{ijm} - \sum_p B_{ijp}\right).$$

Note, that I_{ijp}-variables can be relaxed to continuous, due to total unimodularity of the model with respect to these variables:

$$0 \leq I_{ijp},\ I_{ijp+1} \leq 1$$

Problem (TP2) may not have a unique solution. Alternative combinations of mass exchange matches, featuring the minimum MSA cost, may be generated by solving (TP2) iteratively and including integer cuts. These do not necessarily correspond to networks of the same overall cost.

The expanded transshipment model can also be employed to determine the minimum MSA cost, considering variable mass loads for the lean streams. Then, forbidden or restricted mass exchange operations can be explicitly accounted for.

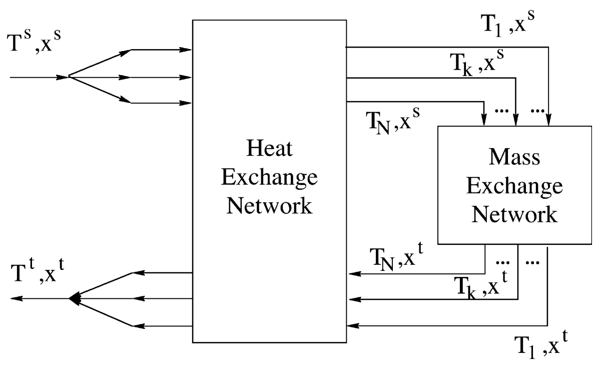
Although (TP2) does not determine the network structure, stream splitting and exchanger connectivity may be guided by the resulting mass exchange load distribution in each composition interval [4]. The actual network configuration is found in a next step, employing heuristic methods [3,5] or superstructure methods (NLP models).

Special cases of mass exchange networks have been studied:

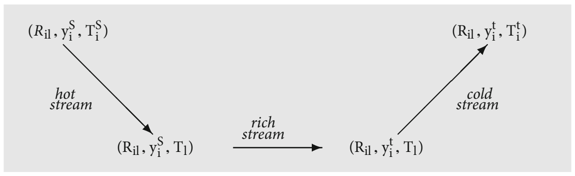
- MEN and *regeneration networks* [5,11].
  The regeneration of mass separating agents by auxiliary streams can be considered simultaneously with the main MEN, in another mass exchanger network, where the MSAs behave as the rich streams. In this case, the CID is extended to include the equivalent composition scales of the regenerating agents. The inlet and outlet compositions of the lean streams in the main MEN are in general variables.
- Reactive mass exchange networks [6,11,14]
  Rich-to-lean mass transfer may involve interphase mass transfer and chemical reaction in the lean phase, at constant temperature. Mass exchange operations of this kind are considered deriving the equilibrium relations based on chemical equilibrium.

The main advantage of the sequential synthesis method for mass exchange networks is that simple optimization models are solved. However, unless the MSA cost is dominant, as synthesis decisions are fixed from one step to the next, important trade-offs between operating and capital cost are not exploited and overall cost optimality cannot be guaranteed. Furthermore, the minimum composition difference, ϵ that defines the mass recovery levels in (TP1) and (TP2), is in general, an optimization variable for each mass exchanger separately. In the sequential synthesis method this is fixed arbitrarily to a possibly conservative value for the construction of the CID. El-Halwagi and V. Manousiouthakis [4] suggested a two-level optimization procedure to select a unique ϵ for all mass exchange operations, based on the impact of ϵ on the final MEN cost, still, not exploiting the overall cost trade-offs.

When isothermal mass exchange operations take place at different temperature levels, the operating and overall mass integration costs are affected by the heating and cooling requirements of the system. Energy integration between the rich and lean streams can be



Mixed Integer Linear Programming: Mass and Heat Exchanger Networks, Figure 3



Mixed Integer Linear Programming: Mass and Heat Exchanger Networks, Figure 4
Rich substream with $T_i^s \le T_l \le T_i^t$

considered within a mass and heat exchanger network synthesis problem (MHEN) to reduce the total cost. The overall problem is addressed combining MEN and HEN synthesis tools. The optimal temperature of mass exchange is defined for each pair of rich and lean streams by the equilibrium relations that limit mass transfer

$$y_i \ge K_{ij}(T) x_j ,$$

where $K_{ij}(T)$ is a known function of temperature.

In the sequential synthesis framework, the overall minimum operating cost for the network (cost of mass separating agents and heating/cooling utilities) may be calculated from a combined mass and heat transshipment model. Each stream is considered to consist of substreams, of the same inlet and outlet composition and temperature, each of which participates to isothermal mass exchange operations at a different temperatures. Srinivas and El-Halwagi proved [13], that, for monotonic dependence of the equilibrium constant on temperature, the overall utility cost of the combined MHEN is independent of such a stream decomposition, see Fig. 3.

Although the mass exchange temperatures ($T_1, \dots, T_N$) are variables, their relative position with respect to inlet and outlet stream temperatures (greater or less) can be prepostulated. Thus, the rich and lean substreams define hot (or cold) streams before their mass exchange operations and cold (or hot) streams afterwards, cf. Fig. 4.

A CID is constructed, similarly to the simple MEN case, involving the several substreams with variable flows, and thus, variable mass loads in each composition interval. Mass exchange is permitted between substreams of the same temperature. A *temperature interval diagram*, TID, is also constructed, involving the hot and cold substreams and the available heating and cooling utilities, with variable heat loads per interval, due to the variable substream flows. In order to avoid discrete decisions (i. e. presence or not of streams in temperature intervals with variable limits), the temperature range for each mass transfer operation is discretized and a substream is associated with each candidate temperature [13].

The minimum utility cost is found from the solution of the combined LP transshipment model, which, for a single component is as follows:

$$(\text{TP3}) \quad \left\{ \begin{aligned} \min & \left(\sum_{j \in S} c_j L_j \right. \\ & + \sum_{n \in TI} \sum_{h \in HU_n} c_h QHU_{hn} \\ & \left. + \sum_{n \in TI} \sum_{c \in CU_n} c_c QCU_{cn} \right) \end{aligned} \right.$$

such that

$$\delta_{l_i k} - \delta_{l_i k-1} + \sum_{j \in S} \sum_{l'_j \in SS_{jk}} M_{l_i l'_j k} = WR_k^{l_i} ,$$

$$k \in CI, \; i \in R, \; l_i \in RS_{ik} ,$$

$$\sum_{i \in R} \sum_{l'_i \in RS_{ik}} M_{l'_i l_j k} = WS_k^{l_j} ,$$

$$k \in CI, \; j \in S, l_j \in SS_{jk} ,$$

$$\theta_{l_s n} - \theta_{l_s n-1}$$
$$+ \sum_{s' \in R \cup S} \sum_{l'_{s'} \in CS_{s' n}} Q_{l_s l'_{s'} n}$$

$$+ \sum_{c \in CU_n} QC_{l_s cn} = QS_{l_s n},$$

$$n \in TI,\ s \in R \cup S,\ l_s \in HS_{sn}, \qquad \theta^h_{hn} - \theta^h_{hn-1}$$

$$+ \sum_{s \in R \cup S} \sum_{l_s \in CS_{sn}} QH_{hl_s n} = QHU_{hn},$$

$$n \in TI,\ h \in HU_n,$$

$$\sum_{s' \in R \cup S} \sum_{l'_{s'} \in HS_{s'n}} Q_{l'_{s'} l_s n}$$

$$+ \sum_{h \in HU_n} QH_{hl_s n} = QS_{l_s n},$$

$$n \in TI,\ s \in R \cup S,\ l_s \in CS_{sn},$$

$$\sum_{s \in R \cup S} \sum_{l_s \in HS_{sn}} QC_{l_s cn} = QCU_{cn},$$

$$n \in TI,\ c \in CU_n,$$

$$\delta_{l_i k} \geq 0,\ k \in CI,\ i \in R,\ l_i \in RS_{ik},$$

$$\theta_{l_s n} \geq 0,\ n \in TI,\ s \in R \cup S,\ l_s \in HS_{sn},$$

$$\theta^h_{hn} \geq 0, n \in TI,\ h \in HU_n,$$

$$M_{l_i l'_j k} \geq 0,$$
$$k \in CI,\ i \in R,\ j \in S,$$
$$l_i \in RS_{ik},\ l'_j \in SS_{jk},$$

$$M_{l_i l'_j k} = 0,$$
$$k \in CI,\ i \in R,\ j \in S,$$
$$l_i \in RS_{ik},\ l'_j \in SS_{jk},$$
$$(l_i l'_j) \in FM,$$

$$\delta_{l_i 0} = \delta_{l_i N_{CI}} = 0,$$
$$i \in R,\ l_i \in RS_i,$$

$$\theta_{l_s 0} = \theta_{l_s N_{TI}} = 0,$$
$$s \in R \cup S,\ l_s \in HS_s,$$

$$\theta^h_{h0} = \theta^h_{hN_{TI}} = 0,$$
$$h \in HU,$$

where

- CI is the set of composition intervals k,
- TI is the set of temperature intervals n,
- RS_{ik} is the set of substreams of rich stream i, of variable flow, G_{l_i}, such that
$$\sum_l G_{l_i} = G_i,$$
present in interval k, or above,
- SS_{jk} is the set of substreams of lean stream j, of variable flow, L_{l_j}, such that
$$\sum_l L_{l_j} = L_j,$$
present in interval k,
- HS_{sn} is the set of hot substreams of stream s, present in interval n, or above,
- CS_{sn} is the set of cold substreams of stream s, present in interval n,
- HU_n is the set of hot utilities, present in interval n, or above,
- CU_n is the set of cold utilities, present in interval n,
- $WR^{l_i}{}_k$ is the mass exchange load of substream l_i, in interval k,
$$WR^{l_i}_k = G_{l_i}(y_k - \max(y_{k+1}, y_i spt)),$$
- $WS^{l_j}{}_k$ is the mass exchange load of substream l_j, in interval k,
$$WS^{l_j}_k = L_{l_j}(\min(x^t_j, x_{jk}) - x_{jk+1}),$$
- $\delta_{l_i k}$ is the residual mass load of substream l_i in interval k,
- $M_{l_i l'_j k}$ is the mass exchange load between l_i and l'_j, in k,
- FM is the set of mass exchanging substreams that are at different temperatures,
- $QS_{l_s n}$ is the heat load of substream l_s in interval n,
- $Q_{l_s l_{s'} n}$ is the heat exchange load between l_s and $l_{s'}'$ in interval n,
- $\theta_{l_s n}$ is the residual heat load of hot substream l_s in interval n,
- QHU_{hn} is the heat load of hot utility h in interval n,
- QCU_{cn} is the heat load of cold utility c in interval n,
- $QH_{hl_s n}$ is the heat exchange load between hot utility h and l_s in interval n,
- θ^h_{hn} is the residual heat load of hot utility h in interval n,
- $QC_{l_s cn}$ is the heat exchange load between l_s and cold utility c in interval n.

Problem (TP3) results in the minimum utility cost and the corresponding flows of separating agents and heating/cooling utility streams, the optimal decomposition of each stream into substreams of fixed mass exchange temperature and the mass and heat exchange pinch points and corresponding subnetworks.

The minimum operating cost of the combined MHEN can alternatively be found applying the first and second thermodynamic laws (property in (2)) on the composition and temperature interval diagrams [13].

The minimum number of mass and heat exchangers is determined in a second step through the expanded MILP transshipment model, separately in each mass and heat exchanger subnetwork. The final network configurations and unit sizes are determined in a final step, applying heuristic rules or superstructure models.

Additional disadvantages of the sequential MHEN synthesis method, compared to the synthesis of simple MEN, are that:

i) the mass and heat exchange networks are assumed separable and
ii) the intermediate mass exchange temperatures are decided in the first step; this forbids full exploitation of the mass/heat integration trade-offs, as capital cost implications of such decision is not accounted for.

Modeling concepts from the sequential mass and heat exchanger network synthesis methods, employing LP and MILP optimization models, have been extended to explore distillation networks [1], pervaporation systems [12] and other energy-requiring separation networks [2,7].

See also

- ► Chemical Process Planning
- ► Extended Cutting Plane Algorithm
- ► Generalized Benders Decomposition
- ► Generalized Outer Approximation
- ► Global Optimization of Heat Exchanger Networks
- ► MINLP: Application in Facility Location-Allocation
- ► MINLP: Applications in Blending and Pooling Problems
- ► MINLP: Applications in the Interaction of Design and Control
- ► MINLP: Branch and Bound Global Optimization Algorithm
- ► MINLP: Branch and Bound Methods
- ► MINLP: Design and Scheduling of Batch Processes
- ► MINLP: Generalized Cross Decomposition
- ► MINLP: Global Optimization with αBB
- ► MINLP: Heat Exchanger Network Synthesis
- ► MINLP: Logic-based Methods
- ► MINLP: Outer Approximation Algorithm
- ► MINLP: Reactive Distillation Column Synthesis
- ► Mixed Integer Linear Programming: Heat Exchanger Network Synthesis
- ► Mixed Integer Nonlinear Programming

References

1. Bagajewicz MJ, Manousiouthakis V (1992) Mass/heat exchange network representation of distillation networks. AIChE J 38:1769–1800
2. El-Halwagi MM, Hamad AA, Garrison GW (1996) Synthesis of waste interception and allocation networks. AIChE J 42:3087–3101
3. El-Halwagi MM, Manousiouthakis V (1989) Synthesis of mass exchange networks. AIChE J 35:1233–1243
4. El-Halwagi MM, Manousiouthakis V (1990) Automatic synthesis of mass exchange networks with single component targets. Chem Eng Sci 45:2813–2831
5. El-Halwagi MM, Manousiouthakis V (1990) Simultaneous synthesis of mass-exchange and regeneration networks. AIChE J 36:1209–1219
6. El-Halwagi MM, Srinivas BK (1992) Synthesis of reactive mass exchange networks. Chem Eng Sci 47:2113–2119
7. El-Halwagi MM, Srinivas BK, Dunn RF (1995) Synthesis of optimal heat-induced separation networks. Chem Eng Sci 50:81–97
8. Gupta A, Manousiouthakis V (1993) Minimum utility cost of mass exchange networks with variable single component supplies and targets. Industr Eng Chem Res 32:1937–1950
9. Gupta A, Manousiouthakis V (1996) Variable target mass-exchange network synthesis through linear programming. AIChE J 42:1326–1340
10. Kiperstok A, Sharrat PN (1995) On the optimization of mass exchange networks for removal of pollutants. Chem Eng Res Des 73:271–277
11. Papalexandri KP, Pistikopoulos EN, Floudas CA (1994) Mass exchange networks for waste minimization: A simultaneous approach. Chem Eng Res Des 72:279–294
12. Srinivas BK, El-Halwagi MM (1993) Optimal design of pervaporation systems for waste reduction. Comput Chem Eng 17:957–970
13. Srinivas BK, El-Halwagi MM (1994) Synthesis of combined heat and reactive mass exchange networks. Chem Eng Sci 49:2059–2074

14. Srinivas BK, El-Halwagi MM (1994) Synthesis of reactive mass exchange networks with general nonlinear equilibrium relations. AIChE J 40:463–472

Mixed Integer Nonlinear Bilevel Programming: Deterministic Global Optimization

ZEYNEP H. GÜMÜŞ[1,2],
CHRISTODOULOS A. FLOUDAS[3]
[1] Department of Physiology and Biophysics, Weill Medical College, Cornell University, New York, USA
[2] The HRH Prince Alwaleed Bin Talal Bin Abdulaziz Institute for Computational Biomedicine, Weill Medical College, Cornell University, New York, USA
[3] Department of Chemical Engineering, Princeton University, Princeton, USA

Article Outline

Keywords and Phrases

Bilevel programming; Bilevel nonlinear; Bilevel optimization; Mixed integer optimization; Global optimization; Two-level optimization; Mixed integer nonlinear

Introduction

The global optimization of classes of mixed integer nonlinear bilevel optimization problems is addressed. For problems where the integer variables participate in both the inner and the outer problems, the outer level may involve general mixed-integer nonlinear functions. The inner level may involve functions that are mixed-integer nonlinear in outer variables, linear, polynomial, or multilinear in inner integer variables and linear in inner continuous variables. The technique is based on reformulating the mixed-integer inner problem as continuous by its convex hull representation [11,12] and solving the resulting nonlinear bilevel optimization problem by a novel deterministic global optimization framework.

Formulation

The general mixed-integer nonlinear Bilevel Programming Problem (BLP) formulation is:

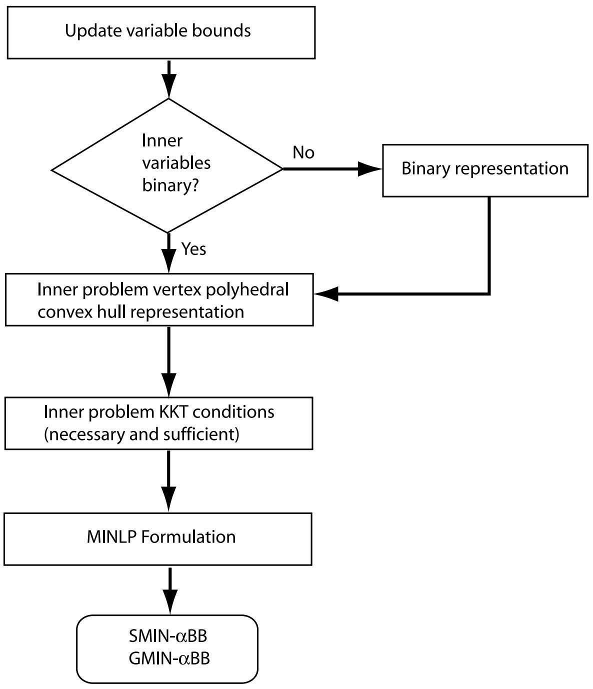
$$\begin{aligned}
&\min_{x} F(\boldsymbol{x},\boldsymbol{y})\\
&\text{s.t. } \boldsymbol{G}(\boldsymbol{x},\boldsymbol{y}) \geq 0\\
&\quad \boldsymbol{H}(\boldsymbol{x},\boldsymbol{y}) = 0\\
&\quad \min_{y} f(\boldsymbol{x},\boldsymbol{y})\\
&\quad \text{s.t. } \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \geq 0\\
&\qquad \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) = 0\\
&\quad x_1,..,x_i \in \Re^{n_1}, y_1,..,y_j \in \Re^{n_2},\\
&\quad x_{i+1},..,x_{n_1} \in Z^+, y_{j+1},..,y_{n_2} \in Y_{\text{IN}} \subseteq Z^+ .
\end{aligned} \tag{1}$$

where $\boldsymbol{x}$ is a vector of outer problem variables, of which i are continuous and $n_1 - i$ are integer, y is a vector of inner problem variables, of which j are continuous and $n_2 - j$ are integer, $F(\boldsymbol{x},\boldsymbol{y})$ is the outer objective function, $\boldsymbol{H}(\boldsymbol{x},\boldsymbol{y})$ are outer equality constraints, $\boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})$ are outer inequality constraints, $f(\boldsymbol{x},\boldsymbol{y})$ is the inner objective function, $\boldsymbol{h}(\boldsymbol{x},\boldsymbol{y})$ are inner equality constraints, and $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y})$ are inner inequality constraints. The applications of BLP are many and diverse [4,6,7]; if these problems involve discrete decisions in addition to continuous ones, then the mixed-integer BLP models arise.

Classes

The nonlinear mixed integer BLP can be classified into four different categories, depending on the existence of integer variables in the outer or the inner problems:

(I). Integer Upper, Continuous Lower BLP;
(II). Purely Integer BLP;
(III). Continuous Upper, Integer Lower BLP;
(IV). Mixed-Integer Upper and Lower BLP.


Update variable bounds
Inner variables binary?
No
Binary representation
Yes
Inner problem vertex polyhedral convex hull representation
Inner problem KKT conditions (necessary and sufficient)
MINLP Formulation
SMIN-αBB
GMIN-αBB

Mixed Integer Nonlinear Bilevel Programming: Deterministic Global Optimization, Figure 1
Algorithm flowsheet for type II,III,IV BLPs

The existence of both integer and nonlinear terms in the above problem classes require special solution techniques. The specific mathematical structure of the mixed integer nonlinear BLP is of great import in developing corresponding solution strategies. Problems of Type I can be addressed with existing BLP solution approaches. For problems of Type II, enumeration methods can be applied. However, BLPs of Type III and IV are the most difficult to solve.

BLPs with Inner Integer Variables

The conventional solution method of the continuous BLP is to transform it into a single level problem by replacing the inner problem with the set of equations that define its Karush–Kuhn–Tucker (KKT) optimality conditions. However, the KKT optimality conditions use gradient information, so the conventional approach is not applicable when integer inner variables exist. Furthermore, if the integrality constraint is relaxed on the inner integer variables, the solution of this relaxed BLP does not provide a valid lower bound on the solution of the mixed-integer BLP [9]. Note that even if the optimal solution of the relaxed BLP is integral in $\boldsymbol{y}$, this may not be a globally optimal solution of the original BLP [9]. Thus, the conventional KKT-based methods inherently fail in locating the global optimum.

The BLP with inner mixed-integer variables can be transformed into an equivalent BLP with inner mixed-binary (0-1) variables as follows. Every inner problem integer variable y_j, with upper and lower bounds $y_j^{\mathrm{L}} \leq y_j \leq y_j^{\mathrm{U}}$ is converted into a set of binary variables using the formula [2]:

$$y_j = y_j^{\mathrm{L}} + z_{j1} + 2z_{j2} + 4z_{j3} + \ldots + 2^{(N_j-1)} z_{jN} \quad (2)$$

where $\boldsymbol{z}_j$ is a vector of (0-1) variables and N_j is the minimum number of (0-1) variables needed:

$$N_j = 1 + \mathrm{INT}\left(\frac{\log(y_j^{\mathrm{U}} - y_j^{\mathrm{L}})}{\log(2)}\right) \quad (3)$$

such that INT truncates its real argument to an integer.

The only time that the KKT optimality conditions are applicable to solve the BLP with mixed-binary $\boldsymbol{y}$ is when the following property is satisfied [8]:

Property 1 *If the inner problem constraint set, Y_{IN}, defines a vertex polyhedral convex hull and all the vertices of the convex hull lie in Y_{IN}, then the optimal inner problem integer solution is equivalent to its linear programming relaxation. As a result, the Karush–Kuhn–Tucker, KKT, conditions of relaxed inner linear problem are necessary and sufficient to define the optimal inner problem integer solution.*

The property is also satisfied when outer variables exist in the inner problem constraints, such that the inner problem vertex polyhedral convex envelope is defined *parametrically in $\boldsymbol{x}$*. Hence, the integer solution of the inner problem lies at a vertex of the inner solution set and the KKT optimality conditions locate the true optimal solution [8].

Here, a global optimization procedure is presented for BLPs of Type II, III and IV that is based on a reformulation/linearization scheme combined with a global optimization framework. The idea is that if the inner problem constraint set has a vertex polyhedral convex envelope, then *Property 1* is satisfied and the mixed-integer inner problem can be converted into a continuous problem of equivalent form. The application of the reformulation/linearization technique results in the convex hull representation for several classes of inner problems.

Reformulation/Linearization

The mixed-binary inner problem constraint set is transformed into the continuous domain by converting it into a polynomial programming problem and then relinearizing it into an extended linear problem by a method based on [11]. First, a polynomial factor is defined as follows:

$$F_n(J_1, J_2) = \left\{ \begin{array}{l} \left(\prod_{j \in J_1} y_j\right)\left(\prod_{i \in J_2} (1-y_j)\right) , \\ J_1, J_2 \subseteq N_y \equiv 1, .., n_y , \\ \text{s.t. } J_1 \cap J_2 = \emptyset,\ |J_1 \cup J_2| = n_y \end{array} \right\} . \tag{4}$$

Using this polynomial factor, the convex hull of the inner problem, Y_{IN}, is obtained. If the inner optimization problem is linear, then the 2-step process is as follows:

Step 1 Reformulation. Multiply every constraint, including $0 \leq y \leq 1$, with every factor defined as above and use the relationship $y_j^2 = y_j, \forall j = 1, .., n_y$ to linearize terms polynomial in $\boldsymbol{y}$. Include in the inner problem constraint set the nonnegativities on all possible factors of degree n_y (i. e. $F_n(J_1, J_2) \geq 0$ for all (J_1, J_2) of order n_y).
Step 2 Linearization. Linearize the inner constraints that are multilinear in $\boldsymbol{y}$, such as $\Pi_{j \in J} y_j$ by substituting a z_J for each set J with $|J| \geq 2$, with the elements of J in increasing order. (i. e. a new variable z_{ij} is introduced to substitute for a bilinear term $(y_i y_j = z_{ij})$ and further substitution is performed for multilinear terms). At constant $\boldsymbol{x}$, the resulting inner constraint set describes a polytope with all vertices defined by binary values and characterizes the convex hull of feasible solutions for any inner problem that is linear or polynomial binary in $\boldsymbol{y}$-variables.

If the inner optimization problem is mixed-binary linear or polynomial, the problem constraints are again multiplied by n_y-degree polynomial factors composed of the n_y binary variables and their complements and the resulting nonlinear problem is linearized by a substitution of new variables. Additional nonlinear terms arise from the multiplication of the n_y-degree polynomial factors with the inner problem linear continuous terms in $\boldsymbol{y}$, that are also linearized through a redefinition [8,12]. This transformation is applicable when in the mixed-binary inner problem, the continuous $\boldsymbol{y}$ are $0 \leq y \leq 1$. Note that there are no such restrictions on the outer problem $\boldsymbol{x}$-variables in both inner and outer problems.

Inner Problem KKT Conditions and Complementarity

After reformulation/relinearization, the inner problem is replaced by the set of equations that define its necessary and sufficient KKT optimality conditions:

$$\begin{aligned} h_i^r(x, y^*) &= 0, \quad i \in I , \\ \frac{\partial f(x, y^*)}{\partial y^*} & \\ + \sum_{j=1}^{J} \lambda_j \frac{\partial g_j^r}{\partial y^*} + \sum_{j=1}^{J} \mu_i^* \frac{\partial h_i^r}{\partial y^*} &= 0 , \\ g_j^r(x, y^*) + s_j^* &= 0 , \quad j \in J \\ \lambda_j^* s_j^* &= 0 , \quad j \in J \quad \text{(CS)} \\ \lambda_j^* s_j^* &\geq 0 , \quad j \in J \end{aligned} \tag{5}$$

where f^r, $\boldsymbol{h}^r$ and $\boldsymbol{g}^r$ are the reformulated inner objective, equality and inequality constraints, $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are the Lagrange multipliers of the inner inequality and equality constraints, and $\boldsymbol{s}$ are the slack variables associated with the complementarity constraints.

Active Set Strategy

The complementarity condition constraints, (CS) involve discrete decisions on the choice of the inner problem active constraint set. The set changes when at least one inequality function and its Lagrange multiplier are equal to zero. This imposes a major difficulty in the solution of the transformed problem. To overcome this difficulty, the Active Set Strategy [5,8] is employed, such that the complementarity constraints are reformulated as:

$$\begin{aligned} \lambda_j - UY_j &\leq 0 , \quad j \in J \\ s_j - U(1 - Y_j) &\leq 0 , \quad j \in J \\ \lambda_j, s_j &\geq 0 , \quad j \in J \\ Y_j &\in \{0, 1\} . \end{aligned} \tag{6}$$

where U is an upper bound on the slack variables $\boldsymbol{s}$ and $\boldsymbol{Y}$ are the additional binary variables introduced. If constraint j is active, ($Y_j = 1$), and if inactive, ($Y_j = 0$). Note that now the integer variable set includes the binary variables $\boldsymbol{Y}$ in addition to the outer problem integer variables.

Transformed BLPP Global Optimization

The problem that results after the reformulated/linearized inner problem is replaced with its KKT optimality conditions and active set strategy is applied can still have nonlinear terms due to complementarity and stationarity conditions. Further, nonlinear terms in the outer problem variables may exist in either the inner or outer problem constraints. Hence, the resulting problem is a mixed-integer (nonlinear) optimization problem and should be solved by a global optimization procedure. If the integer variables are all binary and only appear in linear or mixed-bilinear terms, the Special structure MINLP-αBB, SMIN-αBB [1,3] approach is employed. If the outer integer variables are not restricted to binary and/or participate in nonlinear terms, the General structure MINLP-αBB, GMIN-αBB [1,3] approach is employed. The steps of the proposed framework are given below.

Global Optimization Algorithm

Step 1 Establish variable bounds by solving the problems:

$\boldsymbol{y}^{\mathrm{L}}, \boldsymbol{y}^{\mathrm{U}} = \min\ \boldsymbol{y}, -\boldsymbol{y}$ s.t. inner problem constraint setprotect
to obtain simple lower and upper bounds on $\boldsymbol{y}$,
$\boldsymbol{y}^{\mathrm{L}} \leq \boldsymbol{y} \leq \boldsymbol{y}^{\mathrm{U}}$.

Step 2 If the inner integer variables are integer, convert into a set of binary variables by Eq. (2) and Eq. (3).
Step 3 Obtain the vertex polyhedral convex envelope of the inner problem feasible region via the reformulation/ linearization [11]. The inner problem is now linear in both inner binary and continuous variables and parametric in outer problem variables, $\boldsymbol{x}$.
Step 4 Replace the inner problem with the set of equations that define its necessary and sufficient KKT optimality conditions. The resulting problem is single level.
Step 5 Solve the resulting single level optimization problem to global optimality. The inner integer variables are all separable, linear and binary at the beginning of this step. If the final problem is a Mixed Integer Linear Problem, (MILP), then use CPLEX. Notice that the problem will be an MILP only for the simplest cases. If there are nonlinear continuous variables, but the integer variables are all binary, linear and separable, use SMIN-αBB [1,3] global optimization procedure. If the outer problem has nonlinear integer terms, then use GMIN-αBB [1,3] global optimization procedure.

Illustrative Example

The following problem [10] can not be solved to global optimality using current deterministic solution approaches for integer bilevel programming problems in the literature.

$$\begin{aligned}
\min\ & -\left(-\frac{2}{5}x_1^2x_2 + 4x_2^2\right) y_1 y_2 \\
& -\left(-x_2^3 + 3x_1^2x_2\right)(1-y_1)y_2 - \left(2x_2^2 - x_1\right)(1-y_2) \\
\text{s.t. } & \min -\left(x_1x_2^2 + 8x_2^3 - 14x_1^2 - 5x_1\right) y_1 y_2 \\
& \quad -\left(-x_1x_2^2 + 5x_1x_2 + 4x_2\right)(1-y_1)y_2 \\
& \quad - 8x_1 y_1(1-y_2) \\
& \text{s.t. } y_1 + y_2 \geq 1 \\
& \quad 0 \leq x_1 \leq 10 \\
& \quad 0 \leq x_2 \leq 10 \\
& \quad y_1, y_2 \in \{0,1\}^2\,, \quad x \in \Re\,.
\end{aligned} \tag{7}$$

Steps 1–2 Variable bounds are already given in this problem, with $0 \leq x \leq 10$, and y_1 and y_2 are defined as binary.
Step 3 Determination of the vertex polyhedral convex hull: The inner problem is $Ny = 2$ degrees, second degree factors y_1y_2, $y_1(1-y_2)$, $(1-y_1)y_2$, $(1-y_1)(1-y_2)$ multiply the inner problem constraint $y_1 + y_2 - 1 \geq 0$ and result in:

$$\begin{aligned}
y_1y_2 &\geq 0 \\
y_1 + y_2 - y_1y_2 - 1 &\geq 0\,.
\end{aligned} \tag{8}$$

Linearization: Assign a new variable for the bilinear term $z_{12} = y_1y_2$ that leads to the additional constraints:

$$\begin{aligned} z_{12} &\geq 0 \\ y_1 - z_{12} &\geq 0 \\ y_2 - z_{12} &\geq 0 \\ -y_1 - y_2 + z_{12} &\geq -1 \,. \end{aligned} \tag{9}$$

From Eqs. (8) and (9), $y_1 + y_2 - z_{12} - 1 = 0$. Substituting the definition of z_{12} into Eq. (8), a linear relaxation of the inner problem constraint set leads to the original set of constraints:

$$\begin{aligned} y_1 + y_2 - 1 &\geq 0 \\ 1 - y_2 &\geq 0 \\ 1 - y1 &\geq 0 \,. \end{aligned} \tag{10}$$

Hence, the continuous relaxation of the original problem constraints define the convex hull and no additional constraints are necessary. The inner problem is continuous and linear in y_1 and y_2, and parametric in the outer problem variables $\boldsymbol{x}$.

Step 4 Replace the relaxed inner problem with the equivalent set of equations that define its necessary and sufficient KKT optimality conditions:

$$\begin{aligned} &\min \left(\frac{17}{5}x_1^2x_2 - 4x_2^2 - x_2^3\right) y_1 + \left(\frac{2}{5}x_1^2x_2 - 2x_2^2 - x_1\right) \\ &\cdot y_2 \left(-\frac{17}{5}x_1^2x_2 + 2x_2^2 + x_2^3 + x_1\right) \\ &0 \leq x_1 \leq 10 \\ &0 \leq x_2 \leq 10 \\ &-x_1^2x_2^2 - 8x_2^3 + 14x_1^2 + 5x_1 - x_1x_2^2 + 5x_1x_2 + 4x_2 \\ &\qquad - \lambda_1 + \lambda_2 = 0 \\ &-\lambda_1 - x_1^2x_2^2 - 8x_2^3 + 14x_1^2 + 13x_1 + \lambda_3 = 0 \\ &-y_1 - y_2 + 1 + s_1 = 0 \\ &y_1 + s_2 = 1 \\ &y_2 + s_3 = 1 \\ &\lambda_1 - UY \leq 0 \\ &s_1 + UY \leq U \\ &Y \in \{0,1\}; \quad s_1, \lambda_1, y_1, y_2 \geq 0; \\ &x_1, x_2 \in \Re; \quad y_1, y_2 \leq 1 \,. \end{aligned} \tag{11}$$

Step 5 The single level problem constraint contains the following nonlinear terms: $x_1^2x_2y_1$, $x_2^2y_1$, $x_1^2x_2y_2$, $x_2^2y_2$, x_1y_2, $x_1^2x_2$, ${x_2}^2$, ${x_2}^3$, $x_1x_2^2$, x_1x_2, $x_1^2x_2^2$ that should be underestimated. All integer variables are binary, linear and separable. Solve the resulting single level problem to global optimality using SMIN-αBB [1,3]. The global optimal solution reported in [10] is at $(x_1^*, x_2^*, y_1^*, y_2^*) = (6.038, 2.957, 0, 1)$. We identify the lower global solution at (0, 10, 1, 1). Note that the solution of this problem by enumeration methods could be labor intensive due to the presence of continuous variables.

Conclusions

The global optimization framework addresses the solution of several classes of mixed integer nonlinear bilevel optimization problems. The outer problem may be mixed-integer nonlinear in both inner and outer variables; the inner problem may be mixed-integer nonlinear in outer variables, linear, polynomial or multilinear in inner integer variables and linear in inner continuous variables. This is based on the reformulation of the mixed-integer inner problem feasible space to generate its convex hull, where the vertices correspond to binary solutions. This allows the equivalence of the inner optimization problem to the set of equations that define its KKT optimality conditions, with which it is replaced. The resulting single level optimization problem is solved to global optimality. This is arguably the first deterministic global optimization technique that can solve several classes of mixed-integer nonlinear bilevel optimization problems. Note that if the central decision maker wants to locate the second-best inner or outer integer solutions, simple integer cuts [2] can be added prior to applying the relevant solution strategy.

References

1. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, αbb, for general twice-differentiable constrained nlps i. theoretical advances. Comput Chem Eng 22:1137–1158
2. Floudas CA (1995) Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. In: Topics in Chemical Engineering. Oxford University Press, New York
3. Floudas CA (2000) Deterministic Global Optimization: Theory, Methods and Applications. Nonconvex Optimization and its Applications, vol 37. Kluwer, Dordrecht
4. Floudas CA, Gümüş ZH, Ierapetritou M (2001) Global optimization in design under uncertainty: feasibility test and flexibility index problems. Ind Eng Chem Res 40(20): 4267–82
5. Grossmann IE, Floudas CA (1987) Active constraint strategy for flexibility analysis in chemical processes. Comp Chem Eng 11(6):675

6. Gümüş ZH, Ciric AR (1997) Reactive distillation column design with vapor/liquid/liquid equilibria. Comp Chem Eng 21(S):S983-S988
7. Gümüş ZH, Floudas CA (2001) Global optimization of nonlinear bilevel optimization problems. J Glob Optim 20:1–31
8. Gümüş ZH, Floudas CA (2005) Global optimization of mixed-integer bilevel programming problems. Comp Man Sci 2:181–212
9. Moore JT, Bard JF (1990) The mixed integer linear bilevel programming problem. Oper Res 38:911
10. Sahin KH, Ciric AR (1998) A dual temperature simulated annealing approach for solving bilevel programming problems. Comp Chem Eng 23:11–25
11. Sherali HD, Adams WP (1990) A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM J Discret Math 3: 411–430
12. Sherali HD, Adams WP (1994) A hierarchy of relaxations and convex-hull characterizations for mixed-integer zero-one programming problems. Discret Appl Math 52:83–106

Mixed-Integer Nonlinear Optimization: A Disjunctive Cutting Plane Approach

YUSHAN ZHU
Department of Chemical Engineering,
Tsinghua University, Beijing, China

MSC2000: 49M37, 90C11

Article Outline

Keywords

Mixed-integer nonlinear programming; Branch-and-cut; Disjunctive cutting plane; Lift-and-project; MINLP

Introduction

The mixed-integer nonlinear programming (MINLP) approach was widely used to model and solve the process synthesis problems in chemical engineering filed during the last two decades within the superstructure framework that always involves discrete and continuous variables [3,4,5,6,10]. Recently, the successful employment of the branch-and-cut method for 0–1 integer programming [7,8] and 0–1 mixed-integer linear programming [1,2] has spurred great interest in its application for 0–1 mixed-integer nonlinear optimization due to the significant progress of interior point algorithm for convex optimization problems. Stubbs and Mehrotra [9] generalized the lift-and-project cut or the disjunctive cut for 0–1 integer or mixed-integer linear programming proposed in [1,2,7,8], and extended their method into a branch-and-cut algorithm for the 0–1 mixed-integer nonlinear optimization problem. The disjunctive cutting plane presented by Stubbs and Mehrotra [9] is obtained by solving a convex projection problem, so it is computationally expensive. In [11], a valid disjunctive cutting plane for mixed-integer nonlinear optimization problems was constructed by solving a linear programming problem implemented in an algorithmic package named MINO, i. e., Mixed-Integer Nonlinear Optimizer.

Formulation

The general 0–1 mixed-integer nonlinear optimization problems can be formulated as

$$(P)\left|\begin{array}{ll}\min\limits_{x,y} & dx\\ s.t. & Ax+Gy\le b\\ & g_i(x,y)\le 0,\quad i=1,\dots,l\\ & x\in\Re^n,\quad y\in\{0,1\}^q\end{array}\right.$$

where the constant vectors and matrices are defined as $d\in\Re^n, A\in\Re^{m\times n}, G\in\Re^{m\times q}, b\in\Re^m$. Let the feasible region of the standard continuous relaxation of problem (P) be defined as

$$C=\left\{(x,y)\in\Re^{n+q}\;\middle|\;\begin{array}{l}Ax+Gy\le b,\\ g_i(x,y)\le 0,\quad i=1,\dots,l,\\ 0\le y\le 1,\end{array}\right\}$$

Hence, the feasible set of (P) can be formulated as

$$C^0=\left\{(x,y)\in C:\; y_j\in\{0,\,1\},\quad j=1,\dots,q\right\}.$$

At a generic step of the branch-and-cut algorithm, let $(\overline{x}, \overline{y})$ be a solution to the current NLP relaxation of (P). If any of the components of the binary variables $\overline{y}$ are not in $\{0, 1\}$, we can add a valid inequality into the current feasible set such that this inequality is violated by $(\overline{x}, \overline{y})$. We denote by c the family of inequalities to describe the current feasible set and the newly incorporated inequalities. We denote by $F_0, F_1 \subseteq \{1, \dots, q\}$ the sets of binary variables that have been fixed at 0 and 1, respectively. Let

$$K(C, F_0, F_1) = \left\{ (x, y) \in C \;\middle|\; \begin{array}{l} y_j = 0 \text{ for} j \in F_0 \\ y_j = 1 \text{ for} j \in F_1 \end{array} \right\}$$

And let $NLP(C, F_0, F_1)$ denote the nonlinear program

$$\begin{array}{ll} \min\limits_{x,y} & dx \\ \text{s.t.} & (x, y) \in K(C, F_0, F_1) \end{array}$$

The active nodes of the enumeration tree are represented by a list of S with ordered pairs (F_0, F_1). Let UBD represent the current upper bound, i. e., the value of the best-known solution to problem (P).

Branch-And-Cut Procedure

Input of d, n, q, A, G, b, g_i $(i = 1, \dots, l)$:

(1) Initialization. Set $S = \{(F_0 = \phi, F_1 = \phi)\}$, and let C consist of the nonlinear programming relaxation of (P) and $UBD = \infty$.

(2) Node Selection. If $S = \phi$, stop. Otherwise, choose an ordered pair $(F_0, F_1) \in S$ and remove it from S.

(3) Lower Bounding Step. Solve the nonlinear program $NLP(C, F_0, F_1)$. If the problem is infeasible, go to *Step 2*. Otherwise, let $(\overline{x}, \overline{y})$ denote its optimal solution. If $d\overline{x} \geq UBD$, go to *Step 2*. If $\overline{y}_j \in \{0,\ 1\}$, $j = 1, \dots, q$, let $(x^*, y^*) = (\overline{x}, \overline{y})$, $UBD = d\overline{x}$, and go to *Step 2*.

(4) Branching versus cutting decision. Should cutting planes be generated? If yes, go to *Step 5*, else go to *Step 6*.

(5) Cut generation. Generate cutting plane $\alpha x + \beta y \leq \gamma$ valid for (P) but violated by $(\overline{x}, \overline{y})$. Add the cuts into C and go to *Step 3*.

(6) Branching Step. Pick an index $j \in \{1, \dots, q\}$ such that $0 < \overline{y}_j < 1$. Generate the subproblems corresponding to $(F_0 \cup \{j\}, F_1)$ and $(F_0, F_1 \cup \{j\})$, add them into the node set S. Go to *Step 2*.

When the algorithm terminates, if $UBD < \infty$, (x^*, y^*) is an optimal solution to (P), otherwise (P) is infeasible.

Linear Approximation of NLP Relaxation

The continuous relaxation of problem (P) at some node in an enumeration tree can be described by

$$(NLP)\left|\begin{array}{l} \min\limits_{x,y}\ dx \\ s.t.\ \tilde{A}x + \tilde{G}y \leq \tilde{b}, \\ \quad g_i(x, y) \leq 0, \quad i = 1, \dots, l, \\ \quad y_j = 0, \quad j \in F_0, \\ \quad y_j = 1, \quad j \in F_1, \\ \quad (x, y) \in \Re^{n+q}, \end{array}\right.$$

where the reformulated linear constraint set consists of the original linear constraint set and the upper and lower bound constraints for binary variables, so we have $\tilde{A} \in \Re^{(m+2q)\times n}$, $\tilde{G} \in \Re^{(m+2q)\times q}$, $\tilde{b} \in \Re^{m+2q}$. Assume that the above NLP continuous problem is feasible and has a finite minimum at $(\overline{x}, \overline{y})$, since otherwise the node is done. A linear approximation problem at $(\overline{x}, \overline{y})$ for the above NLP problem can be obtained by

$$(LP)\left|\begin{array}{l} \min\limits_{x,y}\ dx \\ s.t.\ \tilde{A}x + \tilde{G}y \leq \tilde{b}, \\ \quad g_i(\overline{x}, \overline{y}) + \nabla g_i(\overline{x}, \overline{y}) \begin{pmatrix} x - \overline{x} \\ y - \overline{y} \end{pmatrix} \leq 0, \\ \quad i = 1, \dots, l, \\ \quad y_j = 0, \quad j \in F_0, \\ \quad y_j = 1, \quad j \in F_1, \\ \quad (x, y) \in \Re^{n+q}, \end{array}\right.$$

where the original convex and differentiable functions are replaced by their first-order Taylor approximation at $(\overline{x}, \overline{y})$. Accordingly, a MILP problem corresponding to the MINLP problem at the current node can be described by

$$(MILP)\left|\begin{array}{l} \min\limits_{x,y}\ dx \\ s.t.\ \tilde{A}x + \tilde{G}y \leq \tilde{b}, \\ \quad g_i(\overline{x}, \overline{y}) + \nabla g_i(\overline{x}, \overline{y}) \begin{pmatrix} x - \overline{x} \\ y - \overline{y} \end{pmatrix} \leq 0, \\ \quad i = 1, \dots, l, \\ \quad y_j = 0, \quad j \in F_0, \\ \quad y_j = 1, \quad j \in F_1, \\ \quad x \in \Re^n, y \in \{0, 1\}^q, \end{array}\right.$$

[11] proved that if the above NLP achieves its optimal solution at $(\overline{x}, \overline{y})$. Then, $(\overline{x}, \overline{y})$ is also an optimal solution to the aforementioned LP. The geometrical explanation of the above linear approximation is presented in Fig. 1, and it is obvious that the mixed-integer set is expanded after linear approximation.

Disjunctive Cut Generation

For problem (P), it is very attractive to construct the lift-and-project cut in terms of the approximated LP instead of the NLP, but the cut still can cut away the fraction point $(\overline{x}, \overline{y})$. Such cut can be derived by imposing the 0–1 integral condition on a binary variable y_j while $0 < \overline{y}_j < 1$. In Fig. 1, the short dashed line represents the cut generated directly by using the convex hull of the mixed-integer convex set presented by [9], and the long dashed line stands for the cut to be generated in [11] based on the linear approximation. For cut generation, the node sets F_0 and F_1 can be expanded to include additional binary variables whose optimal solutions are taken at 0 or 1 for the NLP problem at the current node. Then, we redefine these two sets as $\overline{F}_0 = \{i : \overline{y}_i = 0\}$ and $\overline{F}_1 = \{i : \overline{y}_i = 1\}$. It is not difficult to verify that the above NLP and LP problems have the same optimal solutions if we change the original node sets to be the expanded ones. Let the feasible region of the above LP be defined as

$$K = K\left(C, \overline{F}_0, \overline{F}_1\right) = \left\{ (x, y) \in \Re^{n+q} \;\middle|\; \begin{array}{l} \overline{A}x + \overline{G}y \le \overline{b} \\ y_i = 0, i \in \overline{F}_0 \\ y_i = 1, i \in \overline{F}_1 \end{array} \right\}$$

where $\overline{A} \in \Re^{(m+2q+l)\times n}, \overline{G} \in \Re^{(m+2q+l)\times q}, \overline{b} \in \Re^{m+2q+l}$, i. e., the newly reformulated linear constraint set consists of the linear approximation set besides the original one, as

$$\overline{A}x + \overline{G}y \le \overline{b} \quad \equiv \left\{ \begin{array}{l} Ax + Gy \le b, \\ \nabla g^x\left(\overline{x}, \overline{y}\right) x + \nabla g^y\left(\overline{x}, \overline{y}\right) y \\ \le \nabla g\left(\overline{x}, \overline{y}\right) \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix} - g\left(\overline{x}, \overline{y}\right), \\ -y_i \le 0 \quad i = 1, \ldots, q, \\ y_i \le 1 \quad i = 1, \ldots, q\,, \end{array} \right.$$

It should be noted that the node sets have already been changed to the expanded ones in the above formulation. If we impose the integrality condition on a binary variable y_j for which $0 < \overline{y}_j < 1$, the disjunctive cut can be obtained by choosing a valid inequality for

$$P_j\left(K\right) = conv\left(K \cap \left\{(x, y) \in \Re^{n+q} : y_j \in \{0, 1\}\right\}\right) \;,$$

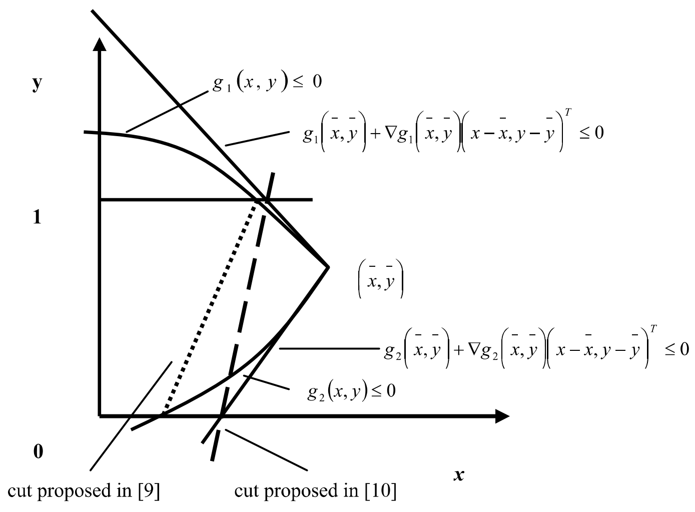
The convex hull of this union set can be further described by its disjunctive form as

$$P_j(K) = conv\Bigg(\left\{K \cap \left\{(x, y) \in \Re^{n+q} : y_j \le 0\right\}\right\} \cup \left\{K \cap \left\{(x, y) \in \Re^{n+q} : -y_j \le -1\right\}\right\}\Bigg)\,,$$

Let $F = \{1, \ldots, q\} \setminus \left(\overline{F}_0 \cup \overline{F}_1\right)$ denote the set of free variables at node $\left(\overline{F}_0, \overline{F}_1\right)$, and the vector corresponding to those free variables can be defined as $y^F = y \setminus \left\{y_i : i \in \left(\overline{F}_0 \cup \overline{F}_1\right)\right\}$. The columns of matrix $\overline{G}$ corresponding to the fixed binary variables can be removed from the constraint set by defining $\overline{G}^F = \overline{G} \setminus \left\{\overline{G}_i : i \in \left(\overline{F}_0 \cup \overline{F}_1\right)\right\}$, and the right-hand side can be calculated accordingly as $\overline{b}^F = \overline{b} - \sum_{i \in F_1} \overline{G}_i$. Finally, the rows in matrices $\overline{A}$ and $\overline{G}^F$, and vector $\overline{b}^F$ that correspond to the upper and lower bounds of the fixed binary variables are removed. After doing the above operations, we can assume without loss of generality, that $F_1 = \phi$. Since if $F_1 \neq \phi$, all the variables y_k in F_1 can be complemented by 1-y_k which amounts replacing the columns G_k and the right-hand side with $-G_k$ and $b - G_k$, respectively. The reduced LP constraint set after removing the fixed binary variables becomes

$$\overline{A}^F x + \overline{G}^F y^F \le \overline{b}^F \equiv \left\{ \begin{array}{l} Ax + \sum\limits_{i \in F} G_j y_j \le b \\ - \sum\limits_{i \in F_1} G_j, \\ \nabla g^x\left(\overline{x}, \overline{y}\right) x \\ + \sum\limits_{i \in F} \nabla g_i^y\left(\overline{x}, \overline{y}\right) y_j \\ \le \nabla g\left(\overline{x}, \overline{y}\right) \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix} \\ -g\left(\overline{x}, \overline{y}\right) \\ - \sum\limits_{i \in F_1} \nabla g_i^y\left(\overline{x}, \overline{y}\right), \\ -y_i \le 0 \quad i \in F, \\ y_i \le 1 \quad i \in F\,, \end{array} \right.$$

where $\overline{A}^F \in \Re^{(m+2|F|+l)\times n}$, $\overline{G}^F \in \Re^{(m+2|F|+l)\times |F|}$, $\overline{b}^F \in \Re^{m+2|F|+l}$. Then, the feasible region of the above


y
1
0
x
$g_1(x,y) \le 0$
$g_1(\bar{x},\bar{y}) + \nabla g_1(\bar{x},\bar{y})(x-\bar{x}, y-\bar{y})^T \le 0$
$(\bar{x},\bar{y})$
$g_2(\bar{x},\bar{y}) + \nabla g_2(\bar{x},\bar{y})(x-\bar{x}, y-\bar{y})^T \le 0$
$g_2(x,y) \le 0$
cut proposed in [9]
cut proposed in [10]

Mixed-Integer Nonlinear Optimization: A Disjunctive Cutting Plane Approach, Figure 1
Linear approximation of the mixed-integer nonlinear optimization problem, where the mixed-integer convex set is described by a continuous variable and a binary variable

LP can be reformulated as

$$K = \left\{ (x, y^F) \in \Re^{n+|F|} \,\middle|\, \overline{A}^F x + \overline{G}^F y^F \le \overline{b}^F \right\}$$

Let $(\overline{x}, \overline{y})$ be the optimal solution when solving $NLP\left(C, \overline{F}_0, \overline{F}_1\right)$. First, we assume that $(\overline{x}, \overline{y})$ is not feasible to problem (P), and let j be the binary variable index such that $0 < \overline{y}_j < 1$. In [11], a disjunctive cut generation linear programming is given by

$$(LP(F)) \left| \begin{array}{l} \min \sum_{i \in N} z_i + \sum_{i \in F} w_i \\ s.t.\ x = u_0 + u_1,\ y^F = v_0 + v_1, \\ \quad \overline{A}^F u_0 + \overline{G}^F v_0 - \overline{b}^F \lambda_0 \le 0, \\ \quad v_{0,j} \le 0, \\ \quad \overline{A}^F u_1 + \overline{G}^F v_1 - \overline{b}^F \lambda_1 \le 0, \\ \quad -v_{1,j} \le -\lambda_1, \\ \quad \lambda_0 + \lambda_1 = 1; \\ \quad -z + x \le \overline{x}, \quad -z - x \le -\overline{x}, \\ \quad -w + y^F \le \overline{y}^F, \quad -w - y^F \le -\overline{y}^F, \\ \quad \lambda_0, \lambda_1 \ge 0, \quad x, u_0, u_1, z \in \Re^n, \\ \quad y^F, v_0, v_1, w \in \Re^{|F|} . \end{array} \right.$$

This linear program has $4n + 4|F| + 2$ variables with $2m + 3n + 7|F| + 2l + 3$ equality or inequality constraints. After solving this linear program, we get its solutions denoted by $(\tilde{x}, \tilde{y}^F, \tilde{u}_0, \tilde{v}_0, \tilde{u}_1, \tilde{v}_1, \tilde{z}, \tilde{w}, \tilde{\lambda}_0, \tilde{\lambda}_1)$ as well as the dual multipliers. Denote by $\pi_x^F, \pi_y^F, \delta_\lambda^F$ the multipliers for the equality constraints, δ_0^F, δ_1^F for the disjunctive inequality constraints, μ_0^F, μ_1^F for the inequality original constraints, and $\varepsilon_+^F, \varepsilon_-^F, \varphi_+^F, \varphi_-^F$ for the additional constraints in LP(F), the cut generated in Theorem 2, i. e. $\alpha x + \beta^F y^F \le \gamma$, can be reformulated by the dual multipliers and the primal solutions to LP(F), as $\pi_x^F x + \pi_y^F y^F \le \pi_x^F \tilde{x} + \pi_y^F \tilde{y}^F$. Since the feasible set of problem (P) at the node (F_0, F_1) is contained in the feasible set of the MILP at that node. Therefore, the inequality $\alpha x + \beta^F y^F \le \gamma$ is valid and proper for problem (P) at the current node denoted by (F_0, F_1), and its descendents where the variables in (F_0, F_1) remain fixed.

Cut Lifting

An important advantage of the cut generated by the lift-and-project technology is that the multipliers, i. e., $\mu_0^F, \delta_0^F, \mu_1^F, \delta_1^F$, obtained along with the solution $(\tilde{x}, \tilde{y}^F)$ by solving LP(F) can be used to calculate the closed form expressions of the coefficients β_i for the binary variables in the index set $F_0 \cup F_1$. First, we lift the inequality obtained at the current node into the complemented original space of the MILP problem, that is

$$\left\{ (x, y) \in \Re^{n+q} : \left| \begin{array}{l} \overline{A}x + \overline{G}^Q y \le \overline{b}^Q, \\ y \in \{0,1\}^q \end{array} \right. \right\}$$

Let $j \in \{1, \ldots, q\}$ be an index such that $0 < \overline{y}_j < 1$ and consider the inequality $\alpha^q x + \beta^q y \le \gamma^q$ generated over the complemented original space of the MILP problem, this is to solve the linear program LP(Q), as

$$
(LP(Q))\left|\begin{array}{l}
\min \sum_{i\in N} z_i + \sum_{i\in Q} w_i \\
s.t.\ x = u_0 + u_1, \quad y = v_0 + v_1, \\
\quad \overline{A}u_0 + \overline{G}^Q v_0 - \overline{b}^Q \lambda_0 \le 0, \\
\quad v_{0,j} \le 0, \\
\quad \overline{A}u_1 + \overline{G}^Q v_1 - \overline{b}^Q \lambda_1 \le 0, \\
\quad -v_{1,j} \le -\lambda_1, \\
\quad \lambda_0 + \lambda_1 = 1, \\
\quad -z + x \le -\overline{x}, \quad -z - x \le -\overline{x}, \\
\quad -w + y \le \overline{y}, \quad -w - y \le -\overline{y}, \\
\quad \lambda_0, \lambda_1 \ge 0, \\
\quad x, u_0, u_1, z \in \Re^n, y, v_0, v_1, w \in \Re^q .
\end{array}\right.
$$

Note that more variables and constraints are added into this linear program compared with LP(F). But, by using the solutions to LP(F) and its multipliers, we can obtain the optimal solution to the above LP(Q). Let $\left(\hat{x}, \hat{y}, \hat{u}_0, \hat{v}_0, \hat{u}_1, \hat{v}_1, \hat{z}, \hat{w}, \hat{\lambda}_0, \hat{\lambda}_1\right)$be the solution to the linear program LP(Q), which can be constructed by the solution to the LP(F), as $\tilde{x} = \hat{x}$, $\hat{y} = (\tilde{y}^F, 0)$, $\hat{u}_0 = \tilde{u}_0$, $\hat{u}_1 = \tilde{u}_1$, $\hat{v}_0 = (\tilde{v}_0, 0)$, $\hat{v}_1 = (\tilde{v}_1, 0)$, $\hat{\lambda}_0 = \tilde{\lambda}_0$, $\hat{\lambda}_1 = \tilde{\lambda}_1$, $\hat{z} = \tilde{z}$, and $\hat{w} = (\tilde{w}, 0)$. Then, the corresponding dual multipliers of LP(Q) denoted by $(\hat{\pi}_x^Q, \hat{\pi}_y^Q, \hat{\mu}_0^Q, \hat{\delta}_0^Q, \hat{\mu}_1^Q, \hat{\delta}_1^Q, \hat{\delta}_\lambda^Q, \hat{\varepsilon}_+^Q, \hat{\varepsilon}_-^Q, \hat{\varphi}_+^Q, \hat{\varphi}_-^Q)$, which is also the solution to the dual linear program DLP(Q), can be constructed by those to DLP(F), as $\hat{\pi}_x^Q = \tilde{\pi}_x^F$, $\hat{\pi}_{y,i}^Q = \tilde{\pi}_{y,i}^F$ for $i \in F$, $\hat{\pi}_{y,i}^Q = \min\{\tilde{\mu}_0^F \overline{G}_i^Q, \tilde{\mu}_1^F \overline{G}_i^Q\}$ for $i \in F_0 \cup F_1$, $\hat{\delta}_0^Q = \tilde{\delta}_0^F$, $\hat{\delta}_1^Q = \tilde{\delta}_1^F$, $\hat{\delta}_\lambda^Q = \tilde{\delta}_\lambda^F$, $\hat{\mu}_0^Q = (\tilde{\mu}_0^F, 0)$, $\hat{\mu}_1^Q = (\tilde{\mu}_1^F, 0)$, $\hat{\varepsilon}_+^Q = \tilde{\varepsilon}_+^Q$, $\hat{\varepsilon}_-^Q = \tilde{\varepsilon}_-^Q$, $\hat{\varphi}_+^Q = (\tilde{\varphi}_+^F, 0)$, and $\hat{\varphi}_-^Q = \left(\tilde{\varphi}_-^F, 0\right)$. The inequality $\alpha^q x + \beta^q y \le \gamma^q$ described by $\hat{\pi}_x^Q x + \hat{\pi}_y^Q y \le \hat{\pi}_x^Q \hat{x} + \hat{\pi}_y^Q \hat{y}$ is valid for the entire enumeration tree, and cuts away $\left(\overline{x}, \overline{y}\right)$.

Conclusion

A branch-and-cut algorithm is introduced in this section to solve 0–1 mixed-integer nonlinear optimization problem where the disjunctive cuts are generated and incorporated into an enumeration process. The lift-and-project cut generation is performed via linear programming, as opposed to the convex nonlinear approach used in [9]. This new approach has the advantage of making the cut generation computationally cheaper and overcoming the nondifferential problems.

References

1. Balas E, Ceria S, Cornuejols G (1993) A lift-and-project cutting plane algorithm for mixed-integer 0–1 programs. Math Program 58:295–324
2. Balas E, Ceria S Cornuejols G (1996) Mixed 0–1 programming by lift-and-project in a branch-and-cut framework. Manag Sci 42:1229–1246
3. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307–339
4. Floudas CA (1995) Nonlinear and mixed-integer optimization, fundamentals and applications. Oxford Univ Press, New York
5. Floudas CA (2005) Research challenges, opportunities and synergism in systems engineering and computational biology. AIChE J 51:1872–1884
6. Grossman IE, Westerberg AW (2000) Research challenges in process systems engineering. AIChE J 46:1700–1703
7. Lovasz L, Schrijver A (1991) Cones of matrices and set functions and 0–1 optimization. SIAM J Optim 1:166–190
8. Sherali H, Adams W (1990) A hierarchy of relaxations between the continuous and convex hull representation for zero-one programming problems. SIAM J Discret Math 3:411–430
9. Stubbs RA, Mehrotra S (1999) A branch-and-cut method for 0–1 mixed convex programming. Math Program 86:515–532
10. Zhu Y, Kuno T (2003) Global optimization of nonconvex MINLP by a hybrid branch-and-bound and revised general Benders decomposition method. Ind Eng Chem Res 42:528–539
11. Zhu Y, Kuno T (2006) A disjunctive cutting plane based branch-and-cut algorithm for 0–1 mixed-integer nonlinear programs. Ind Eng Chem Res 45:187–196

Mixed Integer Nonlinear Programming

MINLP

CHRISTODOULOS A. FLOUDAS
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90C11, 49M37

Article Outline

Keywords

Decomposition; Outer approximation; Branch and bound; Global optimization

A wide range of nonlinear optimization problems involve integer or discrete variables in addition to the continuous variables. These classes of optimization problems arise from a variety of applications and are denoted as *mixed integer nonlinear programming* MINLP problems.

The integer variables can be used to model, for instance, sequences of events, alternative candidates, existence or non-existence of units (in their zero-one representation), while discrete variables can model, for instance, different equipment sizes. The continuous variables are used to model the input-output and interaction relationships among individual units/operations and different interconnected systems.

The nonlinear nature of these mixed integer optimization problems may arise from:

i) nonlinear relations in the integer domain exclusively (e. g., products of binary variables in the quadratic assignment model);
ii) nonlinear relations in the continuous domain only (e. g., complex nonlinear input-output model in a distillation column or reactor unit);
iii) nonlinear relations in the joint integer-continuous domain (e. g., products of continuous and binary variables in the scheduling/planning of batch processes and retrofit of heat recovery systems).

The book [88] studies mixed integer linear optimization and combinatorial optimization, while the [40] studies mixed integer nonlinear optimization problems.

The coupling of the integer domain and the continuous domain with their associated nonlinearities make the class of MINLP problems very challenging from the theoretical, algorithmic, and computational point of view. Mixed integer nonlinear optimization problems are encountered in a variety of applications in all branches of engineering and applied science, applied mathematics, and operations research. These represent very important and active research areas that include:

- *process synthesis*
  - heat exchanger networks
  - retrofit of heat recovery systems
  - distillation sequencing
  - mass exchange networks
  - reactor-based systems
  - reactor-separator-recycle systems
  - utility systems
  - total process systems
  - metabolic engineering
- *process design*
  - reactive distillation
  - design of dynamic systems
  - plant layout
  - environmental design
- process synthesis and *design under uncertainty*
  - uncertainty analysis
  - dynamic systems
  - batch plant design
- *molecular design*
  - solvent selection
  - design of polymers and refrigerants
  - property prediction under uncertainty
- *interaction of design, synthesis and control*
  - steady state operation
  - dynamic operation
- *process operations*
  - scheduling of multiproduct plants
  - design and retrofit of multiproduct plants
  - synthesis, design and scheduling of multipurpose plants

  - planning under uncertainty
- *facility location and allocation*
- *facility planning and scheduling*
- *topology of transportation networks*

The applications in the area of *process synthesis* in chemical engineering include:

i) the synthesis of grassroot heat recovery networks [24,25,43,138,139,140];
ii) the retrofit of heat exchanger systems [25,95];
iii) the synthesis of distillation-based separation systems [8,9,90,102,104,131];
iv) the synthesis of mass exchange networks [54,99];
v) the synthesis of complex reactor networks [71,73,74,119];
vi) the synthesis of reactor-separator-recycle systems [72];
vii) the synthesis of utility systems [65];
viii) the synthesis of total process systems [28,29,68,69,75,76,98]; and
ix) the analysis and synthesis of metabolic pathways [30,58,59,107].

Reviews of the mixed integer nonlinear optimization frameworks and applications in Process Synthesis are provided in [40,49,50], and [7], while algorithmic advances for logic and global optimization in Process Synthesis are reviewed in [44].

The MINLP applications in the area of *process design* include:

i) reactive distillation processes [26];
ii) design of dynamic systems [11,14,117,118];
iii) plant layout systems [47,105]; and
iv) environmentally benign systems [27,123].

The MINLP applications in the area of *process synthesis and design under uncertainty* include:

i) deterministic and stochastic uncertainty analysis [1,33,51];
ii) design of dynamic systems under uncertainty [31,85]; and
iii) design of batch processes under uncertainty [57,63,108,109].

In the area of *molecular design*, the MINLP applications include:

i) the computer-aided molecular design aspects of selecting the best solvents [91];
ii) design of polymers and refrigerants [21,22,23,35,80,111,126]; and
iii) property prediction under uncertainty [81].

The MINLP applications in the area of *interaction of design, synthesis and control* include:

i) studies under steady state operation of chemical processes [78,79,96,97]; and
ii) studies under dynamic operation [85,86,118].

Applications of MINLP approaches have also emerged in the area of *process operations* and include:

i) short term scheduling of batch and semicontinuous processes [85,143];
ii) the design of multiproduct plants [17,18,53];
iii) the synthesis, design and scheduling of multipurpose plants [13,36,37,93,94,116,127,128,132,133,137]; and
iv) planning under uncertainty [62,63,64,77,106].

Reviews of the advances in the design, scheduling and planning of batch plants can be found in [52,113], while a collection of recent contributions can be found in the proceedings of the 1998 FOCAPO meeting.

MINLP applications received significant attention in other engineering disciplines. These include

i) the facility location in a multi-attribute space [45];
ii) the optimal unit allocation in an electric power system [16];
iii) the facility planning of an electric power generation [19,114];
iv) the chip layout and compaction [32];
v) the topology optimization of transportation networks [60]; and
vi) the optimal scheduling of thermal generating units [48].

Mathematical Description

The general algebraic MINLP formulation can be stated as:

$$\left\{\begin{array}{lll} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x},\mathbf{y}) & \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} & \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0} & \\ & \mathbf{x} \in \mathbf{X} \subseteq \mathbb{R}^n & \\ & \mathbf{y} \in \mathbf{Y} & \text{integer.} \end{array}\right. \tag{1}$$

Here $\mathbf{x}$ represents a vector of n continuous variables (e. g., flows, pressures, compositions, temperatures, sizes of units), and $\mathbf{y}$ is a vector of integer variables (e. g., alternative solvents or materials); $\mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0}$ denote the m equality constraints (e. g., mass, energy balances, equilibrium relationships); $\mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}$ are the p inequality constraints (e. g., specifications on purity of distillation products, environmental regulations, feasibility constraints in heat recovery systems, logical constraints); and $f(\mathbf{x}, \mathbf{y})$ is the objective function (e. g., annualized total cost, profit, thermodynamic criteria).

Remark 1 The integer variables $\mathbf{y}$ with given lower and upper bounds

$$\mathbf{y}^L \leq \mathbf{y} \leq \mathbf{y}^U$$

can be expressed through 0–1 variables (i. e., binary), denoted as $\mathbf{z}$, by the following formula:

$$y = y^L + z_1 + 2z_2 + 4z_3 + \cdots + 2^{N-1} z_N ,$$

where N is the minimum number of 0–1 variables needed. This minimum number is given by:

$$N = 1 + \mathrm{INT} \left\{ \frac{\log(y^U - y^L)}{\log 2} \right\} ,$$

where the INT function truncates its real argument to an integer value.

Then, formulation (1) can be written in terms of 0–1 variables:

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\ & \mathbf{x} \in \mathbf{X} \subseteq \mathbb{R}^n \\ & \mathbf{y} \in \{0, 1\}^q , \end{cases} \tag{2}$$

where $\mathbf{y}$ now is a vector of q 0–1 variables (e. g., existence of a process unit ($y_i = 1$) or nonexistence ($y_i = 0$)).

Challenges in MINLP

Dealing with *mixed integer nonlinear optimization* models of the form (1) or (2) present two major challenges. These difficulties are associated with the nature of the problem, namely, the combinatorial domain ($\mathbf{y}$-domain) and the continuous domain ($\mathbf{x}$-domain).

As the number of binary variables $\mathbf{y}$ in (2) increase, one is faced with a large combinatorial problem, and the complexity analysis results characterize MINLP problems as NP-complete [88]. At the same time, due to the nonlinearities the MINLP problems are in general nonconvex which implies the potential existence of multiple local solutions. The determination of a global solution of the *nonconvex MINLP* problems is also NP-hard, since even the global optimization of constrained nonlinear programming problems can be *NP*-hard [100], and even quadratic problems with one negative eigenvalue are *NP*-hard [101]. An excellent book on complexity issues for nonlinear optimization is [129].

Despite the aforementioned discouraging results from complexity analysis, which are worst-case results, significant progress has been achieved in the MINLP area from the theoretical, algorithmic, and computational perspective. As a result, several algorithms have been proposed for convex and nonconvex MINLP models, their convergence properties have been investigated, and a large number of applications now exist that cross the boundaries of several disciplines. In the sequel, we will discuss these developments.

Overview of Local Optimization Approaches for Convex MINLP Models

A representative collection of *local MINLP* algorithms developed for solving *convex MINLP* models of the form (1) or restricted classes of (2) includes the following:

1) *generalized Benders decomposition*, GBD, [42,46, 103];
2) *outer approximation*, OA, [34];
3) outer approximation with equality relaxation, OA/ER, [67];
4) outer approximation with equality relaxation and augmented penalty, OA/ER/AP, [131];
5) *generalized outer approximation*, GOA, [38];
6) *generalized cross decomposition*, GCD, [61];
7) *branch and bound*, BB, [15,20,39,55,92,110];
8) *feasibility approach*, FA, [82];
9) *extended cutting plane*, ECP, [134,135];
10) *logic-based approaches*, [124,130].

In the pioneering work [46] on the generalized benders decomposition, GBD, two sequences of updated upper(nonincreasing) and lower (nondecreasing) bounds are created that converge within ϵ in a finite number of iterations. The upper bounds correspond to solving subproblems in the $\mathbf{x}$ variables by fixing the $\mathbf{y}$ variables, while the lower bounds are based on duality theory.

The outer approximation, OA, addresses problems with nonlinear inequalities, and creates sequences of upper and lower bounds as the GBD, but it has the distinct feature of using primal information, that is the solution of the upper bound problems, so as to linearize the objective and constraints around that point. The lower bounds in OA are based upon the accumulation of the linearized objective function and constraints, around the generated primal solution points.

The OA/ER algorithm extends the OA to handle nonlinear equality constraints by relaxing them into inequalities according to the sign of their associated multipliers.

The OA/ER/AP algorithm introduces an augmented penalty function in the lower bound subproblems of the OA/ER approach.

The generalized outer approximation, GOA, extends the OA to the MINLP problems that the GBD addresses and introduces exact penalty functions.

The generalized cross decomposition, GCD, simultaneously utilizes primal and dual information by exploiting the advantages of Dantzig–Wolfe and generalized Benders decomposition.

An overview of these local MINLP algorithms and extensive theoretical, algorithmic, and applications of GBD, OA, OA/ER, OA/ER/AP, GOA, and GCD algorithms can be found in [40].

The branch and bound, BB, approaches start by solving the continuous relaxation of the MINLP and subsequently perform an implicit enumeration where a subset of the 0–1 variables is fixed at each node. The lower bound corresponds to the NLP solution at each node and it is used to expand on the node with the lowest lower bound or it is used to eliminate nodes if the lower bound exceeds the current upper bound. If the continuous relaxation, NLP in most cases with the exception of the algorithm of [110] where an LP problem is obtained, of the MINLP has a 0–1 solution for the $\mathbf{y}$ variables, then the BB algorithm will terminate at that node. With a similar argument, if a tight NLP relaxation results in the first node of the tree, then the number of nodes that would need to be eliminated can be low. However, loose NLP relaxations may result in having a large number of NLP subproblems to be solved. The algorithm terminates when the lowest lower bound is within a prespecified tolerance of the best upper bound.

The feasibility approach, FA, rounds the relaxed NLP solution to an integer solution with the least local degradation by successively forcing the superbasic variables to become nonbasic based on the reduced cost information. The premise of this approach is that the problems to be treated are sufficiently large so that techniques requiring the solution of several NLP relaxations, such as the branch and bound approach, have prohibitively large costs. They therefore wish to account for the presence of the integer variables in the formulation and solve the mixed integer problem directly. This is achieved by fixing most of the integer variables to one of their bounds (the *nonbasic variables*) and allowing the remaining small subset (the *basic variables*) to take discrete values in order to identify feasible solutions. After each iteration, the reduced costs of the variables in the nonbasic set are computed to measure their effect on the objective function. If a change causes the objective function to decrease, the appropriate variables are removed from the nonbasic set and allowed to vary for the next iteration. When no more improvement in the objective function is possible, the algorithm is terminated. This strategy leads to the identification of a local solution.

The *cutting plane* algorithm proposed in [66] for NLP problems has been extended to MINLPs [134,135]. The ECP algorithm relies on the linearization of one of the nonlinear constraints at each iteration and the solution of the increasingly tight MILP made up of these linearizations. The solution of the MILP problem provides a new point on which to base the choice of the constraint to be linearized for the next iteration of the algorithm. The ECP does not require the solution of any NLP problems for the generation of an upper bound. As a result, a large number of linearizations are required for the approximation of highly nonlinear problems and the algorithm does not perform well in such cases. Due to the use of linearizations, convergence to the global optimum solution is guaranteed only for problems involving inequality constraints which are convex in the $\mathbf{x}$ and relaxed $\mathbf{y}$-space.

An alternative to the direct solution of the MINLP problem was proposed by [124]. Their approach stems from the work of [70] on a modeling/decomposition strategy which avoids the zero-flows generated by the nonexistence of a unit in a process network. The first stage of the algorithm is the reformulation of the

MINLP into a generalized disjunctive program. A vector of Boolean variables indicate the status of a disjunction (True or False) and are associated with the alternatives. The set of disjunctions allows the representation of several alternatives. A set of logical relationships between the Boolean variables is introduced. Instead of resorting to binary variables within a single model, the disjunctions are used to generate a different model for each alternative. Since all continuous variables associated with the nonexisting alternatives are set to zero, this representation helps to reduce the size of the problems to be solved. Two algorithms are suggested by [124]. They are logic-based variants of the outer approximation and generalized Benders decomposition. [130] introduced LOGMIP, a computer code for disjunctive programming and MINLP problems, and studied modeling alternatives and process synthesis applications.

Overview of Global Optimization Approaches for Nonconvex MINLP Models

the previous Section we discussed local MINLP algorithms which are applicable to convex MINLP models. While identification of the global solution for convex problems can be guaranteed, a local solution is often obtained for nonconvex problems. The recent book by [41] discusses the theoretical, algorithmic and applications oriented advances in the global optimization of mixed integer nonlinear models. A number of *global MINLP* algorithms that have been developed to address different types of nonconvex MINLPs are presented in this section. These include:

1) *Branch and reduce* approach, [115];
2) *interval analysis* based approach, [125];
3) *extended cutting plane* approach, [135,136];
4) *reformulation/spatial branch and bound* approach, [121,122];
5) *hybrid branch and bound and outer approximation* approach, [141,142];
6) The *SMIN-αBB* approach, [2,4];
7) The *GMIN-αBB* approach, [2,4].

In the sequel, we will briefly discuss the approaches 1)–7).

Branch and Reduce Algorithm

[115] extended the scope of branch and bound algorithms to problems for which valid convex underestimating NLPs can be constructed for the nonconvex relaxations. The range of application of the proposed algorithm encompasses bilinear problems and separable problems involving functions for which convex underestimators can be built [10,83]. Because the nonconvex NLPs must be underestimated at each node, convergence can only be achieved if the continuous variables are branched on. A number of tests are suggested to accelerate the reduction of the solution space. They are summarized in the following.

Optimality Based Range Reduction Tests

For the first set of tests, an upper bound U on the nonconvex MINLP must be computed and a convex lower bounding NLP must be solved to obtain a lower bound L. If a bound constraint for variable x_i, with $x_i^L \leq x_i \leq x_i^U$, is active at the solution of the convex NLP and has multiplier $\lambda_i^* > 0$, the bounds on x_i can be updated as follows:

1) If $x_i - x_i^U = 0$ at the solution of the convex NLP and $\kappa_i = x_i^U - (U-L)/\lambda_i^*$ is such that $\kappa_i > x_i^L$, then $x_i^L = \kappa_i$.
2) If $x_i - x_i^L = 0$ at the solution of the convex NLP and $\kappa_i = x_i^L + (U-L)/\lambda_i^*$ is such that $\kappa_i < x_i^U$, then $x_i^U = \kappa_i$.

If neither bound constraint is active at the solution of the convex NLP for some variable x_j, the problem can be solved by setting $x_j = x_j^U$ or $x_j = x_j^L$. Tests similar to those presented above are then used to update the bounds on x_j.

Feasibility Based Range Reduction Tests

In addition to ensuring that tight bounds are available for the variables, the underestimators of the constraints are used to generate new constraints for the problem. Consider the constraint $g_i(\mathbf{x}, \mathbf{y}) \leq 0$. If its underestimating function $\underline{g}_i(\mathbf{x}, \mathbf{y}) = 0$ at the solution of the convex NLP and its multiplier is $\mu_i^* > 0$, the constraint

$$\underline{g}_i(\mathbf{x}, \mathbf{y}) \geq -\frac{U-L}{\mu_i^*}$$

can be included in subsequent problems.

The branch and reduce algorithm has been tested on a set of small problems.

Interval Analysis Based Approach

An approach based on interval analysis was proposed by [125] to solve to global optimality problems with a twice-differentiable objective function and once-differentiable constraints. Interval arithmetic allows the computation of guaranteed ranges for these functions [87,89,112]. The approach relies on the same concepts of successive partitioning of the domain and bounding of the objective function, while the branching takes place on the discrete and continuous variables. The main difference with the branch and bound algorithms is that bounds on the problem solution in a given domain are not obtained through optimization. Instead, they are based on the range of the objective function in the domain under consideration, as computed with interval arithmetic. As a consequence, these bounds may be quite loose and efficient fathoming techniques are required in order to enhance convergence. [125] suggested node fathoming tests and branching strategies which are outlined in the sequel. Convergence is declared when best upper and lower bounds are within a prespecified tolerance and when the width of the corresponding region is below a prespecified tolerance.

Node Fathoming Tests

The *upper-bound test* is a classical criterion used in all branch and bound schemes: If the lower bound for a node is greater than the best upper bound for the MINLP, the node can be fathomed.

The *infeasibility test* is also used by all branch and bound algorithms. However, the identification of infeasibility using interval arithmetic differs from its identification using optimization schemes. An inequality constraint $g_i(\mathbf{x}, \mathbf{y}) \leq 0$ is declared infeasible if its interval inclusion over the current domain, is positive. If a constraint is found to be infeasible, the current node is fathomed.

The *monotonicity test* is used in interval-based approaches. If a region is feasible, the monotonicity properties of the objective function can be tested. For this purpose, the inclusions of the gradients of the objective with respect to each variable are evaluated. If all the gradients have a constant sign for the current region, the objective function is monotonic and only one point needs to be retained from the current node.

The *nonconvexity test* is used to test the existence of a solution (local or global) within a region. If such a point exists, the Hessian matrix of the objective function at this point must be positive semidefinite. A sufficient condition is the nonnegativity of at least one of the diagonal elements of its interval Hessian matrix.

[125] suggested two additional tests to accelerate the fathoming process. The first is denoted as *lower bound test*. It requires the computation of a valid lower bound on the objective function through a method other than interval arithmetic. If the upper bound at a node is less than this lower bound, the region can be eliminated. The second test, the *distrust region method*, aims to help the algorithm identify infeasible regions so that they can be removed from consideration. Based on the knowledge of an infeasible point, interval arithmetic is used to identify an infeasible hypercube centered on that point.

Branching Strategies

The variable with the widest range is selected for branching. It can be a continuous or a discrete variable. In order to determine where to split the chosen variable, a relaxation of the MINLP is solved locally.

- Continuous Branching Variable: If the optimal value of the continuous branching variable, x^*, is equal to one of the variable bounds, branch at the midpoint of the interval. Otherwise, branch at $x^* - \beta$, where β is a very small scalar.
- Discrete Branching Variable: If the optimal value of the discrete branching variable, y^*, is equal to the upper bound on the variable, define a region with $y = y^*$ and one with $y^L \leq y \leq y^* - 1$, where y^L is the lower bound on y. Otherwise, create two regions $y^L \leq y \leq \text{int}(y^*)$ and $\text{int}(y^*) + 1 \leq y \leq y^U$, where y^U is the upper bound on y.

This algorithm has been tested on a small example problem and a molecular design problem [125].

Extended Cutting Plane for Pseudoconvex MINLPs

The use of the ECP algorithm for nonconvex MINLP problems was suggested in [135], using a modified algorithmic procedure as described in [136]. The main changes occur in the generation of new constraints for the MILP at each iteration (Step 4). In addition to the construction of the linear function $l_k(\mathbf{x}, \mathbf{y})$ at iteration k, the following steps are taken:

1) Remove all constraints for which $l_i(\mathbf{x}^k, \mathbf{y}^k) > g_{ji}(\mathbf{x}^k, \mathbf{y}^k)$. These correspond to linearizations which did not underestimate the corresponding nonlinear constraint at all points due to the presence of nonconvexities.
2) Replace all constraints for which $l_i(\mathbf{x}^k, \mathbf{y}^k) = g_{ji}(\mathbf{x}^k, \mathbf{y}^k) = 0$ by their linearization around $(\mathbf{x}^k, \mathbf{y}^k)$.
3) If constraint i is such that $g_i(\mathbf{x}^k, \mathbf{y}^k) > 0$, add its linearization around $(\mathbf{x}^k, \mathbf{y}^k)$.

The convergence criterion is also modified. In addition to the test used in Step 3, the following two conditions must be met:

1) $(\mathbf{x}^k - \mathbf{x}^{k-1})^{\mathsf{T}}(\mathbf{x}^k - \mathbf{x}^{k-1}) \leq \delta$, a pre-specified tolerance.
2) $\mathbf{y}^k - \mathbf{y}^{k-1} = 0$.

The ECP algorithm for pseudoconvex MINLPs has been used to address a trim loss problem arising in the paper industry [136]. A comparative study between the outer approximation, the generalized Benders decomposition and the extended cutting plane algorithm for convex MINLPs was presented in [120].

Reformulation/Spatial Branch and Bound Algorithm

A global optimization algorithm of the branch and bound type was proposed in [121]. It can be applied to problems in which the objective and constraints are functions involving any combination of binary arithmetic operations (addition, subtraction, multiplication and division) and functions that are either concave over the entire solution space (such as ln) or convex over this domain (such as exp).

The algorithm starts with an automatic reformulation of the original nonlinear problem into a problem that involves only linear, bilinear, linear fractional, simple exponentiation, univariate concave and univariate convex terms. This is achieved through the introduction of new constraints and variables. The reformulated problem is then solved to global optimality using a branch and bound approach. Its special structure allows the construction of a convex relaxation at each node of the tree. It should be noted that due to the introduction of many new constraints and variables the size of the convex relaxation of the reformulated problem increases substantially even for modest size problems. The integer variables can be handled in two ways during the generation of the convex lower bounding problem. The integrality condition on the variables can be relaxed to yield a convex NLP which can then be solved globally. Alternatively, the integer variables can be treated directly and the convex lower bounding MINLP can be solved using a branch and bound algorithm. This second approach is more computationally intensive but is likely to result in tighter lower bounds on the global optimum solution.

In order to obtain an upper bound on the optimum solution, a local MINLP algorithm can be used. Alternatively, the MINLP can be transformed to an equivalent nonconvex NLP by relaxing the integer variables. For example, a variable $y \in \{0, 1\}$ can be replaced by a continuous variable $z \in [0, 1]$ by including the constraint $z - z \cdot z = 0$.

This algorithm has been applied to reactor selection, distillation column design, nuclear waste blending, heat exchanger network design and multilevel pump configuration problems.

Hybrid Branch and Bound and Outer Approximation

[142] proposed a global optimization MINLP approach for the synthesis of heat exchanger networks without stream splitting. This approach is a hybrid branch and bound with outer approximation. It is based on two alternative convex underestimators for the heat transfer area. The first type of these convex underestimators along with the variable bounds and techniques for the bound contraction are based on a thermodynamic analysis. The second type is based on a relaxation and transformation so as to employ specific underestimation schemes. These convex underestimators result in a convex MINLP that is solved using the Outer Approximation approach and which provides valid lower bounds on the global solution. This approach has been applied to five heat exchanger network examples that employ the MINLP model of [138] that contains linear constraints and nonconvex objective function.

[141] introduced a deterministic branch and contract approach for structured process systems that have univariate concave, bilinear and linear fractional terms.

They proposed properties of the contraction operation and studied their effect on several applications.

The SMIN-αBB Algorithm

The SMIN-αBB global optimization algorithm, proposed by [2] is designed to solve to global optimality mathematical models where the binary/integer variables appear linearly and hence separably from the continuous variables and/or appear in at most bilinear terms, while nonlinear terms in the continuous variables appear separably from the binary/integer variables. These mathematical models become:

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x}) + \mathbf{x}^\top A_0 \mathbf{y} + \mathbf{c}_0^\top \mathbf{y} \\ \text{s.t.} & \mathbf{h}(\mathbf{x}) + \mathbf{x}^\top A_1 \mathbf{y} + \mathbf{c}_1^\top \mathbf{y} = \mathbf{0} \\ & \mathbf{g}(\mathbf{x}) + \mathbf{x}^\top A_2 \mathbf{y} + \mathbf{c}_2^\top \mathbf{y} \leq \mathbf{0} \\ & \mathbf{x} \in \mathbf{X} \subseteq \mathbb{R}^n \\ & \mathbf{y} \in \mathbf{Y} \quad \text{integer}, \end{cases} \tag{3}$$

where $\mathbf{c}_0^\top$, $\mathbf{c}_1^\top$ and $\mathbf{c}_2^\top$ are constant vectors, A_0, A_1 and A_2 are constant matrices and $f(\mathbf{x})$, $\mathbf{h}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ are functions with continuous second order derivatives.

The theoretical, algorithmic and computational studies of the SMIN-αBB algorithm are presented in detail in [41].

The GMIN-αBB Algorithm

The GMIN-αBB global optimization algorithm proposed in [2] operates within a branch and bound framework. The main difference with the algorithms of [56,92] and [20] is its ability to identify the global optimum solution of a much larger class of problems of the form

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{y}} & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0} \\ & \mathbf{x} \in \mathbf{X} \subseteq \mathbb{R}^n \\ & \mathbf{y} \in \mathbb{N}^q, \end{cases}$$

where $\mathbf{N}$ is the set of nonnegative integers and the only condition imposed on the functions $f(\mathbf{x},\mathbf{y})$, $\mathbf{g}(\mathbf{x},\mathbf{y})$ and $\mathbf{h}(\mathbf{x},\mathbf{y})$ is that their continuous relaxations possess continuous second order derivatives. This increased applicability results from the use of the αBB global optimization algorithm for continuous twice-differentiable NLPs [3,5,6,12].

The theoretical, algorithmic and computational studies of the GMIN-αBB Algorithm are presented in detail in [41].

See also

- ▶ Chemical Process Planning
- ▶ Complexity Classes in Optimization
- ▶ Complexity of Degeneracy
- ▶ Complexity of Gradients, Jacobians, and Hessians
- ▶ Complexity Theory
- ▶ Complexity Theory: Quadratic Programming
- ▶ Computational Complexity Theory
- ▶ Continuous Global Optimization: Applications
- ▶ Extended Cutting Plane Algorithm
- ▶ Fractional Combinatorial Optimization
- ▶ Generalized Benders Decomposition
- ▶ Generalized Outer Approximation
- ▶ Global Optimization in the Analysis and Management of Environmental Systems
- ▶ Information-based Complexity and Information-based Optimization
- ▶ Interval Global Optimization
- ▶ Kolmogorov Complexity
- ▶ MINLP: Application in Facility Location-allocation
- ▶ MINLP: Applications in Blending and Pooling Problems
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ MINLP: Branch and Bound Global Optimization Algorithm
- ▶ MINLP: Branch and Bound Methods
- ▶ MINLP: Design and Scheduling of Batch Processes
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Global Optimization with αBB
- ▶ MINLP: Heat Exchanger Network Synthesis
- ▶ MINLP: Logic-based Methods
- ▶ MINLP: Outer Approximation Algorithm
- ▶ MINLP: Reactive Distillation Column Synthesis
- ▶ MINLP: Trim-loss Problem
- ▶ Mixed Integer Linear Programming: Mass and Heat Exchanger Networks
- ▶ Parallel Computing: Complexity Classes

References

1. Acevedo J, Pistikopoulos EN (1996) A parametric MINLP algorithm for process synthesis problems under uncertainty. IĖC Res 35(1):147–158
2. Adjiman CS, Androulakis IP, Floudas CA (1997) Global optimization of MINLP problems in process synthesis and design. Comput Chem Eng Suppl 21:S445–S450
3. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, αBB, for general twice-differentiable NLPs – II. Implementation and computational results. Comput Chem Eng 22(9):1159–1179
4. Adjiman CS, Androulakis IP, Floudas CA (2000) Global optimization of mixed-integer nonlinear problems. AIChE J 46:1769–1797
5. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, αBB, for general twice-differentiable NLPs – I. Theoretical advances. Comput Chem Eng 22(9):1137–1158
6. Adjiman CS, Floudas CA (1996) Rigorous convex underestimators for general twice-differentiable problems. J Global Optim 9:23–40
7. Adjiman CS, Schweiger CA, Floudas CA (1998) Mixed-integer nonlinear optimization in process synthesis. In: Du D-Z, Pardalos PM (eds) Handbook Combinatorial Optim. Kluwer, Dordrecht
8. Aggarwal A, Floudas CA (1990) Synthesis of general distillation sequences – Nonsharp separations. Comput Chem Eng 14(6):631
9. Aggarwal A, Floudas CA (1992) Synthesis of heat integrated nonsharp distillation sequences. Comput Chem Eng 16:89
10. Al-Khayyal FA (1990) Jointly constrained bilinear programs and related problems: An overview. Comput Math Appl 19:53
11. Allgor RI, Barton PI (1997) Mixed integer dynamic optimization. Comput Chem Eng 21:S451–S456
12. Androulakis IP, Maranas CD, Floudas CA (1995) αBB: A global optimization method for general constrained nonconvex problems. J Global Optim 7:337–363
13. Barbosa-Povoa AP, Macchietto S (1994) Detailed design of multipurpose batch plants. Comput Chem Eng 18(11–12):1013–1042
14. Barton PI, Allgor RJ, Feehery WF (1998) Dynamic optimization in a discontinuous world. I–EC Res 37(3):966–981
15. Beale EML (1977) Integer programming. The State of the Art in Numerical Analysis. Acad Press, New York, pp 409–448
16. Bertsekas DL, Lower GS, Sandell NR, Posbergh TA (1983) Optimal short term scheduling of large-scale power systems. IEEE Trans Autom Control AC-28:1
17. Birewar DB, Grossmann IE (1989) Incorporating scheduling in the optimal design of multiproduct batch plants. Comput Chem Eng 13:141–161
18. Birewar DB, Grossmann IE (1990) Simultaneous synthesis, sizing and scheduling of multiproduct batch plants. Comput Chem Eng 29(11):2242
19. Bloom JA (1983) Solving an electricity generating capacity expansion planning problem by generalized benders' decomposition. Oper Res 31(5):84
20. Borchers B, Mitchell JE (1991) An improved branch and bound algorithm for mixed integer nonlinear programs. Techn Report Renssellaer Polytechnic Inst 200
21. Camarda KV, Maranas CD (1999) Optimization in polymer design using connectivity indices. I–EC Res 38:1884–1892
22. Churi N, Achenie LEK (1996) Novel mathematical programming model for computer aided molecular design. Industr Eng Chem Res 35(10):3788–3794
23. Churi N, Achenie LEK (1997) The optimal design of refrigerant mixtures for a two-evaporator refrigeration system. Comput Chem Eng 21(13):349–354
24. Ciric AR, Floudas CA (1989) A retrofit approach of heat exchanger networks. Comput Chem Eng 13(6):703
25. Ciric AR, Floudas CA (1990) A mixed-integer nonlinear programming model for retrofitting heat exchanger networks. I–EC Res 29:239
26. Ciric AR, Gu DY (1994) Synthesis of nonequilibrium reactive distillation processes by MINLP optimization. AIChE J 40(9):1479–1487
27. Ciric AR, Huchette SG (1993) Multiobjective optimization approach to sensitivity analysis waste treatment costs in discrete process synthesis and optimization problems. I–EC Res 32(11):2636–2646
28. Daichendt MM, Grossmann IE (1994) Preliminary screening for the MINLP synthesis of process systems: I. Aggregation techniques. Comput Chem Eng 18:663
29. Daichendt MM, Grossmann IE (1994) Preliminary screening for the MINLP synthesis of process systems: II. Heat exchanger networks. Comput Chem Eng 18:679
30. Dean JP, Dervakos GA (1996) Design of process-compatible biological agents. Comput Chem Eng Suppl A 20:S67–S72
31. Dimitriadis VD, Pistikopoulos EN (1995) Flexibility analysis of dynamic systems. I–EC Res 34(12):4451–4462
32. Dorneigh MC, Sahinidis NV (1995) Global optimization algorithms for chip layout and compaction. Eng Optim 25(2):131–154
33. Dua V, Pistikopoulos EN (1998) Optimization techniques for process synthesis and material design under uncertainty. Chem Eng Res Des 76(A3):408–416
34. Duran MA, Grossmann IE (1986) An outer approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307
35. Duvedi A, Achenie LEK (1997) On the design of environmentally benign refrigerant mixtures: A Mathematical Programming Approach. Comput Chem Eng 21(8):915–923

36. Faqir NM, Karimi IA (1990) Design of multipurpose batch plants with multiple production routes. Proc FOCAPD '89, Snowmass, Colorado, p 451
37. Fletcher R, Hall JAJ, Johns WR (1991) Flexible retrofit design of multiproduct batch plants. Comput Chem Eng 15(12):843
38. Fletcher R, Leyffer S (1994) Solving mixed integer nonlinear programs by outer approximation. Math Program 66(3):327–349
39. Fletcher R, Leyffer S (1998) Numerical experience with lower bounds for MIQP branch and bound. SIAM J Optim 8(2):604–616
40. Floudas CA (1995) Nonlinear and mixed integer optimization: Fundamentals and applications. Oxford Univ. Press, Oxford
41. Floudas CA (2000) Deterministic global optimization: Theory, methods and applications. Nonconvex Optim Appl. Kluwer, Dordrecht
42. Floudas CA, Aggarwal A, Ciric AR (1989) Global optimum search for nonconvex NLP and MINLP problems. Comput Chem Eng 13(10):1117–1132
43. Floudas CA, Ciric AR (1989) Strategies for overcoming uncertainties in heat exchanger network synthesis. Comput Chem Eng 13(10):1133
44. Floudas CA, Grossmann IE (1995) Algorithmic approaches to process synthesis: logic and global optimization. FOCAPD'94, 91 AIChE Symp Ser, pp 198–221
45. Ganish B, Horsky D, Srikanth K (1983) An approach to optimal positioning of a new product. Managem Sci 29:1277
46. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
47. Georgiadis MC, Rotstein GE, Macchietto S (1997) Optimal layout design in multipurpose batch plants. Industr Eng Chem Res 36(11):4852–4863
48. Geromel JC, Belloni MR (1986) Nonlinear programs with complicating variables: theoretical analysis and numerical experience. IEEE Trans Syst, Man Cybern SMC-16:231
49. Grossmann IE (1990) MINLP optimization strategies and algorithms for process synthesis. Proc 3rd Internat Conf on Foundations of Computer-Aided Process Design, p 105
50. Grossmann IE (1996) Mixed integer optimization techniques for algorithmic process synthesis. In: Anderson JL (ed) Advances in chemical engineering, process synthesis, vol 23. Acad Press, New York, pp 171–246
51. Grossmann IE, Floudas CA (1987) Active constraint strategy for flexibility analysis in chemical processes. Comput Chem Eng 11(6):675
52. Grossmann IE, Quesada I, Ramon R, Voudouris VT (1992) Mixed-integer optimization techniques for the design and scheduling of batch processes. Proc NATO Advanced Study. Inst Batch Process Systems Engineering
53. Grossmann IE, Sargent RWH (1979) Optimal design of multipurpose chemical plants. Industr Eng Chem Process Des Developm 18:343
54. Gupta A, Manousiouthakis V (1993) Minimum utility cost of mass exchange networks with variable single component supplies and targets. I–EC Res 32(9):1937–1950
55. Gupta OK (1980) Branch and bound experiments in nonlinear integer programming. PhD Thesis, Purdue Univ
56. Gupta OK, Ravindran R (1985) Branch and bound experiments in convex nonlinear integer programming. Managem Sci 31(12):1533–1546
57. Harding ST, Floudas CA (1997) Global optimization in multiproduct and multipurpose batch design under uncertainty. Industr Eng Chem Res 36(5):1644–1664
58. Hatzimanikatis V, Floudas CA, Bailey JE (1996) Analysis and design of metabolic reaction networks via mixed-integer linear optimization. AIChE J 42(5):1277–1292
59. Hatzimanikatis V, Floudas CA, Bailey JE (1996) Optimization of regulatory architectures in metabolic reaction networks. Biotechnol and Bioengin 52:485–500
60. Hoang HH (1982) Topological optimization of networks: A nonlinear mixed integer model employing generalized Benders decomposition. IEEE Trans Autom Control AC-27:164
61. Holmberg K (1990) On the convergence of the cross decomposition. Math Program 47:269
62. Ierapetritou MG, Pistikopoulos EN (1994) Simultaneous incorporation of flexibility and economic risk in operational planning under uncertainty. Comput Chem Eng 18(3):163–189
63. Ierapetritou MG, Pistikopoulos EN (1996) Batch plant design and operations under uncertainty. Industr Eng Chem Res 35(2):772–787
64. Ierapetritou MG, Pistikopoulos EN, Floudas CA (1996) Operational planning under uncertainty. Comput Chem Eng 20(12):1499–1516
65. Kalitventzeff B, Marechal F (1988) The management of a utility network. Process Systems Engineering. PSE '88, Sydney, Australia, p 223
66. Kelley JE (1960) The cutting plane method for solving convex programs. J SIAM 8(4):703–712
67. Kocis GR, Grossmann IE (1987) Relaxation strategy for the structural optimization of process flow sheets. I–EC Res 26(9):1869
68. Kocis GR, Grossmann IE (1988) Global optimization of nonconvex MINLP problems in process synthesis. I–EC Res 27(8):1407
69. Kocis GR, Grossmann IE (1989) Computational experience with DICOPT solving MINLP problems in process systems engineering. Comput Chem Eng 13:307
70. Kocis GR, Grossmann IE (1989) A modelling and decomposition strategy for the MINLP optimization of process flowsheets. Comput Chem Eng 13(7):797–819
71. Kokossis AC, Floudas CA (1990) Optimization of complex reactor networks-I. isothermal operation. Chem Eng Sci 45(3):595

72. Kokossis AC, Floudas CA (1991) Optimal synthesis of isothermal reactor-separator-recycle systems. Chem Eng Sci 46:1361
73. Kokossis AC, Floudas CA (1994) Optimization of complex reactor networks - II. Nonisothermal operation. Chem Eng Sci 49(7):1037
74. Kokossis AC, Floudas CA (1994) Stability in optimal design: Synthesis of complex reactor networks. AIChE 40(5):849–861
75. Kravanja Z, Grossmann IE (1990) PROSYN - An MINLP process synthesizer. Comput Chem Eng 14:1363
76. Kravanja Z, Grossmann IE (1996) A computational approach for the modeling/decomposition strategy in the MINLP optimization of process flowsheets with implicit models. I–EC Res 35(6):2065–2070
77. Liu ML, Sahinidis NV (1997) Process planning in a fuzzy environment. Europ J Oper Res 100(1):142–169
78. Luyben ML, Floudas CA (1994) Analyzing the interaction of design and control, Part 1: A multiobjective framework and application to binary distillation synthesis. Comput Chem Eng 18(10):933–969
79. Luyben ML, Floudas CA (1994) Analyzing the interaction of design and control, Part 2: Reactor–separator–recycle system. Comput Chem Eng 18(10):971–994
80. Maranas CD (1996) Optimal computer-aided molecular design: A polymer design case study. Industr Eng Chem Res 35(10):3403–3414
81. Maranas CD (1997) Optimal molecular design under property prediction uncertainty. AIChE J 43(5):1250–1264
82. Mawengkang H, Murtagh BA (1986) Solving nonlinear integer programs with large scale optimization software. Ann Oper Res 5:425
83. McCormick GP (1976) Computability of global solutions to factorable nonconvex programs: Part I – Convex underestimating problems. Math Program 10:147–175
84. Mockus L, Reklaitis GV (1996) A new global optimization algorithm for batch process scheduling. In: Floudas CA, Pardalos PM (eds) State of the art in global optimization. Kluwer, Dordrecht, pp 521–538
85. Mohideen MJ, Perkins JD, Pistikopoulos EN (1996) Optimal design of dynamic systems under uncertainty. AIChE J 42, no.8:2251–2272
86. Mohideen MJ, Perkins JD, Pistikopoulos EN (1997) Robust stability considerations in optimal design of dynamic systems under uncertainty. J Process Control 7(5):371–385
87. Moore RE (1979) Methods and applications of interval analysis. SIAM, Philadelphia
88. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Interscience Ser Discrete Math and Optim. Wiley, New York
89. Neumaier A (1990) Interval methods for systems of equations. Encycl Math Appl. Cambridge Univ. Press, Cambridge
90. Novak Z, Kravanja Z, Grossmann IE (1996) Simultaneous synthesis of distillation sequences in overall process schemes using an improved MINLP approach. Comput Chem Eng 20(12):1425–1440
91. Odele O, Macchietto S (1993) Computer aided molecular design: A novel method for optimal solvent selection. Fluid Phase Equilib 82:47
92. Ostrovsky GM, Ostrovsky MG, Mikhailow GW (1990) Discrete optimization of chemical processes. Comput Chem Eng 14:111–117
93. Papageorgaki S, Reklaitis GV (1990) Optimal design of multipurpose batch plants: 1. Problem formulation. I–EC Res 29(10):2054
94. Papageorgaki S, Reklaitis GV (1990) Optimal design of multipurpose batch plants: 2. A decomposition solution strategy. I–EC Res 29(10):2062
95. Papalexandri KP, Pistikopoulos EN (1993) An MINLP retrofit approach for improving the flexibility of heat exchanger networks. Ann Oper Res 42:119
96. Papalexandri KP, Pistikopoulos EN (1994) Synthesis and retrofit design of operable heat exchanger networks: 1. Flexibility and structural controllability aspects. I–EC Res 33:1718
97. Papalexandri KP, Pistikopoulos EN (1994) Synthesis and retrofit design of operable heat exchanger networks: 2. Dynamics and control structure considerations. I–EC Res 33:1738
98. Papalexandri KP, Pistikopoulos EN (1996) Generalized modular representation framework for process synthesis. AIChE J 42:1010
99. Papalexandri KP, Pistikopoulos EN, Floudas CA (1994) Mass exchange networks for waste minimization: A simultaneous approach. Chem Eng Res Developm 72:279
100. Pardalos PM, Schnitger G (1988) Checking local optimality in constrained quadratic programming is NP-Hard. Oper Res Lett 7(1):33
101. Pardalos PM, Vavasis SA (1991) Quadratic programming with one negative eigenvalue is NP-hard. J Global Optim 1:15
102. Paules IV GE, Floudas CA (1988) Synthesis of flexible distillation sequences for multiperiod operation. Comput Chem Eng 12(4):267
103. Paules IV GE, Floudas CA (1989) APROS: Algorithmic development methodology for discrete-continuous optimization problems. Oper Res 37(6):902
104. Paules IV GE, Floudas CA (1992) Stochastic programming in process synthesis: A two-stage model with MINLP recourse for multiperiod heat-integrated distillation sequences. Comput Chem Eng 16(3):189
105. Penteado FD, Ciric AR (1996) An MINLP approach for safe process plant layout. I–EC Res 35(4):1354–1361
106. Petkov SB, Maranas CD (1997) Multiperiod planning and scheduling of multiproduct batch plants under demand uncertainty. I–EC Res 36(11):4864–4881
107. Petkov SB, Maranas CD (1997) Quantitative assessment of uncertainty in the optimization of metabolic pathways. Biotechnol and Bioengin 56(2):145–161

108. Petkov SB, Maranas CD (1998) Design of single product campaign batch plants under demand uncertainty. AIChE J 44(4):896–911
109. Pistikopoulos EN, Ierapetritou MG (1995) Novel approach for optimal process design under uncertainty. Comput Chem Eng 19(10):1089–1110
110. Quesada I, Grossmann IE (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Comput Chem Eng 16:937–947
111. Raman VS, Maranas CD (1999) Optimization in product design with properties correlated with topological indices. Comput Chem Eng 45:997–1017
112. Ratschek H, Rokne J (1984) Computer methods for the range of functions. Ellis Horwood Ser Math Appl. Halsted Press, New York
113. Reklaitis GV (1991) Perspectives on scheduling and planning process operations. Proc. 4th. Internat. Symp. on Process Systems Engineering, Montreal, Canada
114. Rouhani R, Lasdon L, Lebow W, Warren AD (1985) A generalized Benders decomposition approach to reactive source planning in power systems. Math Program Stud 25:62
115. Ryoo HS, Sahinidis NV (1995) Global optimization of nonconvex NLPs and MINLPs with applications in process design. Comput Chem Eng 19(5):551–566
116. Sahinidis NV, Grossmann IE (1991) Convergence properties of generalized Benders decomposition. Comput Chem Eng 15(7):481
117. Schweiger CA, Floudas CA (1997) MINOPT: A software package for mixed-integer nonlinear optimization, user's guide. Manual Computer-Aided Systems Lab Dept Chemical Engin Princeton Univ
118. Schweiger CA, Floudas CA (1998) Interaction of design and control: optimization with dynamic models. In: Hager WW, Pardalos PM (eds) Optimal Control: Theory, Algorithms, and Applications. Kluwer, Dordrecht, pp 388–435
119. Schweiger CA, Floudas CA (1999) Optimization framework for the synthesis of complex reactor networks. I–EC Res 38:744–766
120. Skrifvars H, Harjunkoski I, Westerlund T, Kravanja Z, Pörn R (1996) Comparison of different MINLP methods applied on certain chemical engineering problems. Comput Chem Eng Suppl 20:S333–S338
121. Smith EMB, Pantelides CC (1996) Global optimisation of general process models. In: Grossmann IE (ed) Global Optimization in Engineering Design. Kluwer, Dordrecht, pp 355–386
122. Smith EMB, Pantelides CC (1999) A symbolic reformulation/spatial branch and bound algorithm for the global optimization of nonconvex MINLPs. Comput Chem Eng 23:457–478
123. Stefanis SK, Livingston AG, Pistikopoulos EN (1997) Environmental impact considerations in the optimal design and scheduling of batch processes. Comput Chem Eng 21(10):1073–1094
124. Turkay M, Grossmann IE (1996) Logic-based MINLP algorithms for the optimal synthesis of process networks. Comput Chem Eng 20(8):959–978
125. Vaidyanathan R, El-Halwagi M (1996) Global optimization of nonconvex MINLP's by interval analysis. In: Grossmann IE (ed) Global Optimization in Engineering Design. Kluwer, Dordrecht, pp 175–193
126. Vaidyaraman S, Maranas CD (1999) Optimal synthesis of refrigation cycles and selection of refrigerants. AIChE J 45:997–1017
127. Vaselenak J, Grossmann IE, Westerberg AW (1987) An embedding formulation for the optimal scheduling and design of multiproduct batch plants. I–EC Res 26(1):139
128. Vaselenak J, Grossmann IE, Westerberg AW (1987) Optimal retrofit design of multipurpose batch plants. I–EC Res 26(4):718
129. Vavasis S (1991) Nonlinear optimization: Complexity issues. Oxford Univ. Press, Oxford
130. Vecchietti A, Grossmann IE (1999) LOGMIP: A disjunctive 0-1 nonlinear optimizer for process system models. Comput Chem Eng 23:555–565
131. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer-approximation for MINLP optimization. Comput Chem Eng 14(7):769–782
132. Wellons MC, Reklaitis GV (1991) Scheduling of multipurpose batch chemical plants: 1. Formulation of single-product campaigns. I–EC Res 30(4):671
133. Wellons MC, Reklaitis GV (1991) Scheduling of multipurpose batch chemical plants: 1. Multiple product campaign formulation and production planning. I–EC Res 30(4):688
134. Westerlund T, Pettersson F (1995) An extended cutting plane method for solving convex MINLP problems. Comput Chem Eng 19:131–136
135. Westerlund T, Pettersson F, Grossmann IE (1994) Optimization of pump configuration problems as a MINLP problem. Comput Chem Eng 18(9):845–858
136. Westerlund T, Skrifvars H, Harjunkoski I, Pörn R (1998) An extended cutting plane method for a class of non-convex MINLP problems. Comput Chem Eng 22(3):357–365
137. Xia Q, Macchietto S (1997) Design and synthesis of batch plants: MINLP solution based on a stochastic method. Comput Chem Eng 21:S697–S702
138. Yee TF, Grossmann IE (1990) Simultaneous optimization models for heat integration - II. Heat exchanger network synthesis. Comput Chem Eng 14(10):1165–1184
139. Yee TF, Grossmann IE, Kravanja Z (1990) Simultaneous optimization models for heat integration - I. Area and energy targeting and modeling of multi-stream exchangers. Comput Chem Eng 14(10):1151–1164
140. Yee TF, Grossmann IE, Kravanja Z (1990) Simultaneous optimization models for heat integration - III. Area and energy targeting and modeling of multi-stream exchangers. Comput Chem Eng 14(11):1185–1200

141. Zamora JM, Grossmann IE (1998) Continuous global optimization of structured process systems models. Comput Chem Eng 22(12):1749–1770
142. Zamora JM, Grossmann IE (1998) A global MINLP optimization algorithm for the synthesis of heat exchanger networks with no stream splits. Comput Chem Eng 22(3):367–384
143. Zhang X, Sargent RWH (1994) The optimal operation of mixed production facilities: General formulation and some solution approaches. Proc. 5th Int. Symp. Process Systems Engineering, pp 171–177

Mixed Integer Optimization in Well Scheduling

DIMITRIOS I. GEROGIORGIS,
VASSILEIOS D. KOSMIDIS,
EFSTRATIOS N. PISTIKOPOULOS
Centre for Process Systems Engineering,
Imperial College London, London, UK

MSC2000: 76T30, 90C11, 90C90

Article Outline

Abstract

This Chapter presents a novel, mixed integer nonlinear programming (MINLP) model for the well scheduling problem, where the nonlinear behavior of the reservoir, wells, pipelines and surface facilities has been incorporated into the mathematical formulation. The well scheduling problem is formulated as a snapshot optimization problem with an objective function that expresses the maximization of an economic index. Discrete decisions here include the operational status of the wells, the allocation of wells to manifold or separators and the allocation of surface flowlines to separators. Continuous decisions include the well oil flowrates, and the allocation of gas-to-gas lift wells.

A three-step solution strategy is proposed for the solution of this problem, where logic based relations and piecewise linear approximations of oil field wells are integrated in the MINLP formulation. The model is solved following an Outer Approximation (OA) class algorithm. A number of examples are presented to illustrate the performance and business value of the proposed strategy; a remarkable increase in oil production of up to 10% is demonstrated, compared to results obtained via widespread heuristic methods. A further increase of 2.9% can be achieved by dynamic optimization based on explicit consideration of the multiphase flow within the reservoirs of a particular oil field.

Introduction

In an era of globalized business operations, large and small oil and gas producers alike strive to foster profitability by improving the agility of exploration endeavors and the efficiency of oil production, storage and transport operations [7]. Consequently, they all face acute challenges: ever-increasing international production, intensified global competition, price volatility, operational cost reduction policies, aggressive financial goals (market share, revenue, cash flow and profitability) and strict environmental constraints (offshore extraction, low sulphur): all these necessitate a high level of oilfield modeling accuracy, so as to maximize recovery from certified reserves. Straightforward translation of all considerations to explicit mathematical objectives and constraints can yield optimal oilfield network design, planning and operation policies. Therefore, the foregoing goals and constraints should be explicitly incorporated and easily revised if the generality of production optimization algorithms is to be preserved. This Chapter provides a summary of a new, efficient MINLP optimization formulation for well scheduling, and a novel strategy towards integration of equation-oriented process modeling and multiphase reservoir computational fluid dynamics (CFD), in order to include the dynamic behavior of reservoirs into oil and gas production models.

The problem of fuel production optimization subject to explicit oilfield constraints has attracted significant attention, documented in many petroleum engineering publications. A comprehensive literature review by Kosmidis [18] classifies previous algorithms in 3 broad categories (simulation, heuristic, and mathematical programming methods) and underlines that most are applied either to simple pipeline networks of modest size, relying on heuristic rules of limited applicability, or are suitable for special structures. Reducing the computational burden (focus on natural-flow wells or gas-lift wells only, or reducing well network connectivity discrete variables) is a crucial underlying pattern.

Dynamic oil and gas production systems simulation and optimization is a research trend which has the clear potential to meet the foregoing challenges of the international oil and gas industry and assist producers in achieving business goals and energy needs. Previous work [8,19,20,23] has addressed successfully research challenges in this field, using appropriate simplifying correlations [25] for two-phase flow of oil and gas in production wells and pipelines. A series of assumptions are routinely adopted to achieve manageable computational complexity: the fundamental one is the steady-state assumption for the reservoir model, based on the enormous timescale difference between different spatial levels (oil and gas reservoir dynamics evolve in the order of weeks, the respective timescales of pipeline networks are in the order of minutes, and the production optimization horizon is in the order of days). The decoupling of reservoir simulation from surface facilities optimization is based on these timescale differences among production elements [2,25]. While the surface and pipeline facilities are in principle no different from those found in any petrochemical plant, sub-surface elements (reservoirs, wells) induce complexity which must be addressed via a systematic strategy that has not been hitherto proposed.

In some petroleum fields, such as the Prudhoe Bay [22], a production well can be connected to different manifolds that lead to different separators. In such fields, switching a well from one manifold to another could be an effective way to increase oil production and/or reduce production cost by making optimal use of the existing resources such as the capacity of separators [9]. However, for best results, the well connection must be optimized simultaneously with the well oil rate and gas lift rate. The corresponding optimization problem is known as well scheduling problem.

Problem Statement

The well scheduling problem in integrated oil and gas production can be stated as follows: given are (i) a *set of wells*, which could be closed (shut in) or connected to manifolds or separators, (ii) a *set of flowlines* which could be connected to separators. The goal is to determine: (i) the *operational status of the wells*, i. e. closed or open, (ii) the *connection of wells* to manifolds or separators, (iii) the *connection of flow lines* to separators, (iv) the *well oil flowrate* and the (v) the *allocation of gas* to gas lift wells, which maximize the net revenue (oil sales minus the cost of gas compression), while satisfying physical laws and operational constraints such as: (i) a *well bore model*, (ii) *mass, energy and momentum balances* throughout the production network, (iii) *upper and lower well oil rate constraints* and minimum pressure constraints at the inlet and outlet of the flowlines, (iv) *maximum oil, gas and water capacity constraints* in the separators, (v) an *upper bound on gas lift availability* and (vi) a *maximum number of well switches*. The first step in order to determine the optimal well configuration that maximizes the revenue is to develop a suitable superstructure that includes all the possible pipeline network configurations. Such a production network superstructure is shown in Fig. 1 and includes the *reservoir (R)*, the *wells (W)*, the *manifolds (M)*, and the *separators (S)* nodes as well as the potential connection of wells to manifolds or separators and flowlines to separators. Two types of wells are considered: (i) *type A wells* that can be connected only to manifolds, and (ii) *type B wells* that can be connected to separators. It must be noted that a feasible production network should satisfy the following requirements: (i) a *type A well* should be either shut in or else connected to one manifold, (ii) a *type B well* should be either shut in or else connected to one separator, and (iii) a manifold flowline must be connected to one separator.

Optimization Model

This section presents the MINLP optimization model for the well scheduling problem, based on the following assumptions:

- the system is under steady state conditions,
- a homogeneous slip model which is applied to determine the pressure drop in the pipelines,
- the temperature of the reservoir is known,
- the operating pressures of the separators are constant, and
- the thermodynamic description of the fluid is based on the *black oil model*.

For the development of the MINLP optimization model, the following sets, variables and parameters are defined:

Sets

I	set of wells
I_A	set of wells of type A
I_B	set of wells of type B
M	set of manifolds
S	set of separators

Indices

i, i_A, i_B	well in set I, I_A, I_B respectively
m	manifold in set M
s	separator in set S

Binary decision variables

$$y_i = \begin{cases} 1 & \text{if well } i \text{ is open} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{i,m} = \begin{cases} 1 & \text{if well } i \text{ is connected to manifold } m \\ 0 & \text{otherwise} \end{cases}$$

$$y_{m,s} = \begin{cases} 1 & \text{if manifold } m \text{ is connected to separator } s \\ 0 & \text{otherwise} \end{cases}$$

$$y_{i,s} = \begin{cases} 1 & \text{if well } i \text{ is connected to separator } s \\ 0 & \text{otherwise} \end{cases}$$

Continuous variables

$q_{p,i}$	flowrate in stock tank conditions of phase p from well i
$q_{p,i,m}$	flowrate in stock tank condition of phase p from well i to manifold m
$q_{p,m,s}$	flowrate in stock tank condition of phase p from manifold m to separator s
$q_{p,s}$	flowrate in stock tank condition of phase p in separator s
$P_{i,m}$	pressure of well i at the manifold level
P_m	manifold pressure
$H_{i,m}$	total enthalpy of well i at the manifold level
H_m	manifold enthalpy

The proposed model includes the following elements: (i) the well bore model, (ii) the mass, momentum and energy balances in well, manifold and separator nodes, (iii) the network logic constraints, (iv) the well and flowline momentum and energy balances, (v) the maximum number of allowable well switches, and (vi) the objective function.

Wellbore Model

The wellbore model describes the multiphase fluid flow from the reservoir to the wellbore and comprises the following equations:

$$q_{o,i} = PI_i(P_{R,i} - P_i^{\mathrm{wf}}) \,, \quad \forall i \in I \tag{1}$$

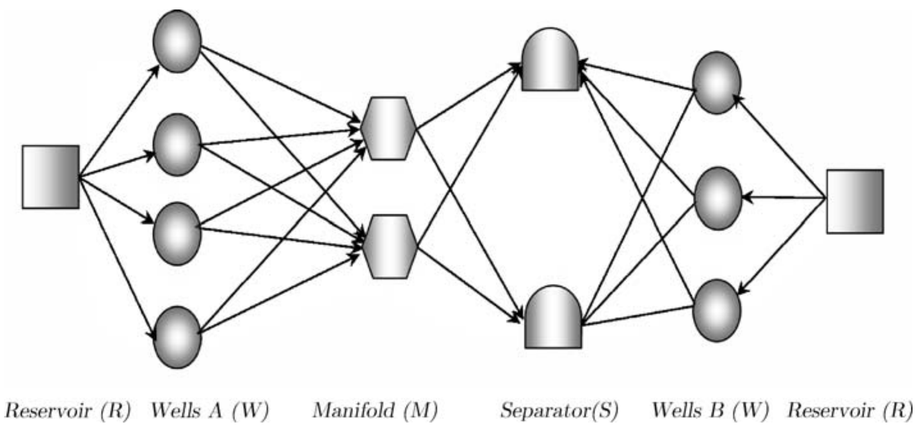
$$q_{g,i}^{\mathrm{f}} = f_o(q_{o,i}) \,, \quad \forall i \in I \tag{2}$$

$$q_{w,i} = f_w(q_{o,i}) \,, \quad \forall i \in I \tag{3}$$

$$T_i = T_R \,, \quad \forall i \in I \tag{4}$$

$$H_i = f_H(P_i^{\mathrm{wf}}, T_i, q_{o,i}, q_{w,i}, q_{g,i}) \,, \quad \forall i \in I \tag{5}$$

where P_i^{wf} is the bottomhole pressure and $q_{g,i}^{\mathrm{f}}$ is the formation gas flowrate in stock tank conditions. Equations (2) and (3) can be nonlinear in order to model the case of gas and water coning wells. These nonlinear relations are generated either by using Addington's correlations [1] or by repetitively solving a well coning model for different values of well oil rate $q_{o,i}$, in order to calculate the corresponding water $q_{w,i}$ and gas



Mixed Integer Optimization in Well Scheduling, Figure 1
Production network superstructure for the well scheduling problem

flowrates $q^{\text{f}}_{g,i}$. In the latter case, Eq. (2) and (3) are constructed via curve fitting to the data series $(q_{o,i}, q_{o,i})$ and $(q_{o,i}, q^{\text{f}}_{g,i})$ respectively. For naturally flowing wells, the total gas flowrate is given by:

$$q_{g,i} = q^{\text{f}}_{g,i} \,, \; \forall i \in D = \{i \in I \mid \text{natural flow}\} \qquad (6)$$

while for gas lift wells the total gas flowrate is equal to:

$$q_{g,i} = q^{\text{f}}_{g,i} + q^{\text{inj}}_{g,i} \,, \; \forall i \in F = \{i \in I \mid \text{gas lift}\} \,. \qquad (7)$$

Mass, Momentum and Energy Balances in Well, Manifold and Separator Nodes

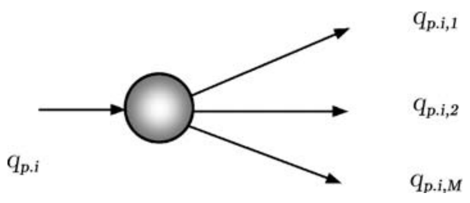
A well node of type A can be modeled as a splitter $i \in I_A$, which consists of an inlet stream that represents the fluid flow from the reservoir, and a set of outlet streams that represents the potential connections of a well to manifolds as shown in Fig. 2. The mass balances around the splitter for each phase are given by the following relations:

$$q_{p,i} = \sum_m q_{p,i,m} \,, \; \forall p \in \{o, w, g\} \,, \; i \in I_A \,. \qquad (8)$$

Similarly, the mass balances around a well node of type B are given by:

$$q_{p,i} = \sum_s q_{p,i,s} \,, \; \forall p \in \{o, w, g\} \,, \; i \in I_B \,. \qquad (9)$$

There is also an upper and a lower bound in the well oil flowrates. The upper bound is enforced to prevent



Mixed Integer Optimization in Well Scheduling, Figure 2
Splitter node

sand production [4], while the lower bound is imposed to satisfy stable flow [27]:

$$y_i q^L_{o,i} \le q_{o,i} \le q^U_{o,i} y_i \,, \; \forall i \in I \,. \qquad (10)$$

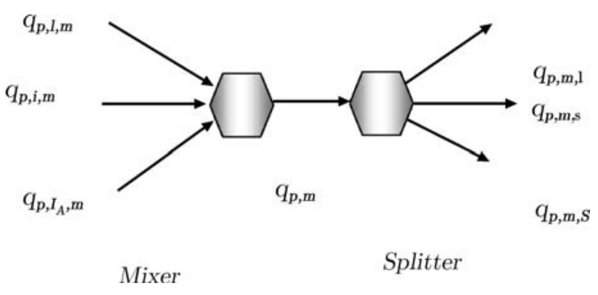
Equation (10) states that if well i is open ($y_i = 1$), then the well oil flowrate $q_{o,i}$ is constraint by an upper and a lower bound, while if well i is shut in ($y_i = 0$), then the well oil flowrate $q_{o,i}$ is zero.

Manifold Node A manifold node is shown in Fig. 3 and performs two tasks: (i) mixing and (ii) splitting. The mass balance of the mixer for each phase is given by:

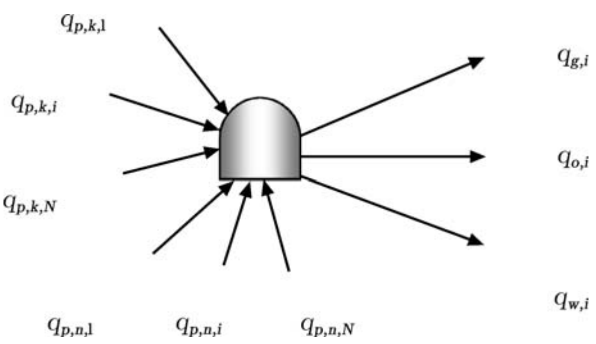
$$\sum_{i \in I_A} q_{p.i.m} = q_{p.m} \,, \; \forall p \in \{o, w, g\} \,, \; m \in M \,. \quad (11)$$

The splitter allocates the manifold fluid $q_{p,m}$ to one separator s. The mass balances for each phase around the splitter are given by:

$$q_{p,m} = \sum_s q_{p,m,s} \,, \; \forall p, m \in M \,. \qquad (12)$$



Mixed Integer Optimization in Well Scheduling, Figure 3
Manifold node



Mixed Integer Optimization in Well Scheduling, Figure 4
Separation node

All wells that are connected to manifold m must operate at the same pressure (P_m):

$$L^P(1-y_{i,m}) \leq P_i^m - P_m \leq U^P(1-y_{i,m}) \,, \quad \forall i \in I_A \,,\ m \in M \tag{13}$$

where $P_{i,m}$ is the pressure of well i at manifold level m, and L^P, U^P are the corresponding upper and lower bounds, respectively. Moreover, if manifold m is connected to separator s, then its inlet pressure must be greater than the separator pressure:

$$P_s y_{m,s} \leq P_m \,, \ \forall m \in M \,,\ s \in S \,. \tag{14}$$

The pressure of the flowline at the separator level $P_{m,s}$ is equal to the separator pressure:

$$P_{m,s} = \sum_s y_{m,s} P_s \,. \tag{15}$$

The energy balance in the manifold is given by:

$$\sum_{i \in I_A} H_i^m = H_m \,, \ \forall m \in M \tag{16}$$

where H_i^m is the enthalpy of well i at manifold level m.

Separator Node Each separator s has a set of inlet streams coming from the flowlines and type B wells, as shown in Fig. 4. The mass balances for each phase are given by the following relation:

$$\sum_m q_{p,m,s} + \sum_{i \in I_B} q_{p,i,s} = q_{p,s} \,, \ \forall p, s \in S \tag{17}$$

while the separator capacity constraints must also be satisfied:

$$q_{p,s} \leq C_{p,s} \,, \ \forall p, s \in S \,. \tag{18}$$

Finally, the total amount of gas available for gas lift is restricted by the compressor capacity (C_C):

$$\sum_i q_{g,i}^{\text{inj}} \leq C_c \,. \tag{19}$$

Network Logic Constraints

A well of type A could either be shut in, or else connected to one manifold:

$$\sum_m y_{i,m} = y_i \,, \ \forall i \in I_A \tag{20}$$

$$y_{i,m} \leq y_i \,, \ \forall i \in I_A \,,\ m \in M \,. \tag{21}$$

The integer Eq. (20) states that if the well is open ($y_i = 1$) then it should be connected to one manifold, while Eq. (21) states that if the well is shut in ($y_i = 0$) then all binary variables $y_{i,m}$ which represent the connection of well i to manifold m are zero.

Similarly, a well of type B could either be shut in, or else connected to one separator:

$$\sum_s y_{s,i} = y_i \,, \ \forall i \in I_B \tag{22}$$

$$y_{s,i} \leq y_i \,, \ \forall i \in I_B \,,\ s \in S \,. \tag{23}$$

Furthermore, it is also necessary to enforce the condition that each manifold flowline is connected to one separator:

$$\sum_s y_{m,s} = 1 \,, \ \forall m \in M \,. \tag{24}$$

Moreover, if the connection of well i to manifold m or separator s does not exist, then its corresponding flowrates and enthalpies must be zero:

$$0 \leq H_i^m \leq H^U y_{i,m} \,, \ \forall i \in I_A \tag{25}$$

$$0 \leq q_{p,i,m} \leq q_{p,i,m}^U y_{i,m} \,, \ \forall p, i \in I_A \,, \ m \in M \tag{26a}$$

$$0 \leq q_{p,i,s} \leq q_{p,i,s}^U y_{i,s} \,, \ \forall p, i \in I_B \,, \ s \in S \tag{26b}$$

Well and Flowline Momentum and Energy Balances

1. *Naturally flowing wells of type A.* Kosmidis [18] discusses how naturally flowing wells of type A can be accurately approximated by piecewise linear functions:

$$q_{o,i}^{\max} = \sum_j \eta_{i,j} q_{o,i,j}^{d,\max} \,, \ \forall i \in I_A \tag{27a}$$

$$P_i^m = \sum_j \eta_{i,j} P_{i,j}^d \,, \ \forall i \in I_A \tag{27b}$$

$$H_i^m = \sum_j \sum_k \lambda_{i,j,k} H_{i,j,k}^d \,, \ \forall i \in I_A \tag{27c}$$

$$q_{o,i} = \sum_j \sum_k \lambda_{i,j,k} q_{o,i,j,k}^d \,, \ \forall i \in I_A \tag{27d}$$

$$q_{o,i} \leq q_{o,i}^{\max} \,, \ \forall i \in I_A \tag{27e}$$

$$\sum_j \sum_k \lambda_{i,j,k} = y_i \,, \ \forall i \in I_A \tag{27f}$$

$$\begin{aligned} &\eta_{i,j} = \sum_k \lambda_{i,j,k} \,, \ \xi_{i,k} = \sum_j \lambda_{i,j,k} \,, \\ &\zeta_{i,t} = \sum_j \lambda_{i,j,j+t} \,, \ \forall i \in I_A \end{aligned} \tag{27g}$$

$$\eta_{i,j} \,, \ \xi_{i,k} \,, \ \zeta_{i,t} \geq 0 \,, \ SOS \,, \ \forall i \in I_A \,. \tag{27h}$$

It must be noted that if well i is shut in ($y_i = 0$), then all continuous variables in constraint (27) are set equal to zero, as it can be observed from constraint (27f). The piecewise linear approximation of the well model is constructed in a pre-processing step by discretizing:

(i) the manifold pressure between the valid lower (L^P) and upper (U^P) bound, and

(ii) the well oil rate in the interval $[q_{o,i}^L, \ q_{o,i}^U]$.

The lower bound (L^P) is equal to the lowest operating pressure of the separators, while the upper bound (U^P) must be greater than the highest operating pressure of the separators.

2. *Naturally flowing wells of type B.* For the case of naturally flowing wells of type B, the oil flowrate $q_{o,i,s}$ of well i in separator s is given by:

$$q_{o,i,s} \leq q_{o,i,s}^{d,\max} y_{i,s} \,, \ \forall i \in I_B \tag{28}$$

where $q_{o,i,s}^{d,\max}$ is calculated in a pre-processing step for each fixed pressure separator s by setting the choke fully open.

3. *Gas lift wells of type A.* These can be accurately approximated by the following set of mixed integer linear relations:

$$q_{o,i} = \sum_j \sum_k \lambda_{i,j,k} q_{o,i,j,k}^d \,, \ \forall i \in I_B \tag{29a}$$

$$q_{g,i}^{\text{inj}} = \sum_j \sum_k \lambda_{i,j,k} q_{g,i,k}^{d,\text{inj}} \,, \ \forall i \in I_B \tag{29b}$$

$$P_i^m = \sum_j \sum_k \lambda_{i,j,k} P_{i,j}^d \,, \ \forall i \in I_B \tag{29c}$$

$$H_i^m = \sum_j \sum_k \lambda_{i,j,k} H_{i,j,k}^d \,, \ \forall i \in I_B \tag{29d}$$

$$\sum_j \sum_k \lambda_{i,j,k} = y_i \,, \ \forall i \in I_B \tag{29e}$$

$$\begin{aligned} &\eta_{i,j} = \sum_j \lambda_{i,j,k} \,, \ \xi_{i,k} = \sum_k \lambda_{i,j,k} \,, \\ &\zeta_{i,t} = \sum_j \mu_{i,j,j+t} \,, \ \forall i \in I_B \end{aligned} \tag{29f}$$

$$\eta_{i,j} \,, \ \xi_{i,k} \,, \ \zeta_{i,t} \geq 0 \, (SOS) \,, \ \forall i \in I_B \,. \tag{29g}$$

These relations are constructed in a pre-processing step by discretizing:

(iii) the manifold pressure in the interval $[\, L^P \,, \, U^P]$, and

(iv) the well gas injection rate in the interval $[0, \ q_{g,i}^{\text{inj},U}]$, where $q_{g,i}^{\text{inj},U}$ is the gas injection rate at the upper bound pressure (U^P), where the well oil flowrate is reduced despite the increase in gas injection rate.

4. *Gas lift wells of type B.* These are connected to a fixed pressure separator and they can be accurately approximated as follows:

$$q_{o,i,s} = \sum_j \lambda_{i,j,s} q^d_{o,i,j,s} \, , \quad \forall i \in I_B \, , \; s \in S \tag{30a}$$

$$q^{\text{inj}}_{g,i,s} = \sum_j \lambda_{i,j,s} q^{d,\text{inj}}_{g,i,j,s} \, , \quad \forall i \in I_B \, , \; s \in S \tag{30b}$$

$$\sum_j \lambda_{i,j,s} = y_{i,s} \, , \quad \forall i \in I_B \, , \; s \in S \tag{30c}$$

$$\lambda_{i,j,s} \geq 0 \, , \; SOS \, . \tag{30d}$$

Flowline Momentum Balance The momentum balance in the manifold flowlines is given by:

$$P_{m,s} - f_P(P_m, H_m, q_{o,m}, q_{g,m}, q_{w,m}) = 0 \, , \quad \forall m \in M \, , \; s \in S \tag{31}$$

where $P_{m,s}$ is the pressure of flowline m at separator level s.

Remark During construction of the piecewise linear approximations or calculation of $q^{\max}_{o,i,s}$, it is possible to identify naturally flowing wells of type A or type B which are unable to flow towards certain manifolds or separators. To exclude these infeasible connections, the following logic constraints are incorporated in the mathematical formulation:

$$y_{i,m} \leq 1 - y_{m,s} \, , \quad \forall i \in I_A \tag{32}$$

$$y_{i,s} \leq 0 \, , \quad \forall i \in I_B \, . \tag{33}$$

Constraint (32) states that if flowline m is connected to separator $s (y_{m,s} = 1)$, then well i cannot be connected to manifold m.

Maximum Number of Well Switches

There is an upper bound on the number of well switches (for wells of both types A and B) that can be performed within a day. This is an operational constraint and is applied to avoid huge flow variations which may eventually lead to a surface facility shut down. To consider and model this requirement, the following binary variables and parameters are introduced:

Binary variables

$$c^{\text{f}}_i = \begin{cases} 1 & \text{if the well } i \text{ is open and in the previous day was closed} \\ 0 & \text{otherwise} \end{cases}$$

$$c^{\text{nf}}_i = \begin{cases} 1 & \text{if the well } i \text{ is closed and in the previous day was open} \\ 0 & \text{otherwise} \end{cases}$$

$$c_{i,m} = \begin{cases} 1 & \text{if the well } i \text{ of type A is connected to a new manifold } m \text{ on this day} \\ 0 & \text{otherwise} \end{cases}$$

$$c_{i,s} = \begin{cases} 1 & \text{if the well } i \text{ of type B is connected to a new separator } s \text{ on this day} \\ 0 & \text{otherwise} \end{cases}$$

$$c_{m,s} = \begin{cases} 1 & \text{if the flowline } m \text{ is connected to a new separator } s \text{ on this day} \\ 0 & \text{otherwise .} \end{cases}$$

Parameters

$NC^{\max}_A$	maximum number of switches for the wells of type A.
$NC^{\max}_B$	maximum number of switches for the wells of type B.
y^b	binary parameters representing the well structure of the previous day.

One switch is accounted for in the following cases:

(i) *A well i was closed and is currently open.* This case is modeled by incorporating the following constraint in the formulation:

$$y_i - y^b_i \leq c^{\text{f}}_i \, , \quad \forall i \in I \, . \tag{34}$$

Thus, if well i is open $(y_i = 1)$ while it was previously closed $(y^b_i = 0)$, then one well switch is accounted for by forcing the binary variable c^{f}_i to be 1.

(ii) *A well i was open and is currently closed.* To incorporate this well switch in the formulation, the following constraint is used:

$$y^b_i - y_i \leq c^{\text{nf}}_i \, , \quad \forall i \in I \, . \tag{35}$$

(iii) *A well of type A switches manifold.* If well i is currently connected to manifold $m (y_{i,m} = 1)$ and it was previously connected to manifold $m' \neq m$

(that implies $y^b_{i,m} = 0$), then there is one well switch, which is modeled by the following constraint in the formulation:

$$y_{i,m} - y^b_{i,m} \leq c_{i,m}\,, \quad \forall i \in B\,,\ m \in M \qquad (36)$$

where the set $B = \{i \in I_A \mid y^b_i \neq 0\}$ is the set of wells of type A that were open during the previous day. Constraint (36) is applicable only for wells of type A that were previously open ($y^b_i \neq 0$), to avoid double counting a well switch; the case of a well that was closed ($y^b_i = 0$) and is currently open ($y_i = 1$) is considered by constraint (34).

(iv) *A well of type B switches to a new separator.* If well i is currently connected to separator $s(y_{i,s} = 1)$ and was previously connected to separator $s' \neq s(y^b_{i,s} = 0)$, then there is one well switch, which is modeled by the following constraint in the formulation:

$$y_{i,s} - y^b_{i,s} \leq c_{i,s}\,, \quad \forall i \in C\,,\ s \in S \qquad (37)$$

where the set $C = \{i \in I_B \mid y^b_i \neq 0\}$ is the set of wells of type B that were open during the previous day.

(v) *A manifold flowline switches to a new separator.* If a manifold flowline m is currently connected to a separator $s(y_{m,s} = 1)$ and was previously connected to separator $s' \neq s(y^b_{m,s} = 0)$, then there is one switch, which is accounted for by forcing the binary variable $c_{m,s}$ to be 1:

$$y_{m,s} - y^b_{m,s} \leq c_{m,s}\,, \quad \forall m \in M\,,\ s \in S\,. \qquad (38)$$

The sum of switches for the wells of type A and B must be less then an upper bound:

$$\sum_{i\in I_A}(c^{\mathrm{f}}_i + c^{\mathrm{nf}}_i) + \sum_{i\in B}\sum_{m\in M} c_{i,m} + \sum_{m\in M}\sum_{s\in S} c_{m,s} \leq NC^{\max}_A \qquad (39)$$

$$\sum_{i\in I_A}(c^{\mathrm{f}}_i + c^{\mathrm{nf}}_i) + \sum_{i\in C}\sum_{s\in S} c_{i,s} \leq NC^{\max}_B\,. \qquad (40)$$

Objective Function

The objective function is the maximization of daily revenue:

$$\max\ w_o \sum_{i\in I} q_{o,i} - w_g \sum_{i\in I} q^{\mathrm{inj}}_{g,i} \qquad (41)$$

The control variables are:

(i) the well operational status (open or close),
(ii) the well connections to manifolds and separators,
(iii) the flowline connections to separators,
(iv) the well oil flowrates, and
(v) the gas injection rates into gas lift wells.

An MINLP Formulation for the Well Scheduling Problem

By defining the vectors $\mathbf{x} = [P, H]$, $\mathbf{q}_p = [q_o, q_g, q_w, q^{\mathrm{inj}}_g]$, $\mathbf{y}^s = [y_i, y_{i,m}, y_{m,s}, y_{i,s}]$, $\mathbf{c} = [c^{\mathrm{f}}_i, c^{\mathrm{nf}}_i, c_{i,m}, c_{i,s}, c_{m,s}]$, $\boldsymbol{\phi} = [\lambda_{i,j}, \xi_{i,k}, \zeta_{i,t}]$ and $\mathbf{y}^a$ (the vector of binary variables that are used to impose the adjacency condition in *SOS*-type variables), the mathematical programming formulation (P) for the well scheduling problem can be concisely expressed as:

$$P\colon\ \max\ \psi(q_{o,i}, q^{\mathrm{inj}}_{g,i}) \qquad (42)$$

subject to

$$m_1(\mathbf{x}_i,\ \mathbf{q}_{p,i}) = 0 \qquad (43)$$

$$m_2(\mathbf{q}_{p,i},\ \mathbf{q}_{p,i,m}, \mathbf{q}_{p,i,s}, \mathbf{q}_{p,s}, \mathbf{x}_i, \mathbf{x}_{i,m}, \mathbf{y}^s) \leq 0 \qquad (44)$$

$$m_3(\mathbf{q}_{p,i,m}, \mathbf{q}_{p,i,s}, \mathbf{y}^s) \leq 0 \qquad (45)$$

$$m_4(q_{o,i}, q^{\mathrm{inj}}_{g,i}, q_{o,i,s}, q^{\max}_{o,i,s}, q^{\mathrm{inj}}_{g,i,s}\mathbf{x}_{i,m}, \phi, \mathbf{y}^s, \mathbf{y}^a) = 0 \qquad (46)$$

$$m_5(\mathbf{x}_m, \mathbf{q}_{p,m}) = 0 \qquad (47)$$

$$m_6(\mathbf{y}^s, \mathbf{c}) \leq 0\,. \qquad (48)$$

The equivalence of the equations within the above model (P) is explained as follows. Equation (42) is equivalent to the linear objective function (41). Equation (43) represents the nonlinear wellbore model Eq. (1)–(7). Equation (44) represents the mixed integer linear mass, momentum and energy balances in the wells, manifold and separator nodes and is equivalent to Eq. (8)–(19). Equation (45) represents the mixed integer linear network logic constraints and is equivalent to Eq. (20)–(26). Equation (46) represents the well piecewise linear approximation and is equivalent to Eq. (27)–(30). Equation (47) represents the nonlinear momentum balance in the flowlines and is equivalent to Eq. (31). Finally, Eq. (48) represents the integer logic relations associated with the ability of naturally flowing wells to flow and with the relevant well switches, and is equivalent to Eq. (32)–(40).

The mathematical programming formulation P includes: (i) binary variables, and (ii) nonlinear equations. Therefore, it belongs to the class of mixed integer nonlinear programming (MINLP) problems. There are two categories of binary variables: the one that is associated with the structure of the production network and the well switches ($\mathbf{y}^s, \mathbf{c}$), and a second that is used to impose the adjacency condition on *SOS*-type variables ($\mathbf{y}^a$). Moreover, the number of nonlinear equations is equal to the number of coning wells plus the number of flowlines.

The most popular methods for solving MINLP problems are those that proceed by solving a sequence of nonlinear (NLP) and mixed integer linear programs (MILP) problems. These include Generalized Benders decomposition (GBD, Geoffrion [3]) and Outer Approximation (OA, Kocis and Grossmann [5]). The disadvantage of GBD is that it may require a significant number of major iterations of the NLP subproblem and the MILP master problem. The major advantage of OA is that it typically requires fewer iterations to achieve a solution, since its MILP master problem contains more information than the GBD formulation. Conversely, because the OA master problem is richer, it is also more time-consuming to solve. A detailed review of the various MINLP algorithms has been published by Floudas [10].

This Chapter considers an approach based on Outer Approximation (OA), since it typically requires fewer iterations when compared to other MINLP techniques. Also, its modified version (Outer Aproximation/Augmented Penalty (OA/AP), (Viswanathan and Grossmann, 1990)) has been found to be capable of handling mild nonconvexities present in the MINLP problems.

Optimization Strategy

The first NLP subproblem of the OA/AP algorithm involves solving an optimization problem where the structure of the pipeline network is the one of the previous day. The lth NLP subproblem ($l > 1$) involves fixing the discrete decisions $\mathbf{y}^s$ and $\mathbf{c}$ to a given set of values ($\mathbf{y}^{s(l)}, \mathbf{c}^{(l)}$). Therefore, there is no need to introduce the logic constraints (45), (48) and hence the NLP subproblem (P) is equivalent to the well operation and gas lift allocation problem. It must be noted that the solution of the NLP subproblem provides a lower bound on the solution of the MINLP problem since the binary variables $\mathbf{y}^s$ and $\mathbf{c}$ are fixed to values that are not necessarily optimal.

The master problem is formulated from the linearization of the nonlinear constraints (43) and (47) at the solution points of the subproblems ($l = 1, \dots, L$) and relaxation of them to inequalities using the sign of the Lagrange multipliers [17]. It is therefore, a MILP problem. The master problem provides (i) an upper bound to the MINLP problem and (ii) a new set of binary variables $\mathbf{y}^s$ and $\mathbf{c}$. The master MILP problem is as follows:

$$PM\colon\ \max\ \psi(q_{o,i}, q_{g,i}^{\text{inj}}) - (\mathbf{w}_l^p)^{\mathrm{T}}\mathbf{p}_l - (\mathbf{w}_l^q)^{\mathrm{T}}\mathbf{q}_l \tag{49}$$

subject to

$$\mathbf{T}^l \left\{ [\nabla_{\mathbf{x}_i} m_1(\mathbf{x}_i^l, \mathbf{q}_{p.i}^l)\ \nabla_{\mathbf{q}_{p,i}} m_1(\mathbf{x}_i^l, \mathbf{q}_{p.i}^l)] \begin{bmatrix} (\mathbf{x}_i - \mathbf{x}_i^l) \\ (\mathbf{q}_{p.i} - \mathbf{q}_{p.i}^l) \end{bmatrix} \right\} \le \mathbf{p}_l,\ \forall l = 1, \dots, L \tag{50}$$

$$m_2(\mathbf{q}_{p,i}, \mathbf{q}_{p,i,m}, \mathbf{q}_{p,i,s}, \mathbf{q}_{p,s}, \mathbf{x}_i, \mathbf{x}_{i,m}, \mathbf{y}^s) \le 0 \tag{51}$$

$$m_3(\mathbf{q}_{p,i,m}, \mathbf{q}_{p,i,s}, \mathbf{y}^s) \le 0 \tag{52}$$

$$m_4(q_{o,i}, q_{g,i}^{\text{inj}}, q_{o,i,s}, q_{o,i,s}^{\max}, q_{g,i,s}^{\text{inj}} \mathbf{x}_{i,m}, \phi, \mathbf{y}^s, \mathbf{y}^a) = 0 \tag{53}$$

$$\mathbf{T}^l \left\{ [\nabla_{\mathbf{x}_m} m_5(\mathbf{x}_m^l, \mathbf{q}_{p,m}^l)\ \nabla_{\mathbf{q}_{p,s}} m_5(\mathbf{x}_m^l, \mathbf{q}_{p,m}^l)] \begin{bmatrix} \mathbf{x}_m - \mathbf{x}_m^l \\ \mathbf{q}_{p,m} - \mathbf{q}_{p,m}^l \end{bmatrix} \right\} \le \mathbf{q}_l\ ,\ l = 1, \dots, L \tag{54}$$

$$m_6(\mathbf{y}^s, \mathbf{c}) \le 0 \tag{55}$$

$$\sum_{n \in G^l} y_n^s - \sum_{n \in NG^l} y_n^s \le |G^l|\ ,\ \forall l = 1, \dots, L \tag{56}$$

where $\mathbf{w}_l^p$ and $\mathbf{w}_l^q$ are both vectors whose dimension is equal to the number of equations in (50) and (54), respectively. Each element of these vectors is a positive scalar which is greater than the absolute value of the Lagrange multiplier ν_j^l associated with the jth constraint in (50) and (54) at the lth iteration. Moreover, $\mathbf{T}^l$ is a diagonal matrix whose elements are defined as follows:

$$t_{jj}^l = \begin{cases} -1 & \text{if } \nu_j^l < 0 \\ 1 & \text{if } \nu_j^l > 0 \\ 0 & \text{if } \nu_j^l = 0 \,. \end{cases} \tag{57}$$

Furthermore, $\mathbf{p}^l$ and $\mathbf{q}l$ are vectors whose elements are positive slack variables associated with each of the constraints in Eq. (50) and (54), respectively. Finally, the constraint (56) is known as an *integer cut* and is applied to ensure that any pipeline configuration that has already been considered is not selected again. The notion $|G^l|$ denotes the cardinality of the set G^l whose elements are all the structural binary variables $y_n{}^s$ that have a value of 1 at the lth iteration, while NG^l is the set of structural binary variables that have a value of zero at lth iteration.

The MINLP problem terminates when the difference of the best lower bound from the NLP subproblems ($\max_l LB^l$) and the current upper bound from the MILP problem (UB^l) are within a prespectified tolerance ε:

$$\frac{\max_l LB^l - UB^l}{UB^l} \le \varepsilon \tag{58}$$

or when the MILP problem is integer-infeasible. The optimal solution is the one given by the best NLP subproblem.

However, the solution of the MILP problem on the full space of both structural and interpolation binary variables is computationally intensive, since the number of interpolation binary variables becomes very large as the number of wells increases. For instance, a problem with 10 gas lift wells involves about 300 interpolation binary variables. This motivates the need to reformulate the MILP problem (PM), so as to involve only structural and switching binary variables $\mathbf{y}^s$ and $\mathbf{c}$. As mentioned, the master MILP problem is constructed from (i) linearization of the nonlinear constraints, and (ii) relaxation of the nonlinear equality constraints using the sign of Lagrange multipliers. Fortunately, information for both is available from the solution of the NLP subproblem. Consider for instance the case of a gas lift well of type A (29), where the subscript i has been dropped for simplicity. At the optimal point, three adjacent μ coefficients are active (Williams, 1990). The active triplet is assumed to be $(\mu_{j,k}, \mu_{j+1,k}, \mu_{j,k+1})$, without loss of generality. Then the gas lift model (29) can be written as:

$$q_o = \mu_{j,k} q_{o,j,k}^d + \mu_{j+1,k} q_{o,j+1,k}^d + \mu_{j,k+1} q_{o,j,k+1}^d \tag{59a}$$

$$q_g^{\text{inj}} = \mu_{j,k} q_{g,k}^{d,\text{inj}} + \mu_{j+1,k} q_{g,k}^{d,\text{inj}} + \mu_{j,k+1} q_{g,k+1}^{d,\text{inj}} \tag{59b}$$

$$P_m = \mu_{j,k} P_j^{d,m} + \mu_{j+1,k} P_{j+1}^{d,m} + \mu_{j,k+1} P_j^{d,m} \tag{59c}$$

$$\mu_{j,k} + \mu_{j+1,k} + \mu_{j,k+1} = 1 \tag{59d}$$

By substituting Eq. (59d) into (59b) and (59c), both $\mu_{j+1,k}$ and $\mu_{j,k+1}$ are given by:

$$\mu_{j+1,k} = \frac{P_m - P_j^{d,m}}{P_{j+1}^{d,m} - P_j^{d,m}} \tag{60a}$$

$$\mu_{j,k} = \frac{q_g^{\text{inj}} - q_{g,k}^{d,\text{inj}}}{q_{g,k+1}^{d,\text{inj}} - q_{g,k}^{d,\text{inj}}} \,. \tag{60b}$$

Substituting Eq. (60a) and (60b) into (59a), the following equation is obtained:

$$\begin{aligned} q_o = q_{o,j,k} &+ \frac{q_{o,j+1,k}^d - q_{o,j,k}^d}{P_{j+1}^{d,m} - P_j^{d,m}} (P_m - P_j^{d,m}) \\ &+ \frac{q_{o,j,k+1}^d - q_{o,j,k}^d}{q_{g,k+1}^{d,\text{inj}} - q_{g,k}^{d,\text{inj}}} (q_g^{\text{inj}} - q_{g,k}^{\text{inj}}) \,. \end{aligned} \tag{61}$$

Equation (61) is the linearization of the nonlinear gas lift well model, where the derivatives are calculated by forward finite difference formulae. If Eq. (61) replaces Eq. (59), a new NLP subproblem is obtained; then, by applying KKT conditions to both NLP subproblems, it is easy to prove that the Lagrange multiplier of Eq. (59a) is equal to the Lagrange multiplier of Eq. (61). Consequently, the active triplet of μ's is obtained from the solution of the NLP subproblem, along with the Lagrange multiplier; moreover, the new MILP master problem (PM') is formulated:

$$\begin{aligned} PM' \colon \ \max \ & \psi(q_{o,i}, q_{g,i}^{\text{inj}}) \\ & - (\mathbf{w}_l^p)^{\mathrm{T}} \mathbf{p}_l - (\mathbf{w}_l^q)^{\mathrm{T}} \mathbf{q}_l - (\mathbf{w}_l^r)^{\mathrm{T}} \mathbf{r}_l \end{aligned} \tag{62}$$

subject to

$$\mathbf{T}^l \left\{ [\nabla_{\mathbf{x}_i} m_1(\mathbf{x}_i^l, \mathbf{q}_{p.i}^l), \nabla_{\mathbf{q}_{p,i}} m_1(\mathbf{x}_i^l, \mathbf{q}_{p.i}^l)] \begin{bmatrix} (\mathbf{x}_i - \mathbf{x}_i^l) \\ (\mathbf{q}_{p.i} - \mathbf{q}_{p.i}^l) \end{bmatrix} \right\} \leq \mathbf{p}_l \ , \ \forall l = 1, \ldots, L \quad (63)$$

$$m_2(\mathbf{q}_{p,i}, \ \mathbf{q}_{p,i,m}, \ \mathbf{q}_{p,i,s}, \ \mathbf{q}_{p,s}, \mathbf{x}_i, \mathbf{x}_{i,m}, \mathbf{y}^s) \leq 0 \quad (64)$$

$$m_3(\mathbf{q}_{p,i,m}, \mathbf{q}_{p,i,s}, \mathbf{y}^s) \leq 0 \quad (65)$$

$$\mathbf{T}^l \left\{ [\nabla_{\mathbf{x}_i} m_4(\mathbf{x}_i^l, \mathbf{q}_{p.i}^l) \nabla_{\mathbf{q}_{p,i}} m_4(\mathbf{x}_i^l, \mathbf{q}_{p.i}^l)] \begin{bmatrix} (\mathbf{x}_i - \mathbf{x}_i^l) \\ (\mathbf{q}_{p.i} - \mathbf{q}_{p.i}^l) \end{bmatrix} \right\} \leq \mathbf{r}_l \ , \ \forall l = 1, \ldots, L \quad (66)$$

$$\mathbf{T}^l \left\{ [\nabla_{\mathbf{x}_m} m_5(\mathbf{x}_m^l, \mathbf{q}_{p,s}^l) \nabla_{\mathbf{q}_{p,s}} m_5(\mathbf{x}_m^l, \mathbf{q}_{p,s}^l)] \begin{bmatrix} \mathbf{x}_m - \mathbf{x}_m^l \\ \mathbf{q}_{p,s} - \mathbf{q}_{p,s}^l \end{bmatrix} \right\} \leq \mathbf{q}_l \ , \ l = 1, \ldots, L \quad (67)$$

$$m_6(\mathbf{y}^s, \mathbf{c}) \leq 0 \quad (68)$$

$$\sum_{n \in G^l} y_n^s - \sum_{n \in NG^l} y_n^s \leq |G^l| \ , \ \forall l = 1, \ldots, L \, . \quad (69)$$

The MILP problem (PM') involves only structural ($\mathbf{y}^s$) and switching ($\mathbf{c}$) binary variables. Figure 5 depicts the linearization of the nonlinear gas lift model, according to the foregoing analysis.

Solution Procedure

Based on the foregoing sections, the steps of the proposed MINLP optimization strategy for the well scheduling problem are formally presented as follows:

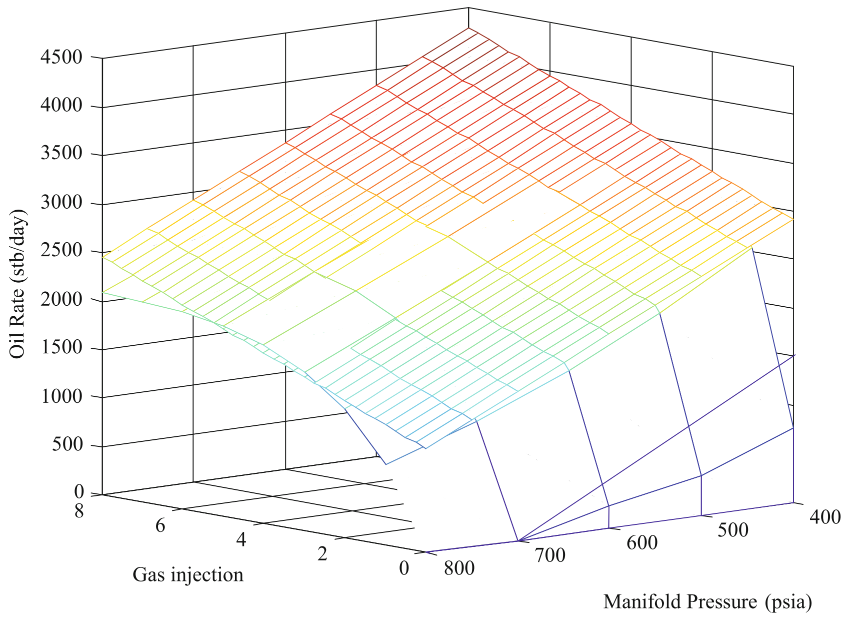
(1) Pre-processing step
1. The reservoir information (productivity index, *GOR* and *WOR*) is updated, using a reservoir simulator.
2. For each naturally flowing well, the manifold pressure and the well oil rate are discretized between a lower and an upper bound. Then, the well model is simulated for each pair of discrete points, and the momentum and energy balances are approximated with piecewise linear functions.
3. For each gas lift well, the manifold pressure and the gas injection rate are discretized between a lower and an upper bound. Then, the well model is simulated for each pair of discrete points, and the momentum and energy balances are approximated with piecewise linear functions.
4. If a naturally flowing well cannot flow towards a separator, then the corresponding logic constraint is incorporated into the formulation.
5. If Vertical Flowing Tables are used, then the approximation of momentum and energy balances in the wells is simpler: there is no need for well simulation using each pair of discrete points, and simple interpolation calculations are used to approximate the momentum and energy balances in the wells.

(2) Processing step
This step involves the solution of the MINLP problem:
1. Set the iteration counter at $l = 0$, and the upper bound at $UB^0 = +\infty$.
2. Solve the NLP subproblem as a sequence of MILP problems, following the algorithm described by Kosmidis [18] 4 to obtain a lower bound (LB^l).
3. Add linearizations and integer cuts cumulatively, and solve the MILP master problem (PM') and update the upper bound (UB^l).
4. If $(UB^l - \max_l LB^l)/(LB^l) \leq \varepsilon$ or the MILP problem is integer-infeasible, then STOP. The optimal structure is the one which corresponds to the best lower bound $\max_l LB^l$. Else, set $l = l + 1$ and go to step (2).2.

(3) Post-processing step
For each well, fix the manifold pressure and the well oil flowrate and perform a a well simulation (based on the system of well equations) to calculate the precise well choke settings.

Example Problems

This section illustrates the performance of the proposed MINLP algorithm in two different example problems.



Mixed Integer Optimization in Well Scheduling, Figure 5
Linearization of a gas lift well model

The first example is a small production network which involves three wells connected to a manifold, and it is used to illustrate the economic impact of incorporating discrete decisions in the well scheduling problem. In the second example, the proposed MINLP optimization strategy is applied to a field consisting of 3 separators, 2 manifolds and 11 naturally flowing wells. To evaluate the economic benefits of the proposed MINLP optimization strategy, heuristic rules are also applied to the same problem for comparison. Finally, the proposed method is applied to an oil field, which consists of 22 (both naturally flowing and gas lift) wells.

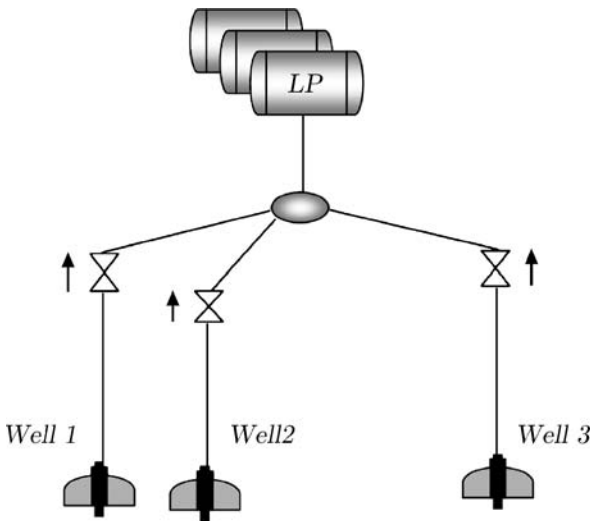
Example 1

The mathematical formulation and the solution procedure developed in this Chapter has been applied to the production network presented in Fig. 6. The well characteristics, separator pressures and capacities are given in Table 1 and 2, respectively. The problem has been formulated as an MINLP problem, where binary variables are used to model the operational status of each (closed or open) well. The MILP problems have been implemented in GAMS [5] and solved using CPLEX® as the MIP solver. The problem involves 3 binary variables, 26 interpolation binary variables and 81 constraints. Initially, the manifold pressure and the well oil flowrate are discretized to construct a piecewise linear approximation of the well model. Then, the initial structure (all wells tied to the manifold) has been evaluated by solving the corresponding NLP subproblem: the optimal solution has thus been determined equal to $LB^1 = 12010$ STB/day. The master MILP problem is then formulated and solved: the MILP problem solution generates a new production network structure, where well 1 is shut in. The new structure has then been evaluated in the NLP subproblem, and a new lower bound equal to $LB^2 = 12104.2$ STB/day has been determined. The algorithm terminates, since the MILP master problem is found to be integer-infeasible. Therefore, the optimal structure involves only wells 2 and 3 connected to the manifold, with their chokes fully open.

A typical heuristic rule for maximization of oil production states that the well chokes must be fully open for oil maximization. The application of this heuristic rule to this particular production network yields an oil production level equal to 11929.2 STB/day. The results from the application of heuristic rules and the proposed strategy are summarized in Table 3 and suggest that: (i) these heuristic rules may lead to suboptimal solutions, and (ii) an increase in oil production of 175 STB/day is observed when the proposed formal

Mixed Integer Optimization in Well Scheduling, Table 1
Well characteristics for a three-well production network (illustrative example)

Reservoir / pipeline parameters	Well 1	Well 2	Well 3	Flowline
Reservoir pressure (psia)	2370	4650	4250	
Productivity index (STB/psia day)	3.0	9.0	3.3	
GOR (SCF/STB)	5100	1900	1600	
WC	0.93	0.165	0.15	
Vertical length (ft)	8000	6000	7000	22000 ft
Horizontal length (ft)	6000	4000	3000	0
Diameter (in)	3 in	3 in	3 in	6 in
Roughness	0.0001	0.0001	0.001	0.0001
Flowrate upper bound (STB/day)	1600	10000	5300	
Flowrate lower bound (STB/day)	200	530	470	



Mixed Integer Optimization in Well Scheduling, Figure 6
Production network structure for Example 1 (illustrative example)

Mixed Integer Optimization in Well Scheduling, Table 2
Surface facilities: separator capacities for Example 1 (illustrative example)

Pressure (psia)	400
Oil Capacity (STB/day)	17000
Gas Capacity (MSCF/day)	33000
Water Capacity (STB/day)	22000

MINLP optimization technique is applied to the well scheduling problem. The above result can be explained by considering the interaction of wells that share a common flowline. This particular three-well network problem has a well with a very high water cut (well 1), as can

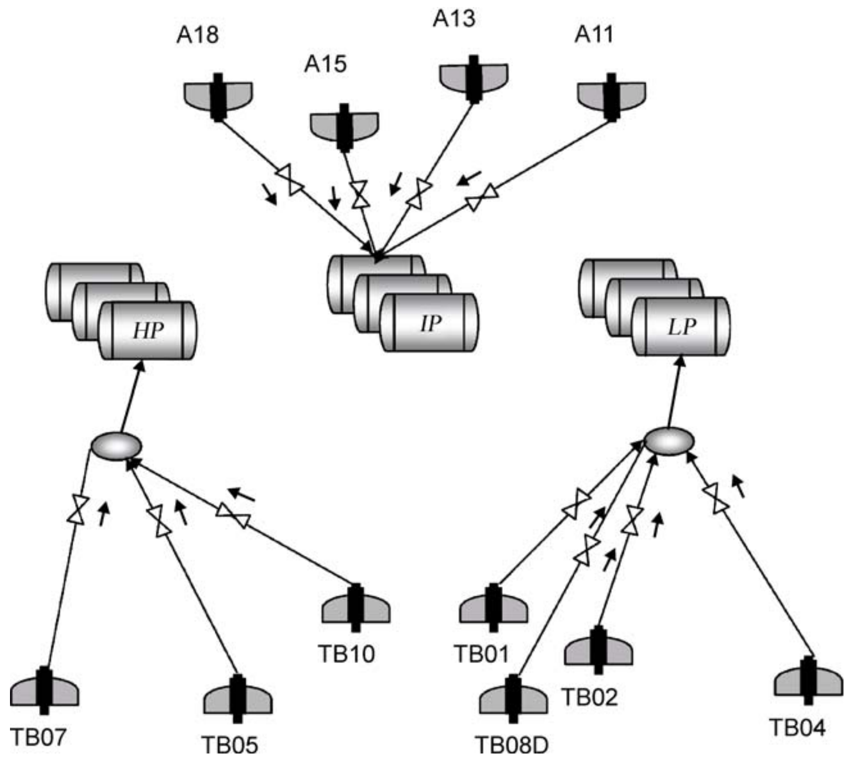
Mixed Integer Optimization in Well Scheduling, Table 3
Comparison of structure and oil production results: heuristics vs. optimization

Structure	Objective function (STB/day)
$(y_1, y_2, y_3) = (1, 1, 1)$	11929.2 (Heuristics)
$(y_1, y_2, y_3) = (0, 1, 1)$	12104.2 (Optimization)

be seen from Table 1: this results in increased back pressure in the manifold flowline, which restricts oil production from wells 2 and 3. By shutting in well 1, the pressure drop in the flowline is reduced: the increased production from wells 2 and 3 thus compensates losses in oil production by shutting in well 1.

Example 2

The proposed MINLP optimization strategy is also applied to an oil field that comprises 11 naturally flowing wells, 2 manifolds and 3 separators: this production network is depicted in Fig. 7. Two types of wells are considered: (i) *type A wells*, designated as TB01, TB02, TB04, TB05, TB07, TB08D and TB10, and (ii) *type B wells*, designated as A11, A13, A15 and A18. All these are naturally flowing wells and their well oil flowrate upper bounds are given in Table 4. The surface facilities consist of a high (HP), an intermediate (IP) and a low (LP) pressure separator, and the respective operating pressures and capacities are summarized in Table 5. Two case studies are considered: the first is a gas coning oil field, while the second is a water coning oil field. The well bore model is generated from a reservoir



Mixed Integer Optimization in Well Scheduling, Figure 7
Production network structure for Example 2a (initial pipeline configuration)

Mixed Integer Optimization in Well Scheduling, Table 4
Maximum flowrate values for wells

TB01	2300 STB/day	TB07	7000 STB/day	A11	4100 STB/day
TB02	4300 STB/day	TB08D	1000 STB/day	A13	4200 STB/day
TB04	2500 STB/day	TB10	7000 STB/day	A15	1800 STB/day
TB05	7500 STB/day			A18	1200 STB/day

Mixed Integer Optimization in Well Scheduling, Table 5
Operating pressures and capacities of separators for Example 2a (gas coning)

	HP separator		IP separator		LP separator	
	Capacity	Optimal	Capacity	Optimal	Capacity	Optimal
Pressure (psia)	1235		460		165	
Oil (STB/day)	15000	12541.9	10000	7191.2	10000	9584.1
Gas (MMSCF/day)	24000	24000	18000	18000	18000	18000
Water (STB/day)	2000	1236.5	4000	2057.8	8000	8000
Total oil production	29317					
NLP (LB)	28567	28910	29317	28735	29020	
MILP (UB)	32104	31580	31210	30920	30067	

Mixed Integer Optimization in Well Scheduling, Table 6
Optimal well flowrates by MINLP optimization (Example 2a)

Well	Q_o (STB/day)	Q_g (MSCF/day)	Q_w (STB/day)
TB01	2300	5574.83	3552.06
TB05	5685.67	10806.03	1123.67
TB08D	Shut in		
TB02	632.906	2223.614	0.0
TB04	836.88	1971.94	2055.46
TB07	6647.67	12543.21	109.594
TB10	5814.3	8229.61	2392.47
A11	4100	9151.44	71.62
A13	1291.18	3930.83	1553.64
A15	1800	4917.72	432.51
A18	208.552	650.76	3.213

simulator using a coning model. Details about this second example (oil flowrate as a function of bottomhole pressure, *GOR* and *WC* for both cases) are presented by Kosmidis [18].

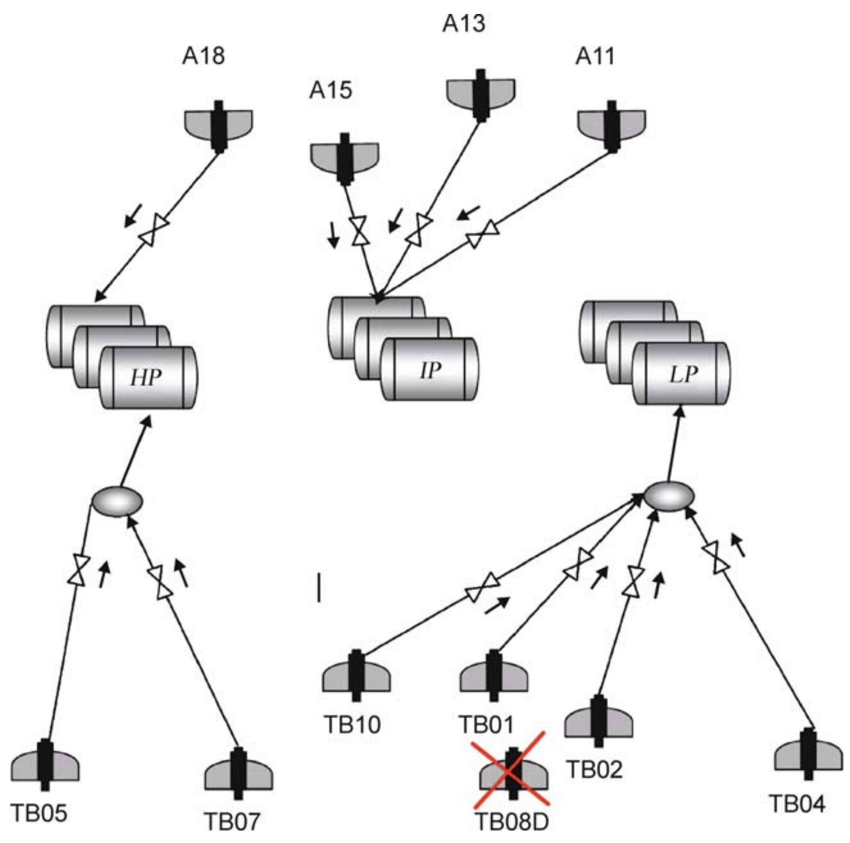
Example 2a (gas coning problem) The initial structure of the production network is shown in Fig. 7, and five (5) well interconnection changes are allowed for wells of type A and type B. The MINLP optimization problem involves 89 binary variables, 260 interpolation binary variables, 924 continuous variables, 1082 constraints and the objective is the maximization of oil production. The optimization requires 5 OA/AP iterations and the total oil production is 29317.2 STB/day; the optimal production network structure is presented in Fig. 8. Table 5 summarizes the amount of oil, gas and water in the separators and the convergence history of the MINLP algorithm; the individual well fluid flowrates are reported in Table 6. A remarkable observation is that the gas capacity of all separators is fully utilized at the optimal operating point, as can be observed from the results of Table 5.

Example 2b (water coning problem) This problem is again solved following the proposed MINLP optimization strategy. The initial structure of the field is presented in Fig. 8; the maximum number of allowable well interconnection changes is seven (7) for wells of type A and type B. The MINLP problem converges in 6 OA/AP iterations and the optimal structure is depicted in Fig. 9. Table 7 presents the amount of oil, gas and water in the separators and the convergence history of the MINLP problem, while well fluid flowrates are reported in Table 8. The results of Table 7 suggest that the production bottleneck of the oil field is the water separator capacity, and the proposed MINLP optimization method manages to allocate and operate the wells in such a way that the available water separator capacity is almost fully utilized. The manifold flowline that is connected to the HP separator in the initial structure (Fig. 7) is reallocated to the IP separator, since the latter has a larger water capacity compared to the HP separator (Table 7).

Heuristic Rules vs. Optimization Examples 2a and 2b are also both solved with heuristic rules, by applying the following procedure:
STEP 0. Consider an initial pipeline structure identical to that of the previous day.
STEP 1. Set the chokes fully open and solve the corresponding production network problem.
STEP 2. If some of the resulting well flowrates from Step 1 violate their upper bounds, then choke back these wells until the respective upper bounds are satisfied.
STEP 3. The following two heuristic rules are applied sequentially (one well at a time):

(i) Choke back the well according to the following heuristic rule: if gas and/or water capacity constraints are violated, then choke back the well with the highest *GOR* and/or *WC*, respectively, until the capacity constraints are satisfied. Terminate or else go to Step 3 (ii).
(ii) Allocate high *GOR* wells to the *HP* separator and high *WC* wells to the LP separator, and go back to Step 1.

It must be noted that: (i) the heuristic rules are applied sequentially, and (ii) the termination criterion is based on the satisfaction of the operator. The results from the application of heuristic rules are based on repetitively applying the procedure described, until the maximum number of allowable interconnection changes is reached. The production network structures resulting from the application of heuristic rules in Examples 2a and 2b are depicted in Fig. 10 and 11, respectively. Tables 9 and 10 summarize the results derived from both MINLP optimization and heuristic strategies. The comparison clearly demonstrates the economic bene-

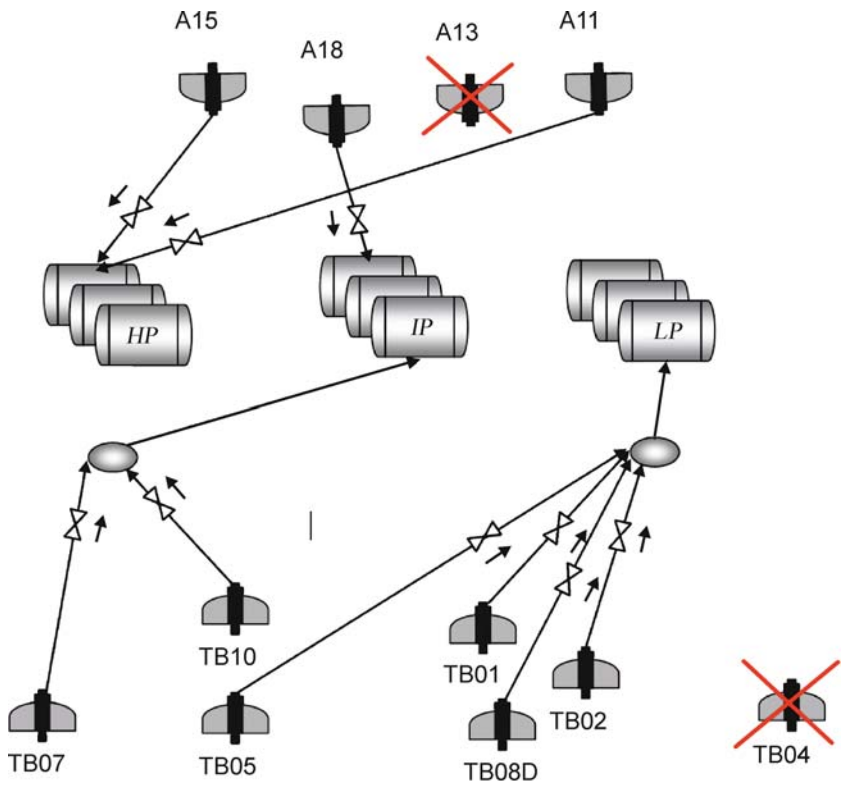


Mixed Integer Optimization in Well Scheduling, Figure 8
Optimal production network structure by MINLP optimization (Example 2a)

Mixed Integer Optimization in Well Scheduling, Table 7
Optimal surface separator capacities by MINLP optimization (Example 2b)

	HP separator		IP separator		LP separator	
	Capacity	Optimal	Capacity	Optimal	Capacity	Optimal
Pressure (psia)	1235		460		165	
Oil (STB/day)	15000	5900	10000	9714.5	10000	9684.4
Gas (MMSCF/day)	24000	14069.2	18000	18000	18000	18000
Water (STB/day)	2000	1926.9	4000	4000	8000	8000
Total oil production	25299					
NLP (LB)	23210	24820	25170	25299	24870	24320
MILP (UB)	29102	28670	27332	26703	26209	26023

fits from the application of the proposed MINLP optimization strategy, which in both examples achieves of up to 10% in oil production. There are many reasons which can explain these superior results: (i) the simplistic nature of heuristic rules, which consider only the individual well *GOR* and *WC*, and neglect other parameters (e. g. productivity index, pipeline length and diameter), (ii) heuristic strategies do not account directly for system interactions, which become important when the wells share a common flowline, and (iii) heuristic methods often have ad hoc or unclear termination criteria.



Mixed Integer Optimization in Well Scheduling, Figure 9
Optimal production network structure by MINLP optimization (Example 2b)

Mixed Integer Optimization in Well Scheduling, Table 8
Optimal well flowrates by MINLP optimization (Example 2b)

	Q_o (STB/day)	Q_g (MSCF/day)	Q_w (STB/day)
TB01	2300	2125.14	3552.1
TB05	5200	10429.3	1874.8
TB08D	757.308	2723.6	375.1
TB02	1427.04	2721.95	2198.01
TB04	Shut in		
TB07	6864.4	12937.2	1893.7
TB10	1691.2	2359.6	1573.4
A11	4100	9151.4	1494.4
A13	Shut in		
A15	1800	4917.7	432.5
A18	1158.8	2703.2	532.9

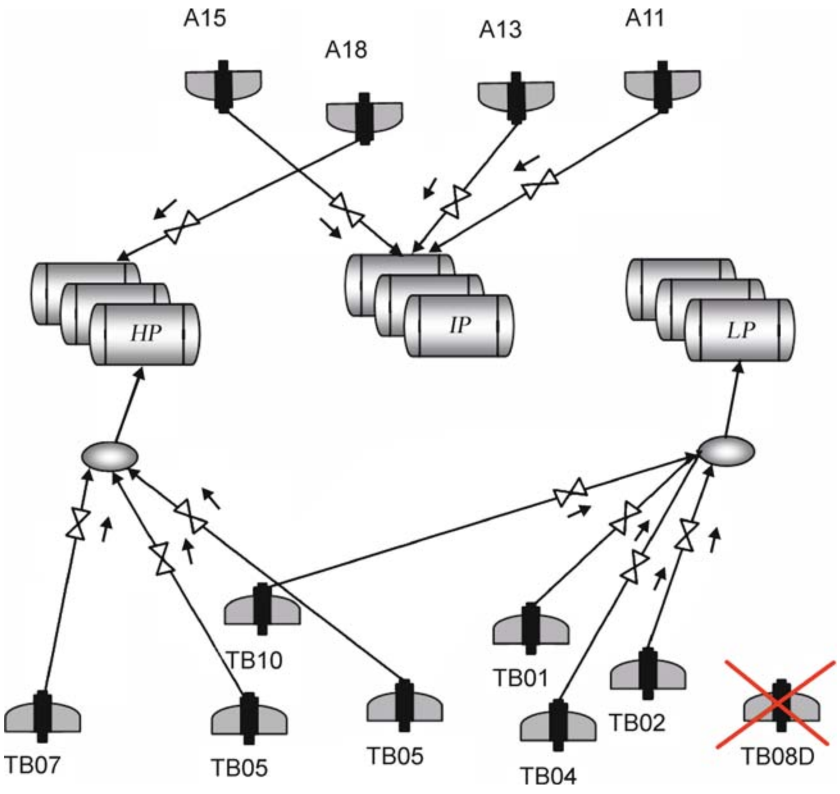
Integration of Reservoir Multiphase Flow Simulation and Optimization

Dynamic oil and gas production systems simulation and optimization is a research trend with a potential to meet the challenges faced by the international oil and gas industry, as has been already demonstrated in a wide variety of publications in the open literature. The multiphase flow in reservoirs and wells governs fuel transport and production, but is mostly handled by algebraic approximations in modern optimization applications: true reservoir state variable profiles (initial/boundary conditions) are generally not known. Nevertheless, oil reservoirs, wells, pipelines, manifolds and surface facilities are all equally important elements of a spatially and temporally distributed complex system, and the potential contribution of CFD methods has not been fully explored so far, even though it is generally recognized that computing accurate reservoir and well state variable profiles can be extremely useful for optimization. This section discusses a strategy for interfacing reservoir simulation (ECLIPSE®) with equation-oriented process optimization (gPROMS®) and presents a relevant application [13].

The complex multiphase flow in oil production fields is of paramount importance. Despite intensive

Mixed Integer Optimization in Well Scheduling, Table 9
Comparison of results: MINLP optimization vs. heuristics (Example 2a)

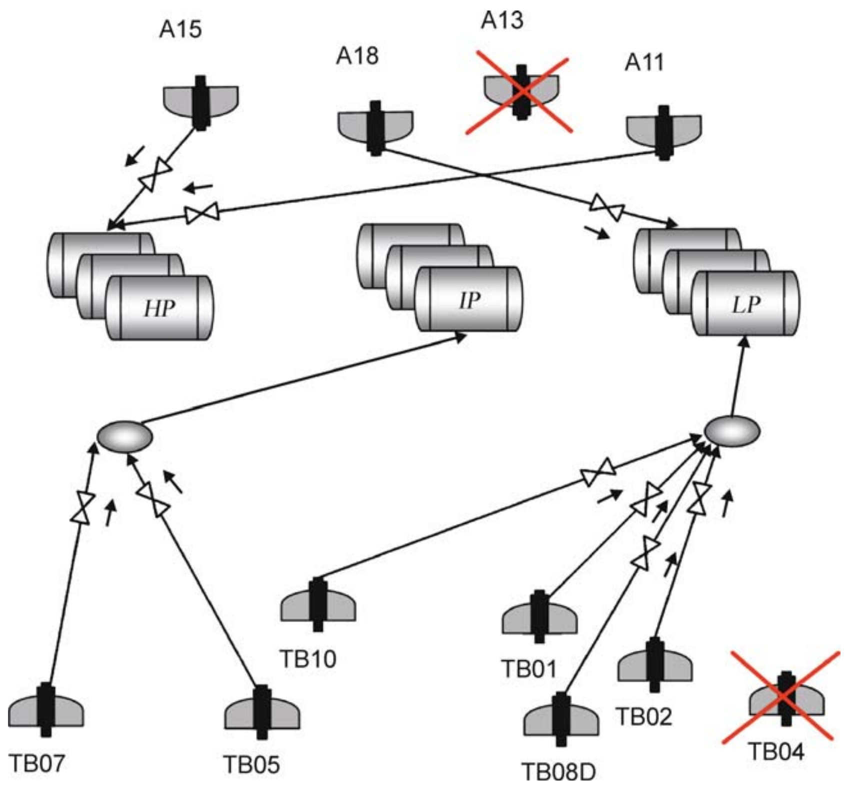
Example 2a (High GORs)	Capacity	Optimization	Heuristics
	Q_o (STB/day)	Q_o (STB/day)	Q_o (STB/day)
LP	10000	9584.15	9004.296
IP	10000	7191.18	7191.186
HP	15000	12541.894	12321.138
Total		29317.2	28516.6
Benefit (STB/day)		800.5 (+2.3%)	



Mixed Integer Optimization in Well Scheduling, Figure 10
Heuristic production network structure (Example 2a)

Mixed Integer Optimization in Well Scheduling, Table 10
Comparison of results: MINLP optimization vs. heuristics (Example 2b)

Example 2b (High WCs)	Capacity	Optimization	Heuristics
	Q_o (STB/day)	Q_o (STB/day)	Q_o (STB/day)
LP	10000	9684.4	7424.407
IP	10000	9714.5	9311.058
HP	15000	5900	5900
Total		25298.8	22635.5
Benefit (STB/day)		2663.3 (+11.8%)	



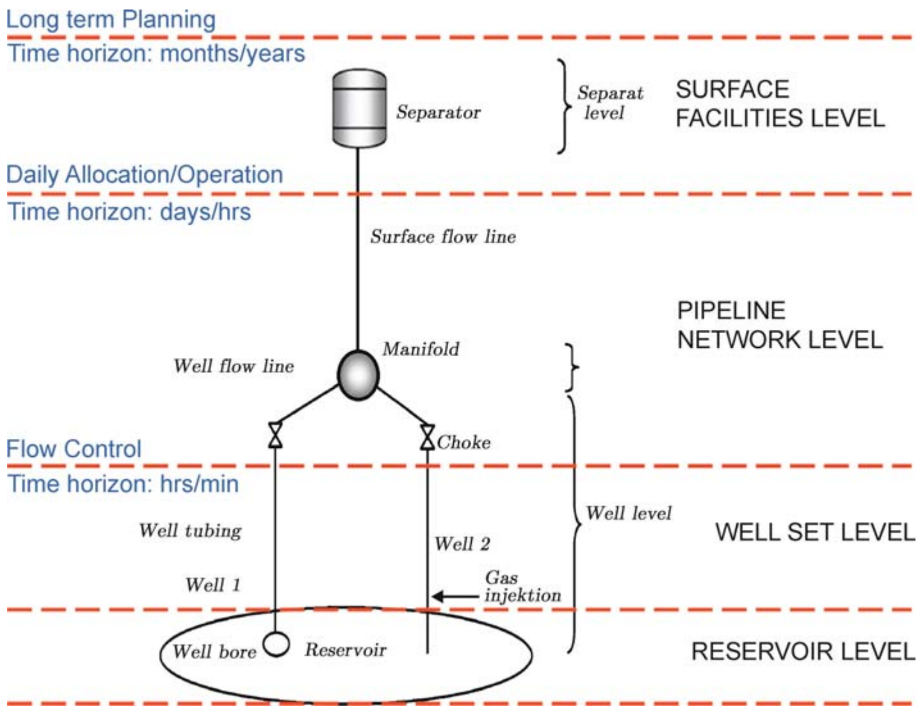
Mixed Integer Optimization in Well Scheduling, Figure 11
Heuristic production network structure (Example 2b)

experimentation and extensive CFD simulations towards improved understanding of flow and phase distribution, commercial optimization applications have not benefited adequately from accurate sub-surface multiphase CFD modeling, and knowledge from field data is not readily implementable in commercial software. Model integration can enable the employment of two-phase reservoir CFD simulation, towards enhanced oil or gas production from depleted or gas-rich reserves, respectively.

The concept of integrated modeling and optimization of oil and gas production treats oil reservoirs, wells and surface facilities as a single (albeit multiscale) system, and focuses on computing accurate reservoir state variable profiles (as initial/boundary conditions). The upper-level optimization can thus benefit from the low-level reservoir simulation of oil and gas flow, yielding flow control settings and production resource allocations. The components of this system are tightly interconnected (well operation, allocation of wells to headers and manifolds, gas lift allocation, control of unstable gas lift wells). These are only some of the problems that can be addressed via this unified framework. Figure 12 presents the concept of integrated modeling of oil and gas production systems.

Literature Review and Challenges for Integrated Modeling and Optimization

A number of scientific publications address modeling and simulation of oil extraction: they either focus on accurate reservoir simulation, without optimization considerations [15,22], or on optimal well planning and operations, with reduced [8,23,29,32] or absent [28,30] reservoir models. A recent paper [16] is the only considering a three-dimensional field topology (without additional flow constraints) for well placement optimization. Computational Field Dynamics (CFD) is a powerful technology, suitable for studying the dynamic behavior of reservoirs for efficient field operation [2]. The MINLP formulation for oilfield production optimization of Kosmidis [19] uses detailed well



Mixed Integer Optimization in Well Scheduling, Figure 12
Integrated modeling concept for oil and gas production systems optimization: illustration of the hierarchy of levels and all production circuit elements

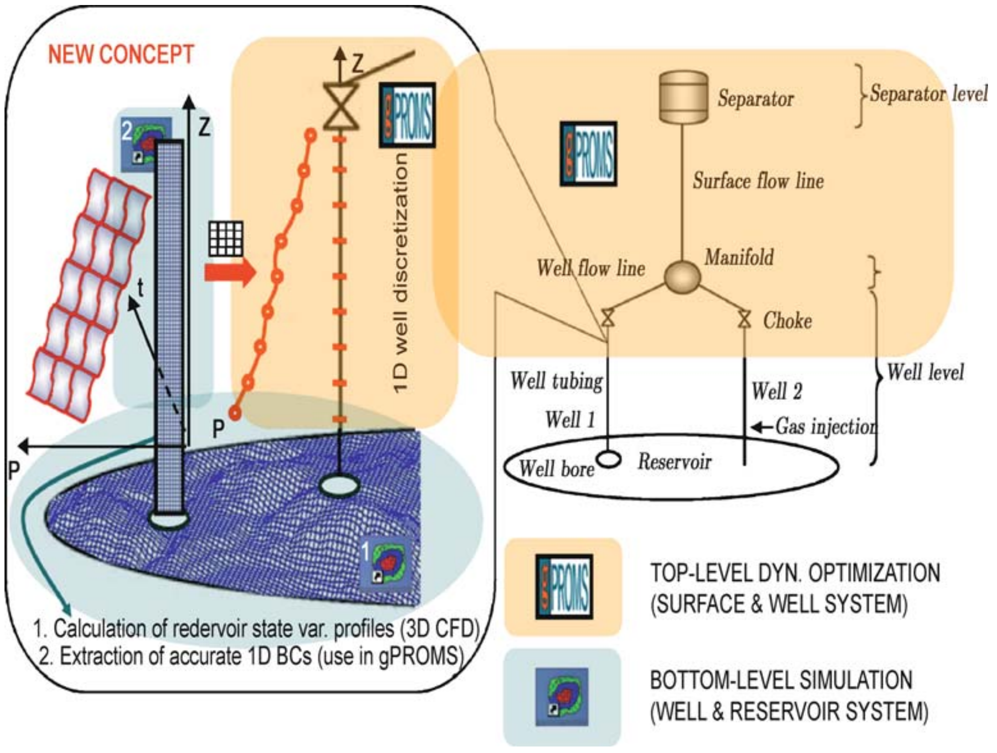
models and serves as a starting point in the case examined in this section. Therein, the nonlinear reservoir behavior, the multiphase flow in pipelines, and surface capacity constraints are *all* considered (multiphase flow is handled by DAE systems, which in turn comprise ODEs for flow equations and algebraics for phys. properties). The model uses a degrees-of-freedom analysis and well bounding, but most importantly approximates each well model with piecewise linear functions (via data preprocessing).

Here, explicit reservoir flow simulation via a dynamic reservoir simulator (ECLIPSE®) is combined with an equation-oriented process optimizer (gPROMS®), towards integrated modeling and optimization of a literature problem 13. An asynchronous fashion is employed: the first step is the calculation of state variable profiles from a detailed description of the production system (reservoir) via ECLIPSE®. This is possible by rigorously simulating the multiphase flow within the reservoir, with real-world physical properties (whose extraction is laborious [7]). These dynamic state variable profiles (pressure, oil, gas and water saturation, flows) are a lot more accurate than piecewise linear approximations [18], serving as initial conditions for the higher-level dynamic optimization model (within gPROMS®). Crucially, these profiles constitute major sources of uncertainty in simplified models. Considering the oil and gas pressure drop evolution within the reservoir and along the wells, one can solve single-period or multi-period dynamic optimization problems that yield superior optima, because piecewise linear pressure underestimation is avoided. While integrating different levels (sub-surface elements and surface facilities – Fig. 12) is vital, interfacing CFD simulation with MINLP optimization is here pursued in an asynchronous fashion (given the computational burden for CFD nested within MINLP).

The concept of integrated modeling and optimization is illustrated in Fig. 13.

Problem Definition and Model Formulation

Dynamic CFD modeling for explicit multiphase flow simulation in reservoirs and wells comprises a large


NEW CONCEPT
Z
P
t
1D well discretization
gPROMS
Separator
Separator level
Surface flow line
Well flow line
Manifold
Choke
Well tubing
Well 1
Well 2
Gas injection
Well level
Well bore
Reservoir
1. Calculation of redervoir state var. profiles (3D CFD)
2. Extraction of accurate 1D BCs (use in gPROMS)
TOP-LEVEL DYN. OPTIMIZATION (SURFACE & WELL SYSTEM)
BOTTOM-LEVEL SIMULATION (WELL & RESERVOIR SYSTEM)

Mixed Integer Optimization in Well Scheduling, Figure 13
Integrated modeling and optimization of oil and gas production systems: illustration of the explicit consideration of multiphase flow within reservoirs and wells

number of conservation laws and constitutive equations for closure: Table 1 presents only the most important ones, which are implemented in ECLIPSE®. The *black-oil model* [25] is adopted in this study, to manage complexity. More complicated, *compositional models* are widely applied [2], accounting explicitly for different hydrocarbon real- or pseudo-species concentrations. A black-oil model allows for multiphase simulation via only 3 phases (oil, water, gas):

Multiphase flow CFD model equations (Nomenclature [19]):
Oil:

$$\nabla\left[\frac{k\,k_{ro}}{\mu_o B_o}\nabla(P_o+\rho g h)\right]+q_o=\frac{\partial}{\partial t}\left(\phi\frac{S_o}{B_o}\right) \quad (70)$$

Water:

$$\nabla\left[\frac{k\,k_{rw}}{\mu_w B_w}\nabla(P_w+\rho g h)\right]+q_w=\frac{\partial}{\partial t}\left(\phi\frac{S_w}{B_w}\right) \quad (71)$$

Gas:

$$\begin{aligned}&\nabla\left[\frac{k\,k_{rg}}{\mu_g B_g}\nabla(P_g+\rho g h)\right]\\&+\nabla\left[R_s\frac{k\,k_{ro}}{\mu_o B_o}\times\nabla(P_o+\rho g h)\right]+q_g\\&=\frac{\partial}{\partial t}\left(\phi\frac{S_g}{B_g}+R_s\frac{\phi S_o}{B_o}\right)\end{aligned} \quad (72)$$

Total pressure gradient:

$$\frac{\mathrm{d}P}{\mathrm{d}x}=-g\rho_m(x)\sin(\theta)-\frac{\tau_w(x)S}{A} \quad (73)$$

Capillary pressure (oil/gas):

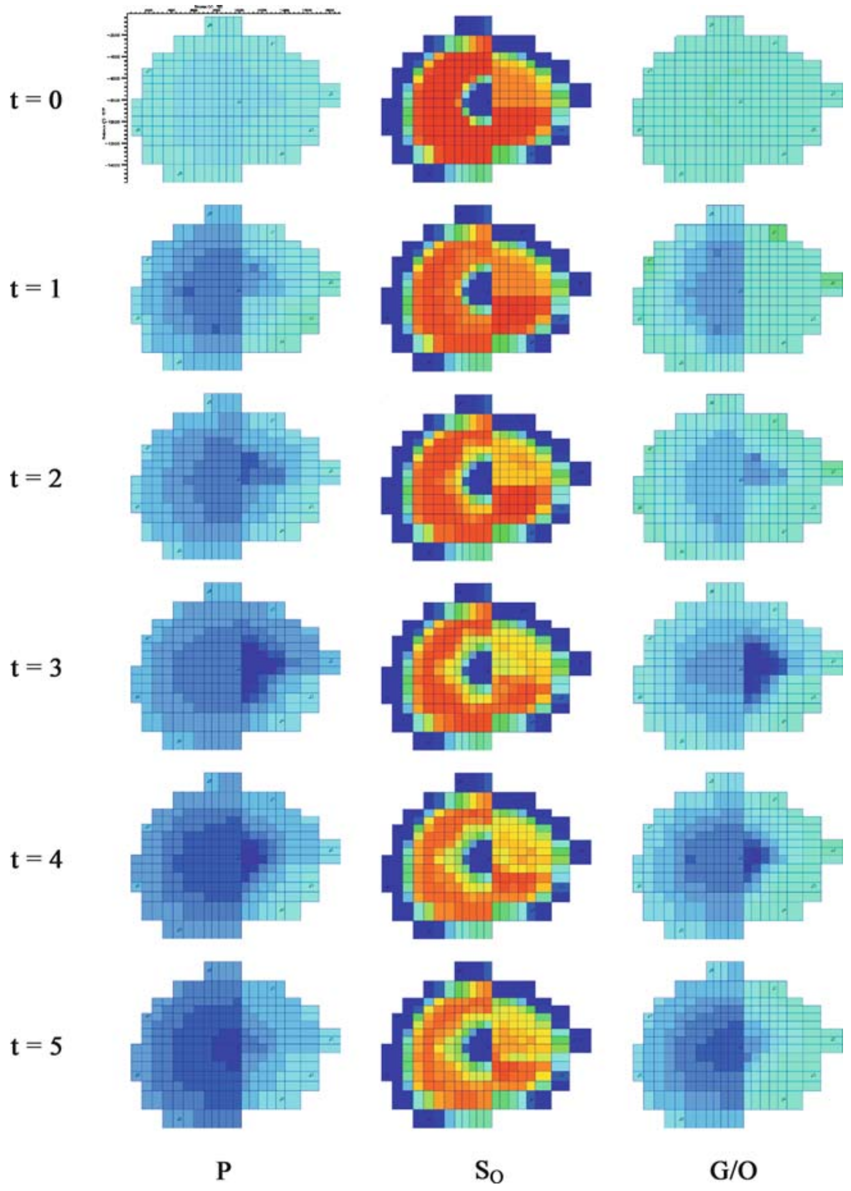
$$P_{\text{cog}}(S_o,S_g)=P_o-P_g \quad (74)$$

Capillary pressure (oil/water):

$$P_{\text{cow}}(S_o,S_w)=P_o-P_w \quad (75)$$

Multiphase mixture saturation:

$$S_o+S_w+S_g=1 \quad (76)$$



Mixed Integer Optimization in Well Scheduling, Figure 14
Temporal evolution of pressure, oil saturation and gas/oil ratio in an oilfield: the gradual depletion of oil in reservoirs is explicitly considered for optimization (t:yr)

Multiphase mixture density:

$$\rho_m(x) = \rho_l(x)E_l(x) + \rho_g(x)E_g(x) \tag{77}$$

Multiphase mixture viscosity:

$$\mu_m(x) = \mu_l(x)E_l(x) + \mu_g(x)E_g(x) \tag{78}$$

Multiphase mixture sup. velocity:

$$U_m(x) = \frac{\rho_l(x)}{\rho_m(x)}U_{sl}(x) + \frac{\rho_g(x)}{\rho_m(x)}U_{sg}(x) \tag{79}$$

Multiphase mixture holdup closure:

$$E_g(x) + E_l(x) = 1 \tag{80}$$

Drift flux model (gas holdup):

$$E_g = f_d(U_{sl}, U_{sg}, \text{ mixture properties}) \tag{81}$$

Choke model (for well & valve i):

$$q_{L,i} = f_c(d_i, P_i(x_{ch}^-), P_i(x_{ch}^+), c_i, q_{g,i}, q_{w,i}) \tag{82}$$

Mixed Integer Optimization in Well Scheduling, Table 11
Oil production optimization by explicit CFD simulation boundary conditions

Example 2a, Kosmidis et al.[20]	Total capacity	Via performance indices	With explicit reservoir simulation
Oil production (STB/day)	35000	29317.2	30193.7 (+2.9%)
Gas production (MSCF/day)	60000	60000	60000
Water production (STB/day)	14000	11294.3	11720.1 (+3.8%)

Choke setting (for well & valve i):

$$c_i = \max(c_c, P_i(x_{ch}^-), P_i(x_{ch}^+)) \qquad (83)$$

Performance (flow vs. pressure):

$$q_{j,i} = f_j(P_{\text{wf},j,i}), \ \forall i \in I, \ \forall j \in \{o, w, g\} \, . \qquad (84)$$

Reduced (1D) multiphase flow balances were solved using a fully implicit formulation and Newton's method, but only for the wells and not for the reservoir [18]. The present section uses: (a) explicit reservoir and well 3D multiphase flow simulation, (b) elimination of Eq. (84) (performance relations/preprocessing obsolete due to CFD), (c) CFD profiles as initial conditions (asynchronous fashion) for dynamic optimization. The MINLP optimization objective (maximize oil production) and model structure is adopted from the literature [20] via a gPROMS®–SLP implementation. Adopting an SQP strategy can increase robustness as well as computational complexity.

Reservoir Multiphase Flow Simulation Results

Dynamic multiphase flow simulation results (ECLIPSE®) are presented in Fig. 14.

Oil Production Optimization Results

Dynamic optimization via explicit CFD simulation of a particular oil field problem can improve on results from MINLP optimization: the comparison is presented in Table 11.

Conclusions

A novel MINLP optimization formulation for the well scheduling problem has been proposed in this Chapter: the optimal connectivity of wells to manifolds and separators is treated simultaneously with the optimal well operation and gas lift allocation. The algorithm avoids examining infeasible connections of wells to manifolds or separators by incorporating appropriate integer cuts in the formulation: these, along with the incorporation of operational logic constraints pertinent to the maximum number of well switches, lead to satisfactory computational performance: convergence has been achieved in less then 6 iterations in all cases examined. The business value of the new MINLP formulation has been investigated by comparing the proposed method with established heuristic rules, and an increase of up to 10% in oil production has been observed for the cases studied [18].

The combination of dynamic multiphase CFD simulation and MINLP optimization has the potential to yield improved solutions towards efficiently maximizing oil production. This Chapter also addresses integrated oilfield modeling and optimization, treating the oil reservoirs, wells and surface facilities as a combined system: most importantly, it stresses the benefit of computing accurate state variable profiles for reservoirs via CFD. Explicit CFD simulations via a dynamic reservoir simulator (ECLIPSE®, Schlumberger) are combined with equation-oriented process optimization software (gPROMS®, PSE): the key idea is to use reduced-order copies of CFD profiles for dynamic optimization. The literature problem solved shows that explicit use of CFD results in optimization yields improved optima at additional cost (CPU cost *and* cost for efficient separation of the additional water; the percentage difference is due to accurate reservoir simulation). These can also be evaluated systematically for larger case studies under various conditions [14].

References

1. Addington DV (1981) An approach to gas coning correlations for a large grid cell reservoir simulator. J Pet Eng 33:2267–2274
2. Aziz K, Settari A (1979) Petroleum Reservoir Simulation. Appl Sci Publ, London
3. Benders JF (1962) Partitioning procedures for solving mixed variables programming problems. Numer Math 4:238–252

4. Brill JP, Mukherjee H (1999) Multiphase Flow in Wells. SPE Monograph, Henry L Doherty Series (17), Richardson
5. Brooke A, Kenderick D, Meeraus A (1992) GAMS: A User Guide. Scientific Press, Redwood City
6. Duran MA, Grossmann IE (1986) A mixed integer nonlinear programming algorithm for process system synthesis. AIChE J 32:592–606
7. Economides M, Hill AD, Ehlig-Economides CA (1994) Petroleum Production Systems. Prentice Hall, Englewood Cliffs
8. Fang WY, Lo KK (1996) A generalized well management scheme for reservoir simulation. Soc Pet Eng Pap SPE 29124:116–120
9. Fentor DJ (1984) A multilevel well management program for modeling offshore fields. Soc Pet Eng Pap SPE 12964:75–82
10. Floudas CA (1995) Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. Oxford University Press, Oxford
11. Geoffrion AM (1972) Generalized Benders decomposition. Optim Theory App 10:237–260
12. GeoQuest (2000) Eclipse 300 Technical Description 2000A. GeoQuest, Schlumberger Information Solutions (SIS), Houston
13. Gerogiorgis DI, Georgiadis MC, Bowen G, Pantelides CC, Pistikopoulos EN (2006) Dynamic oil and gas production optimization via explicit reservoir simulation. In: Marquardt W, Pantelides C (eds) Proceedings of ESCAPE-16/PSE2006. Elsevier, Amsterdam, pp 179–184
14. Gerogiorgis DI, Pistikopoulos EN (2006) Wells-to-tankers: Dynamic oil and gas production optimization via explicit reservoir CFD simulation. In: Proceedings of the PRES-CHISA 2006 Conference (CD) Prague
15. Hepguler G, Barua S, Bard W (1997) Integration of field surface and production network with a reservoir simulator. Soc Pet Eng Pap SPE 38937:88–93
16. Ierapetritou MG, Floudas CA, Vasantharajan S, Cullick AS (1999) Optimal location of vertical wells: decomposition approach. AIChE J 45:844–859
17. Kocis GR, Grossmann IE (1987) Relaxation strategy for the structural optimization of process flowsheets. Ind Eng Chem Res 45:1869–1880
18. Kosmidis VD (2003) Integrated Oil and Gas Production Optimization. PhD Thesis, Department of Chemical Engineering, Imperial College London
19. Kosmidis VD, Perkins JD, Pistikopoulos EN (2004) Optimization of well oil rate allocations in petroleum fields. Ind Eng Chem Res 43:3513–3527
20. Kosmidis VD, Perkins JD, Pistikopoulos EN (2005) A mixed integer optimization formulation for the well scheduling problem on petroleum fields. Comput Chem Eng 29:1523–1541
21. Litvak ML, Darlow BL (1995) Surface network and well tubinghead pressure constraints in compositional simulation. Soc Pet Eng Pap SPE 29125:325–336
22. Litvak ML, Clark AJ, Fairchield JW, Fossum MP, McDonald CJ, Wood ARO (1997) Integration of Prudhoe Bay surface pipeline network and full field reservoir models. Soc Pet Eng Pap SPE 38885:435–443
23. Lo KK (1992) Optimum lift-gas allocations under multiple production constraints. Soc Pet Eng Pap SPE 26017
24. Lo KK, Starley GP, Holden CW (1995) Application of linear programming to reservoir development evaluations. Soc Pet Eng Pap SPE 26637:52–58
25. Peaceman DW (1977) Fundamentals of Numerical Reservoir Simulation. Elsevier, Amsterdam
26. Process Systems Enterprise (PSE) Ltd (2000) gPROMS® Advanced User Guide. PSE, London
27. Pucknell JK, Mason JNE, Vervest EG (1993) An evaluation of recent mechanistic models of multiphase flow for predicting pressure drops in oil and gas wells. Soc Pet Eng Pap SPE 26682:1–11
28. Saputelli L, Malki H, Canelon J, Nikolaou M (2002) A critical overview of artificial neural network applications in the context of continuous oil field optimization. Soc Pet Eng Pap SPE 77703:1–11
29. Stewart G, Clark AC, McBride SA (2001) Field-wide production optimization. Soc Pet Eng Pap SPE 59459:1–10
30. Van den Heever SA, Grossmann IE (2000) An iterative aggregation/disaggregation approach for solution of an MINLP oilfield infrastructure planning model. Ind Eng Chem Res 39:1955–1971
31. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer approximation method for MINLP optimization. Comput Chem Eng 14:769–782
32. Wang P, Litvak M, Aziz K (2002) Optimization of production from mature fields. Proceedings of the 17th World Petroleum Congress Rio de Janeiro, 2002
33. Williams HP (1990) Model Building in Mathematical Programming. Wiley, Chichester

Mixed Integer Programming/Constraint Programming Hybrid Methods

ABDERRAHMANE AGGOUN[1], CHRISTOS MARAVELIAS[2], ALKIS VAZACOPOULOS[3]
[1] KLS-OPTIM, Villebon sur Yvette, France
[2] University of Wisconsin – Madison, Madison, USA
[3] Dash Optimization, Englewood Cliffs, USA

Article Outline

Background

Mixed-Integer Programming

Mixed-Integer Programming (MIP) [5] emerged in the mid 1950s as an extension of Linear Programming (LP) to include both integer and continuous variables. It was developed to address a variety of problems (facility location, scheduling, design of plants and networks, etc.) where discrete decisions needed to be made. There are two main algorithms used to solve MIP models: branch-and-bound [5,31] and cutting planes. When the two solution methods are combined we have the branch-and-cut algorithm, where cutting planes are added until either an integral solution is found or it becomes impossible or too expensive to find another cutting plane. In the latter case, a traditional branch operation is performed and the search for cutting planes continues for the subproblems. Balas developed an algorithm for 0–1 problems to obtain dual bounds and check primal feasibility [3]. The idea of cutting planes was originally proposed by Gomory in [17], and a cutting plane algorithm was presented by Gomory in [18]. A general procedure for bounded programs was proposed by Chvatal in [13].

The results of Edmonds and Fulkerson in the late 1960s led several authors to propose other, specific types of cutting planes: cover inequalities [4,5], flow cover inequalities [4], and GUB constraints [51]. Due to the incorporation of these theoretical results, the efficiency of the commercial solvers has greatly been enhanced during the last decade. Advances in preprocessing, more sophisticated branching and node selection rules, as well as the use of primal heuristics have also contributed to the improvement of MIP solvers. Special techniques have also been used extensively for the solution of MIP problems, when the set of constraints exhibits a special structure. The most popular of these schemes are Benders decomposition [9] and Lagrangean relaxation [16,19]. More information on MIP can be found in [38], and [52], while an exposition in recent progress in solution techniques for MIP models can be found in [30].

Constraint Programming

Constraint Programming [24,47] is a relatively new modeling and solution paradigm that was originally developed to solve feasibility problems, but it has been extended to solve optimization problems as well. Constraint Programming (CP) has emerged as a very interesting sub-field of logic programming that aims at combining the declarative aspect of logic programming and constraint solving in an efficient problem solving environment [29]. Optimization problems in Constraint Programming are solved as Constraint Satisfaction Problems (CSP), where we have a set of variables, a set of possible values for each variable (domain) and a set of constraints among the variables. Constraints are solved with methods and advanced techniques originating in various areas, from Artificial Intelligence, Operations Research and Discrete Mathematics. The computation domains handled by CP solvers are quite diverse, including Boolean algebra, linear programming, finite domains, and list and set handling. Successful industrial applications were implemented with CP solvers over finite domains in production planning, scheduling and resource applications [44]. Finite domain constraints are expressed over variables, which range over a finite set of possible values. Constraints may be arithmetic, symbolic or global constraints [1] that have been developed to efficiently model and solve complex problems. A CP program is usually structured as follows: (1) declaration of decision variables, (2) constraints and (3) the enumeration/optimization. The question to be answered is as follows: Is there an assignment of values to variables that satisfy all constraints? Constraint Programming is very expressive as continuous, integer, as well as boolean variables are permitted and moreover, variables can be indexed by other variables. A CP problem can be seen as a network of constraints. As soon as some information becomes available at some points in this network, constraints are invoked to check consistency and to remove inconsistent values by applying efficient handling methods. The new domain reductions are propagated through the network. The solution of CP models is based on performing constraint propagation at each node by reducing the domains of the variables. If an empty domain is found the node is pruned. Branching is performed whenever a domain of an integer, binary or boolean variable has more than one element, or when the bounds of the domain of a continuous variable do not lie within a tolerance. Whenever a solution is found, or a domain of a variable is reduced, new constraints are added. The search terminates when no further nodes must be examined.

The effectiveness of CP depends on the propagation mechanism behind the constraints. Thus, even though many constructs and constraints are available, not all of them have efficient propagation mechanisms. For some problems, such as scheduling, propagation mechanisms have been proven to be very effective. Some of the most common propagation rules for scheduling are the "time-table" constraint [32], the "disjunctive-constraint" propagation [6,45], the "edge-finding" [12,39] and the "not-first, not-last" [7]. Constrained-based scheduling algorithms can be found in [8]. General information on CP can be found in [24,27,36,47].

Methods

Several authors have compared MIP and CP based approaches for solving a variety of problems [21,26], and the main findings are as follows:

- MIP based techniques are very efficient when the LP relaxation is tight and the models have a structure that can be effectively exploited.
- CP based techniques are better for highly constrained discrete optimization problems.

Since the two approaches appear to have complementary strengths, in order to solve difficult problems that are not effectively solved by either of the two, several researchers have proposed models that integrate the two paradigms. The integration between MIP and CP can be achieved in two ways [26,48]:

1 By combining MIP and CP constraints into one hybrid model. In this case a hybrid algorithm that integrates constraint propagation with linear programming in a single search tree is also needed for the solution of the model (e. g. see [21,42]).

2 By decomposing the original problem into two subproblems: one MIP and one CP subproblem. Each model is solved separately and information obtained while solving one subproblem is used for the solution of the other subproblem [11,28].

Bockmayr and Kasper [10] have presented a unifying framework, called *Branch and Infer*, which can be used for the development of various integration schemes. Hooker et al. [25] have proposed a new modeling paradigm to perform efficient integration of MIP and CP techniques. In general, it is not clear whether an integration strategy performs better than a standalone MIP or CP approach, especially when the problem at hand is solved effectively by one of the two approaches. For some problems, however, the integration of the two approaches has led to significant computational improvements. Common integration schemes include the derivation of cuts for MIP formulations using CP techniques, the use of CP to accelerate column generation, and the use of CP local search to solve MIP scheduling problems. Integration schemes are described in [21,23,26,27,37], and [48].

MIP/CP Hybrid Schemes are particularly successful for scheduling problems that often arise in manufacturing, chemical and food industry, in transportation industries and in computing environments. To solve a scheduling problem one has to (i) allocate limited resources to tasks, and (ii) sequence the tasks allocated to a single resource. We will refer to the first set of decisions as the *assignment* problem, and the second set of decisions as the *sequencing* problem.

While heuristic methods are widely used, rigorous optimization methods have also been studied. To solve some hard scheduling problems to optimality, several authors have proposed MIP/CP hybrid schemes that exploit the complementary strengths of Mathematical and Constraint Programming. The main idea behind these approaches is to solve a relaxed MIP model to determine the allocation of machines to tasks, and use CP to check the feasibility of a given assignment and to generate cuts that are added in the relaxed MIP model. Thus, the complementary strengths of the two methods are combined: Mathematical Programming is used for optimization (i. e. identify potentially good assignments) and Constraint Programming to check feasibility.

Applications

A scheduling problem that has been widely studied using hybrid schemes is the Multi-Machine Assignment Scheduling Problem (MMASP) with Release and Due Times. In this problem a set I of N jobs have to be processed on a set J of M machines; the processing of job

$i \in I = \{1, \ldots N\}$ on any machine $j \in J = \{1, \ldots M\}$, must start after its release time r_i and must be completed before its due time d_i; the processing time and processing cost of job $i \in I$ on machine $j \in J$ are P_{ij}, and C_{ij} respectively. The objective is to minimize the total processing cost. The MMSAP was first studied by Hooker et al. [22] in a hybrid optimization framework.

A MIP model (M) for the MMASP consists of constraints (1)–(6):

$$\min Z = \sum_{i \in I} \sum_{j \in J} C_{ij} x_{ij} \tag{1}$$

$$\sum_{j \in J} x_{ij} = 1 \quad \forall i \in I \tag{2}$$

$$s_i \geq r_i \quad \forall i \in I \tag{3}$$

$$s_i + \sum_{j \in J} P_{ij} x_{ij} \leq d_i \quad \forall i \in I \tag{4}$$

$$y_{ii'} + y_{i'i} \geq x_{ij} + x_{i'j} - 1 \quad \forall j \in J, \forall i \in I, \forall i' \in I | i < i' \tag{5}$$

$$s_i + \sum_{j \in J} P_{ij} x_{ij} \leq s_{i'} + M(1 - y_{ii'}) \quad \forall i \in I, \forall i' \in I | i \neq i' \tag{6}$$

where binary x_{ij} is 1 if job i is assigned to machine j, binary $y_{ii'}$ is 1 is job i is scheduled before job i' in the same machine, and s_i is the start time of processing of job i.

Constraint (2) ensures that each job is processed on exactly one machine. Constraints (3) and (4) restrict each job to start after its release, and finish before its due time, respectively. Constraint (5) imposes the condition that if both jobs i and i' are assigned to the same machine j (i. e. $x_{ij} + x_{i'j} - 1 = 1$), then jobs i and i' must be sequenced (i. e. $y_{ii'} = 1$ or $y_{i'i} = 1$). Constraint (6) is a big-M sequencing constraint that is active when $y_{ii'}$ is 1.

Hooker et al. [22] and Jain and Grossmann [28] showed that model (M) is not efficient, due to the poor LP relaxation caused by the big-M sequencing constraint (6). Furthermore, they showed that standalone CP models are not efficient either, due to the large number of different assignments. To overcome this, the authors proposed a scheme where an IP master problem and a CP subproblem are solved iteratively. The IP master problem is a relaxation of model (M) and it is used to determine an assignment. The CP subproblem is used to check feasibility of the current assignment; if infeasible, integer cuts are added and the IP master problem is re-solved; if feasible, the subproblem gives a feasible sequence, and the algorithm terminates. The IP master problem consists of constraints (1)–(2), (7) and the integer cuts that are added at each iteration. Constraint (7) is used to eliminate infeasible assignments:

$$\sum_{i \in I} P_{ij} x_{ij} \leq \max_{i \in I}\{d_i\} - \min_{i \in I}\{r_i\} \quad \forall j \in J \tag{7}$$

The IP master problem does not include the sequencing binary variables $y_{ii'}$ and big-M constraint (6), it is solved fast, and at iteration k, yields a complete job-machine assignment $\boldsymbol{x}^k$. The CP subproblem is then used to check whether the current assignment $\boldsymbol{x}^k$ is feasible. At each iteration k, the set I_j^k of jobs assigned on machine $j \in J$, the processing time $\bar{P}_i^k$ of each job, and the domain D_i^k for the start time of job i (i. e. $s_i \in D_i^k$) are given by (8)–(10), respectively:

$$I_j^k = \{i | x_{ij}^k = 1\} \quad \forall j \in J \tag{8}$$

$$\bar{P}_i^k = \sum_{j \in J} P_{ij} x_{ij}^k \quad \forall i \in I \tag{9}$$

$$D_i^k = [r_i, d_i - \bar{P}_i^k] \quad \forall i \in I \tag{10}$$

Thus, the CP subproblem reduces to |J| one-machine independent problems, and for each one of these problems we try to find a sequence of jobs in I_j^k that satisfies constraint (10) and the non-overlapping of jobs assigned to machine k (see (5) and (6)). This problem can be solved using the global constraint *cumulative* [1], and various propagation techniques (time-table, disjunctive, edge-finding, etc.).

$$\text{cumulative}_{i \in O}\left((s_i, d_i, r_i),\ l,\ e\right) \tag{11}$$

The basic version of cumulative, (see detailed examples in [2]) takes 3 arguments, argument 1 is the set of operations O where each operation is characterized by three parameters, which can be either domain variables or values; the starting time s_i, the duration d_i, and the amount of some resource r_i used by the operation. The second argument l is the upper bound on the resource consumption. The third argument e is the completion time. In this case, Eq. (11) can be written as follows:

$$\text{cumulative}_{i \in I_j}\left(\left(s_i, \bar{P}_i^k, 1\right), 1, \max_{i \in I_j}\{d_i\}\right) \tag{12}$$

The global cumulative constraint is satisfied if the following conditions hold:

$$\sum_{i \in O: s_i \le t \le s_i + d_i, t \in 1..l} r_i \le l \quad AND \quad \max_{i \in O}(s_i + d_i) \le e \tag{13}$$

If there is no sequence for machine j that satisfies constraint (12), then the current assignment $\boldsymbol{x}^k$ is infeasible. For every infeasible one-machine problem we add the following integer cut in the cut-pool of the master problem:

$$\sum_{i \in I_j^k} x_{ij} \le |I_j^k| - 1 \tag{14}$$

If the IP master problem is solved to optimality, the lower bound provided by the optimal solution Z^k of the IP is non-decreasing, and the first feasible assignment is the assignment that yields the optimal solution with a minimum assignment cost. A schematic of the proposed algorithm is given in Fig. 1. The hybrid iterative approach was shown to be considerably faster than standalone MIP and CP models.

The above hybrid decomposition can also be implemented in a branch-and-cut framework (B&C), where the IP master problem is not solved to optimality before adding cuts. In the B&C framework, cuts are added either at a (possibly suboptimal) integer solution to the master problem or a *partially* feasible node, i. e. a node with integer assignments for a subset of machines.

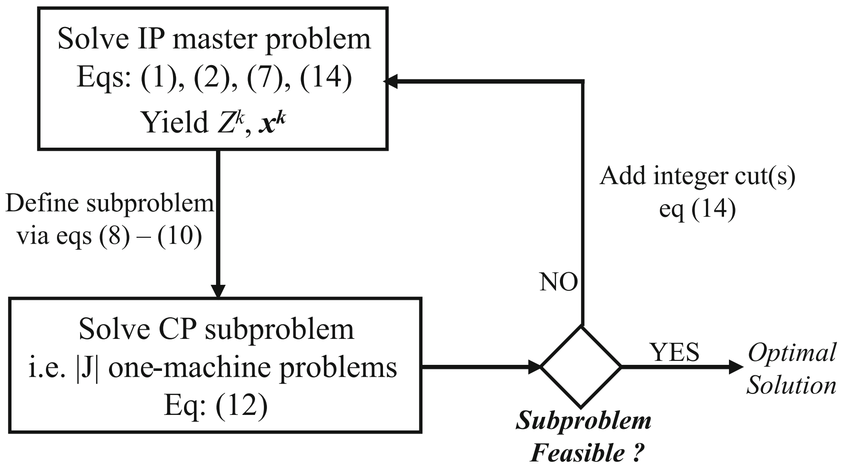
Bockmayr and Pisaruk [11] proposed a hybrid branch-and-cut scheme where the master problem is solved using an IP solver and the CP solver is called at a node of the tree, in order to generate integer cuts. The advantage of this method is that the IP model is not solved from scratch every time an integer solution (i. e. an assignment) is found. Furthermore, the authors were able to obtain cuts that are stronger than the ones proposed by Jain and Grossmann [28]. They were also able to generate cuts from fractional LP solutions of the IP model. The computational performance of the proposed hybrid branch-and-cut approach is better than the iterative IP/CP approach. Vazacopoulos and Verma [49] proposed certain Disjunctive and preemptive cuts to a priori forbid infeasible assignments and developed two hybrid MIP/CP algorithms for the MMASP. Sadykov and Wolsey [43] studied several hybrid approaches and developed two IP/CP hybrid schemes that appear to be better than those previously proposed. In the first, the authors were able to develop two classes of tightening inequalities, in the space of x_{ij} variables, which exclude many infeasible assignments and thus lead to smaller trees. The tightening inequalities are knapsack constraints, similar to constraint (7), but for subsets of set I. They also proposed a column generation algorithm using the tightening inequalities.

While the MMASP has been extensively studied due to its simple structure, hybrid schemes have also been developed for more complex scheduling problems. Harjunkoski and Grossmann [20], Timpe [46] and Constantino [14] presented hybrid schemes for complex chemical plants. Maravelias and Grossmann [33,34] proposed a general framework for integrating Mathematical and Constraint Programming methods for the solution of scheduling problems, while Maravelias [35] proposed the integration of MIP methods with heuristic algorithms. Hybrid methods that combine Mathematical and Constraint Programming have also been applied to transportation, inventory management and resource allocation problems.

Conclusions

While the computational efficiency of MIP/CP methods varies significantly, there is evidence that for some classes of problems they outperform existing methods. In general, if the structure of the problem at hand is exploited by efficient preprocessing and the generation of strong cuts, it is expected that hybrid schemes will be more effective because they combine the complementary strengths of two solution techniques.

The computational performance of hybrid methods relies on (i) the quality of the decomposition, (ii) the solution efficiency of the two subproblems, and (iii) the number of subproblems needed to be solved to prove optimality. Ideally, the original problem should be decomposed/reformulated into a tight MIP subproblem that is easily solved yielding potentially good feasible


Solve IP master problem
Eqs: (1), (2), (7), (14)
Yield Z^k, $\boldsymbol{x}^k$
Define subproblem via eqs (8) – (10)
Add integer cut(s) eq (14)
NO
Solve CP subproblem i.e. |J| one-machine problems Eq: (12)
YES
Optimal Solution
Subproblem Feasible ?

Mixed Integer Programming/Constraint Programming Hybrid Methods, Figure 1
Iterative hybrid IP/CP scheme of Jain and Grossmann [28]

solutions, and a feasibility CP subproblem that is used to check feasibility and generate cuts.

In particular, MIP/CP methods have been shown to be very effective in tackling scheduling problems where both assignment and sequencing decisions have to be made. The key idea in these methods is the decomposition of the original problem into two sub problems; Mathematical Programming is used for the assignment of tasks to resources, while Constraint Programming is used for the sequencing of tasks on resources.

References

1. Aggoun A, Beldiceanu N (1993) Extending CHIP in order to solve complex scheduling problems. J Math Comput Model 17(7):57–73
2. Aggoun A, Vazacopoulos A (2004) Solving sports scheduling and timetabling problems with constraint programming. In: Butenko S, Gil-Lafuente J, Pardalos PM (eds) Economics, Management and Optimization in Sports. Springer, Dordrecht, pp 243–264
3. Balas E (1965) An Additive Algorithm for Solving Linear Programs with 0–1 Variables. Oper Res 13:517–546
4. Balas E (1975) Facets of the Knapsack Polytope. Math Program 8:146–164
5. Balas E (2001) Integer Programming. In: Floudas T, Pardalos P (eds) Encyclopedia of Optimization, vol 2. Kluwer, pp 492–499
6. Baptiste P, Le Pape C (1996) Disjunctive Constraints for Manufacturing Scheduling: Principles and Extensions. Int J Comput Integr Manuf 9(4):306–3410
7. Baptiste P, Le Pape C (1996) Edge-Finding Constraint Propagation Algorithms for Disjunctive and Cumulative Scheduling. In: Proc 15th Workshop of the UK Planning Special Interest Group
8. Baptiste P, Le Pape C, Nuijten W (2001) Constrained-Based Scheduling: Applying Constraint Programming to Scheduling Problems. Kluwer
9. Benders JF (1962) Partitioning Procedures for Solving Mixed Variables Programming Problems. Numer Math 4:238–252
10. Bockmayr A, Kasper T (1998) Branch and Infer: A Unifying Framework for Integer and Finite Domain Constraint Programming. INFORMS J Comput 10(3):287–300
11. Bockmayr A, Pisaruk N (2003) Detecting Infeasibility and Generating Cuts for MIP Using CP. In: Proc CP-AI-OR 2003. Montreal, pp 24–34
12. Caseau Y, Laburthe F (1994) Improved CLP Scheduling with Task Intervals. In: Proc 11th Int Conf Log Program
13. Chvatal V (1973) Edmonds Polytops and a Hierarchy of Combinatorial Problems. Discret Math 4:305–337
14. Constantino M (2003) Integrated Lot-sizing and Scheduling of Barbot's Paint Production Using Combined MIP/CP. LISCOS Project Summary Meeting
15. Dakin RJ (1965) A Tree Search Algorithm for Mixed Integer Programming Problems. Comput J 8:250–255
16. Fisher ML (1981) The Lagrangean Relaxation Method for Solving Integer Programming Problems. Manag Sci 27: 1–18
17. Gomory RE (1958) Outline of an Algorithm for Integer Solutions to Linear Programs. Bull Am Math Soc 64:275–278
18. Gomory RE (1963) An Algorithm for Integer Solutions to Linear Programs. In: Graves R, Wolfe P (eds) Recent Advances in Mathematical Programming. McGraw Hill, New York, pp 269–302
19. Guignard M, Kim S (1987) Lagrangean Decomposition: A Model Yielding Stronger Lagrangean Bounds. Math Program 39:215–228

20. Harjunkoski I, Grossmann IE (2002) Decomposition Techniques for Multistage Scheduling Problems Using Mixed-Integer and Constrained Programming Methods. Comp Chem Eng 26:1533–1552
21. Heipcke S (1999) An Example of Integrating Constraint Programming and Mathematical Programming. Electron Note Discret Math 1(1):84
22. Hooker JN, Ottosson G, Thorsteinsson ES, Kim HJ (1999) On integrating constraint propagation and linear programming for combinatorial optimization. In: Proc 16th National Conf Artif Intell (AAAI-99), AAAI, The AAAI Press/MIT Press, Cambridge, pp 136–141
23. Hooker JN, Osorio MA (1999) Mixed Logic / Linear Programming. Discret Appl Math 97:395–442
24. Hooker JN (2000) Logic Based Methods for Optimization: Combining Optimization and Constraint Satisfaction. Wiley, New York
25. Hooker JN, Ottosson G, Thornsteinsson ES, Kim HJ (2000) A Scheme for Unifying Optimization and Constraint Satisfaction Methods. Knowl Eng Rev 15:11–30
26. Hooker JN (2002) Logic, Optimization, and Constraint Programming. INFORMS J Comput 14(4):295–321
27. Hooker JN (2006) Integrated Methods for Optimization. Springer, Dordrecht
28. Jain V, Grossmann IE (2001) Algorithms for Hybrid MIP/CP Model for a Class of Optimization Problems. INFORMS J Comput 13:258–276
29. Jaffar J, Maher M (1994) Constraint Logic Programming. Surv J Log Program 19(20):503–581
30. Johnson EL, Nemhauser GL, Savelsbergh MWP (2000) Progress in Linear Programming Based Branch-and-Bound Algorithms: Exposition. INFORMS J Comput 12:2–23
31. Land AH, Doig AG (1960) An automatic Method for Solving Discrete Programming Models. Econom 28:83–97
32. Le Pape C (1998) Implementation of Resource constraints in ILOG SCHEDULE: A Library for the Development of Constrained-Based Scheduling Systems. Intell Syst Eng 3(2):55–66
33. Maravelias CT, Grossmann IE (2004) A Hybrid MIP/CP Decomposition Approach for the Short Term Scheduling of Multipurpose Plants. Comput Chem Eng 28:1921–1949
34. Maravelias CT, Grossmann IE (2004) Using MILP and CP for the Scheduling of Batch Chemical Processes. Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Lect Note Comput Sci 3011:1–20
35. Maravelias CT (2006) A Decomposition Framework for the Scheduling of Single- and Multi-stage Processes. Comput Chem Eng 30(3):407–420
36. Marriott K, Stuckey PJ (1999) Introduction to Constraint Logic Programming. MIT Press, Cambridge, MA
37. Milano M (2003) Constraint and Integer Programming: Toward a Unified Methodology. Springer, Dordrecht
38. Nemhauser GL, Wolsey LA (1989) Integer and Combinatorial Optimization. Wiley, New York
39. Nuijten WPM (1994) Time and Resource Constrained Scheduling: A Constraint Satisfaction Approach. PhD Thesis, Eindhoven University of Technology
40. Padberg MW, van Roy TJ, Wolsey LA (1985) Valid Linear Inequalities for Fixed Charged Problems. Oper Res 33:842–861
41. Pinedo M (2001) Scheduling: Theory, Algorithms, and Systems. Prentice Hall, Englewood Cliffs
42. Rodosek R, Wallace MG, Hajian MT (1999) A New Approach to Integrating Mixed Integer Programming and Constraint Logic Programming. Ann Oper Res 86:63–87
43. Sadykov R, Wolsey L (2006) Integer and Constraint Programming in Solving a Multi-Machine Assignment Scheduling Problem with Deadlines and Release Dates. INFORMS J Comput 18(2):209–217
44. Simonis H (1995) Application Development with the CHIP System. In: Proc ESPRIT WG CONTESSA Workshop on Constraint Databases and Applications. Lect Note Comput Sci 1034:1–21
45. Smith SF, Cheng C-C (1993) Slack-based Heuristics for Constrained Satisfaction Scheduling. In: Proc 11th National Conf Artif Intell
46. Timpe C (2003) Solving BASF's Plastics Production and Lot-sizing Problem Using Combined CP/MIP. LISCOS Project Summary Meeting
47. van Hentenryck P (1989) Constraint Satisfaction in Logic Programming. MIT Press, Cambridge, MA
48. van Hentenryck P (2002) Constraint and Integer Programming in OPL. INFORMS J Comput 14(4):345–372
49. Vazacopoulos A, Verma N (2005) Hybrid MIP-CP techniques to solve the Multi-Machine Assignment and Scheduling Problem in Xpress-CP. In: Geunes J, Pardalos PM (eds) Supply Chain Optimization I. Springer, Dordrecht, pp 391–413
50. Wolsey LA (1975) Faces for a Linear Inequality in 0–1 Variables. Math Program 8:165–178
51. Wolsey LA (1990) Valid Inequalities for Mixed-Integer Programs with Generalized and Variable Upper Bound Constraints. Discret Appl Math 25:251–261
52. Wolsey L (1998) Integer Programming. Wiley, New York

Model Based Control for Drug Delivery Systems

PINKY DUA[1,2], VIVEK DUA[3], EFSTRATIOS N. PISTIKOPOULOS[4]

[1] Centre for Process Systems Engineering, Department of Chemical Engineering, Imperial College London, London, UK

[2] GlaxoSmithKline Research & Development Limited, Harlow, UK

[3] Centre for Process Systems Engineering, Department of Chemical Engineering, University College London, London, UK
[4] Centre for Process Systems Engineering, Department of Chemical Engineering, Imperial College London, London, UK

Article Outline

Abstract

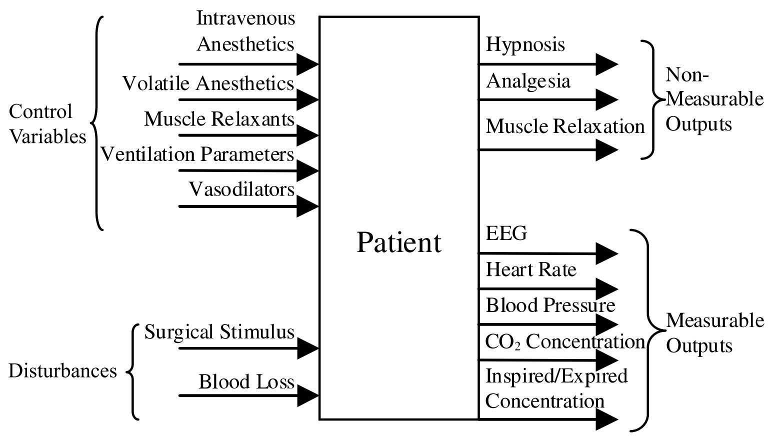
This chapter presents model based controllers for two drug delivery systems: (i) surgery under anesthesia and (ii) insulin delivery for type 1 diabetes. For anesthesia, a compartmental model is presented and then used for deriving model predictive controller for simultaneous control of mean arterial pressure (MAP), cardiac output (CO) and hypnosis. For type 1 diabetes, parametric control techniques are used for obtaining insulin delivery rate as an explicit function of the state of the patient. This reduces the implementation of the model based controller to function evaluations that can be carried out on a portable computational hardware.

Introduction

Drug delivery systems aim to provide effective therapy by minimizing side effects, reducing deviations from the desired state of the patient and increasing patient compliance and safety. Automation of a drug delivery system relies on the mathematical model of the patient that can take into account the pharmacokinetic and pharmacodynamic effects of the drugs on various organs of the body. To reduce the complexity of the mathematical model, some of the organs are lumped and then represented as interconnected compartments. This reduction in complexity is quite important especially for models that are used for controlling the amount of drugs to be infused. In this chapter, models and advanced model based controllers for two drug delivery systems are presented. In Sect. "Surgery Under Anesthesia", the first system which is concerned with the delivery of anesthetics for patients undergoing surgery is discussed. A compartmental model is presented that considers a choice of three drugs, isoflurane, dopamine and sodium nitroprusside, and therefore allows simultaneous control of mean arterial pressure, cardiac output and hypnosis. This model is then used for designing model predictive controller and the performance of the controller is tested for its set-point tracking capabilities. In Sect. "Blood Glucose Control for Type 1 Diabetes" model based parametric controller for the regulation of the blood glucose concentration for people with type 1 diabetes is derived. The key advantage of this controller is that the optimal drug infusion rate is obtained as an explicit function of the state of the patient and therefore requires simple function evaluations for its implementation. Concluding remarks are presented in Sect. "Concluding Remarks".

Surgery Under Anesthesia

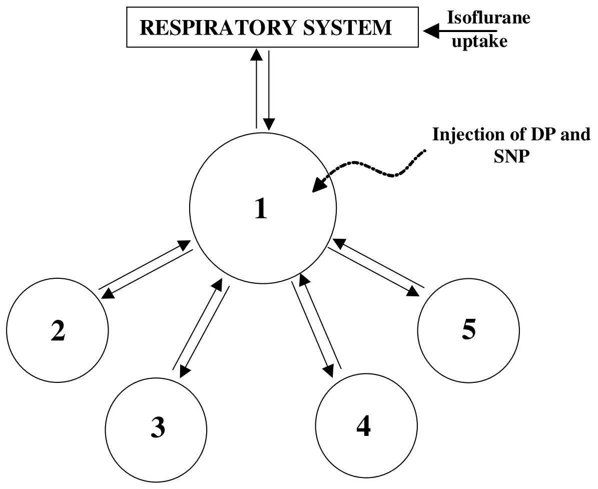
Anesthesia is defined as the absence or loss of sensation. In order to provide safe and adequate anesthesia, the anesthesiologist must guarantee analgesia, provide hypnosis, muscle relaxation and maintain vital functions of the patient. Anesthesiologists administer anesthetics and monitor a wide range of vital functions, such as mean arterial pressure (MAP), heart rate, cardiac output (CO). These vital functions need to be monitored and maintained within tolerable operating ranges by infusing various drugs and/or intravenous fluids as shown in Fig. 1. Automation of anesthesia is desirable as it will provide more time and flexibility to the anesthesiologist to focus on critical issues, monitor the conditions that cannot be easily measured and overall improve patient's safety. Also, the cost of the drugs will be reduced and shorter time will be spent in the postoperative care unit. There is a significant amount of research in the area of developing models and control strategies for anesthesia [10,14,15,17]. Gentilini et



Model Based Control for Drug Delivery Systems, Figure 1
Anesthesia control system (adapted from [6])

al. [6] proposed a model for the regulation of MAP and hypnosis with isoflurane. It was observed that controlling both MAP and hypnosis simultaneously with isoflurane was difficult. Yu et al. [16] proposed a model for regulating MAP and CO using dopamine (DP) and sodium nitroprusside (SNP), but the control of hypnosis was not considered.

In the next section, a compartmental model is presented, which allows the simultaneous regulation of the MAP and the unconsciousness of the patients. The model is characterized by: (i) pharmacokinetics for the uptake and distribution of the drugs, (ii) pharmacodynamics which describes the effect of the drugs on the vital functions and (iii) baroreflex for the reaction of the central nervous system to changes in the blood pressure. The model involves choice of three drugs, isoflurane, DP and SNP. This combination of drugs allows simultaneous regulation of MAP and hypnosis.



Model Based Control for Drug Delivery Systems, Figure 2
Compartmental model

Modeling Anesthesia

The model is based on the distribution of isoflurane in the human body [15]. It consists of five compartments organized as shown in Fig. 2.

The compartments 1–5 represent lungs, vessel rich organs (e. g. liver), muscles, other organs and tissues and fat tissues respectively.

The distribution of the drugs occurs from the central compartment to the peripheral compartments by the arteries and from the peripheral to the central by the veins. The first compartment in Fig. 2 is the central compartment and heart can be considered to be belonging to the central compartment, whereas compartments 2–5 are the peripheral compartments.

Pharmacokinetic Modeling The uptake of isoflurane in central compartment via the respiratory system is modeled as:

$$V\frac{\mathrm{d}C_{\text{insp}}}{\mathrm{d}T} = Q_{\text{in}}C_{\text{in}} - (Q_{\text{in}} - \Delta Q)C_{\text{insp}} - f_{\text{R}}(V_{\text{T}} - \Delta)(C_{\text{insp}} - C_{\text{out}}) ,$$

where C_{insp} is the concentration of isoflurane inspired by the patient (g/ml), C_{in} is the concentration of isoflurane in the inlet stream (g/ml), C_{out} is the concentration of isoflurane in the outlet stream (g/ml), Q_{in} is the inlet flow rate (ml/min), ΔQ is the losses (ml/min), V is the volume of the respiratory system (l), f_{R} is the respiratory frequency (l/min), V_{T} is the tidal volume (l) and Δ is the physiological dead space (ml). For the central compartment, the concentration of isoflurane is given by:

$$V_1 \frac{\mathrm{d}C_1}{\mathrm{d}t} = \sum_{i=2}^{5} \left(Q_i \left(\frac{C_i}{R_i} - C_1 \right) \right) + f_{\text{R}}(V_{\text{T}}\Delta)(C_{\text{insp}}C_1) ,$$

where C_i is the concentration of the drug in compartment i (g/ml), R_i is the partition coefficient between blood and tissues in compartment i, Q_i is the blood flow in compartment i (ml/min). The concentration of DP and SNP in the central compartment is modeled as follows:

$$V_1 \frac{\mathrm{d}C_1}{\mathrm{d}t} = \sum_{i=2}^{5} \left(Q_i \left(\frac{C_i}{R_i} - C_1 \right) \right) + C_{\text{inf}} - \frac{1}{\tau_{\frac{1}{2}}} C_1 V_1 ,$$

where C_{inf} is the concentration of the drug infused (g/min), V_i is the volume of compartment i (ml) and $\tau_{1/2}$ is the half-life of the drug (min). Isoflurane is eliminated by exhalation and metabolism in liver, the 2nd compartment, as follows:

$$V_2 \frac{\mathrm{d}C_2}{\mathrm{d}t} = Q_2 \left(C_1 - \frac{C_2}{R_2} \right) - k_{20} C_2 V_2 ,$$

where k_{20} is the rate of elimination of isoflurane in the 2nd compartment (min^{-1}). The distribution of isoflurane in compartments 3 to 5 is given by:

$$V_i \frac{\mathrm{d}C_i}{\mathrm{d}t} = Q_i \left(C_1 - \frac{C_i}{R_i} \right) , \quad i = 3, \ldots, 5 .$$

The natural decay of DP and SNP in the body, for compartment 2 to 5, is given by:

$$V_i \frac{\mathrm{d}C_i}{\mathrm{d}t} = Q_i \left(C_1 - \frac{C_i}{R_i} \right) - \frac{1}{\tau_{\frac{1}{2}}} C_i V_i , \quad i = 2, \ldots, 5 .$$

Pharmacodynamic Modeling The effect of DP and SNP on two of the heart's characteristic parameters: maximum elastance ($E_{\max}$) and systemic resistance (R_{sys}) is given by:

$$\begin{aligned} \frac{\mathrm{dEff}}{\mathrm{d}t} &= k_1 C_1^N (\mathrm{Eff}_{\max} - \mathrm{Eff}) - k_2 \mathrm{Eff} \\ E_{\max} &= E_{\max,0} \left(1 + \mathrm{Eff}_{\mathrm{DP}-E_{\max}} \right) \\ R_{\text{sys}} &= R_{\text{sys},0} \left(1 - \mathrm{Eff}_{\mathrm{DP}-R_{\text{sys}}} - \mathrm{Eff}_{\mathrm{SNP}-R_{\text{sys}}} \right) , \end{aligned}$$

where Eff is the measure of the effect of drug on the parameters of interest, R_{sys} is the systemic resistance (mmHg/(ml/min)), $E_{\max}$ is the maximum elastance (mmHg/ml), $E_{\max,0}$ is nominal maximum elastance, $R_{\text{sys},0}$ is nominal systemic resistance, $\mathrm{Eff}_{\mathrm{DP}-E\max}$ is effect of DP on $E_{\max}$, $\mathrm{Eff}_{\mathrm{DP}-R\mathrm{sys}}$ is effect of DP on R_{sys}, $\mathrm{Eff}_{\mathrm{SNP}-R\mathrm{sys}}$ is the effect of SNP on R_{sys}, k_1, k_2 are the rate constants and N is the non-linearity constant. MAP can then be expressed as a function of $E_{\max}$ and R_{sys} as:

$$\begin{aligned} &\mathrm{MAP}^2 \frac{1}{R_{\text{sys}}^2} + 2K^2 \mathrm{MAP} - 2K^2 V_{\text{LV}} E_{\max} = 0 \\ &K = \frac{A_{\text{aorta}} A_{\text{LV}}}{\sqrt{\rho} \sqrt{A_{\text{LV}}^2 - A_{\text{aorta}}^2}} , \end{aligned}$$

where MAP is the mean arterial pressure (mmHg), A_{aorta} is the cross sectional area of the aorta (cm^2), A_{LV} is the cross sectional area of the left ventricle (cm^2), V_{LV} is the mean volume of the left ventricle (ml) and ρ is the blood density (g/ml). Isoflurane affects MAP as follows:

$$MAP = \frac{Q_1}{\sum_{i=2}^{5} \left(g_{i,0} \left(1 + b_i C_i \right) \right)} ,$$

where, $g_{i,0}$ is the baseline conductivities (ml/(min.mmHg)) and b_i is the variation coefficient of conductivity (ml/g). There is experimental evidence that a transportation delay exists between the lungs and the site of effect of isoflurane on the unconsciousness of the patient. In order to model this, an effect compartment is linked to the central compartment. The concentration of isoflurane within this compartment is related to the central compartment, which is given by:

$$\frac{\mathrm{d}C_{\text{e}}}{\mathrm{d}t} = k_{\text{e}0}(C_1 - C_{\text{e}}) ,$$

where C_{e} is the concentration of isoflurane in the effect compartment (g/ml), and $k_{\text{e}0}$ is the kinetics in the effect compartment (min^{-1}). The action of isoflurane can be then expressed as follows:

$$\begin{aligned} \Delta\text{BIS} &= \Delta\text{BIS}_{\text{MAX}} \frac{C_e^\gamma}{C_e^\gamma + \text{EC}_{50}^\gamma} \\ \Delta\text{BIS} &= \text{BIS} - \text{BIS}_0 \\ \Delta\text{BIS}_{\text{MAX}} &= \text{BIS}_{\text{MAX}} - \text{BIS}_0 \,, \end{aligned}$$

where BIS_0 is the baseline value of BIS (assumed to be 100), BIS_{MAX} is the maximum value of BIS (assumed to be 0), EC_{50} is the patient's sensitivity to the drug and γ is the measure of the degree of non-linearity.

Baroreflex Baroreflex is obtained from a set of transfer functions relating the mean arterial pressure to the maximum elastance and the systemic resistance and is given by:

$$\text{bfc} = \frac{e^{c(\text{MAP}-\text{MAP0})}}{1 + e^{c(\text{MAP}-\text{MAP0})}} \,,$$

where c is the empirical constant (mmHg).

Control of Anesthesia

The model presented in the previous section was validated by carrying out a number of dynamic simulations for different amounts of drug dosages and disturbances using gPROMS [7]. For designing controllers, this model was linearized at the nominal values of inputs: 0.6% vol. of isoflurane, 2 μg/kg/min of DP and 4 μg/kg/min of SNP and outputs: 57.38 mmHg of MAP, 61.1 BIS and 1.21 l/min of CO, to obtain a state-space model of the following form:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t + Du_t \,, \end{aligned} \tag{1}$$

subject to the following constraints:

$$\begin{aligned} x_{\min} &\le x_t \le x_{\max} \\ y_{\min} &\le y_t \le y_{\max} \\ u_{\min} &\le u_t \le u_{\max} \,, \end{aligned} \tag{2}$$

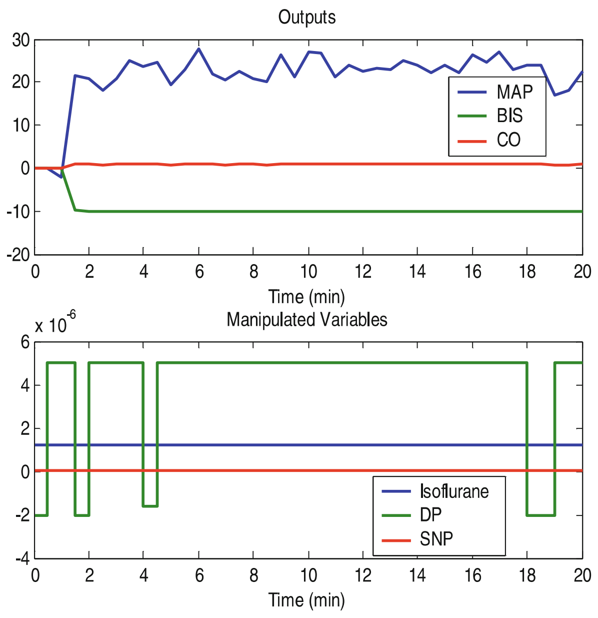
where $x_t \in R^n, y_t \in R^l, u_t \in R^m$, are the state, output and input vectors respectively and the subscripts min and max denote lower and upper bounds respectively. Model predictive control (MPC) [5] problem can then be posed as the following optimization problem:

$$\begin{aligned} \min_U \; & J(U, x(t)) = x_{t+N_y|t}^T P x_{t+N_y|t} \\ & + \sum_{k=0}^{N_y-1} \left[x_{t+k|t}^T Q x_{t+k|t} + u_{t+k}^T R u_{t+k} \right] \\ s.t. \; & x_{\min} \le x_{t+k|t} \le x_{\max} \,, \quad k = 1, \ldots, N_c \\ & y_{\min} \le y_{t+k|t} \le y_{\max} \,, \quad k = 1, \ldots, N_c \\ & u_{\min} \le u_{t+k} \le u_{\max} \,, \quad k = 1, \ldots, N_c \\ & x_{t+k+1|t} = Ax_{t+k|t} + Bu_{t+k} \,, \quad k \ge 0 \\ & y_{t+k|t} = Cx_{t+k} + Du_{t+k} \,, \quad k \ge 0 \\ & u_{t+k} = Kx_{t+k|t} \,, \quad N_u \le k \le N_y \,, \end{aligned} \tag{3}$$

where $U = [u_t^T, \ldots, u_{t+N_u-1}^T]^T$, Q and R are constant, symmetric and positive definite matrices, P is given by the solution of the Riccati or Lyapunov equation, N_y, N_u and N_c are the prediction, control and constraint horizons respectively and the superscript T denotes transpose of the vector. Problem (3) is solved at each time t for the current state x_t and the vector of predicted state variables, $x_{t+1|t}, \ldots, x_{t+Ny|t}$ at time $t+1, \ldots, t+k$ respectively and corresponding control actions $u_t, \ldots, u_{t+k}$ are obtained.

Results

The model for anesthesia consists of 23 states, 3 outputs and 3 inputs. This state-space form of the model is then adapted for designing model predictive controller by using the MATLAB Model Predictive Control Toolbox™ [11]. For designing the MPC controller, the following input: $0 \le \text{DP} \le 7$ μg/kg.min, $0 \le \text{SNP} \le 10$ μg/kg.min, $0 \le \text{Isoflurane} \le 5$%vol., and output constraints: $40 \le \text{MAP} \le 150$mmHg, $40 \le \text{BIS} \le 65$, $1 \le \text{CO} \le 6.5$ l/min are used. A prediction horizon of 5, control horizon of 3 and sampling time of 0.5 minutes are considered. A set point of $[20\text{–}10\ 1]'$ deviation from the nominal point of the output variables is given and the performance of the controller is shown in Fig. 3. It is observed that the MPC tracks the set point quite well. The performance of the MPC was also tested by reducing the model to 15 states and was observed to be very good. From the above results it can be inferred that the model based control technology provides a promising platform for the automation of anesthesia.

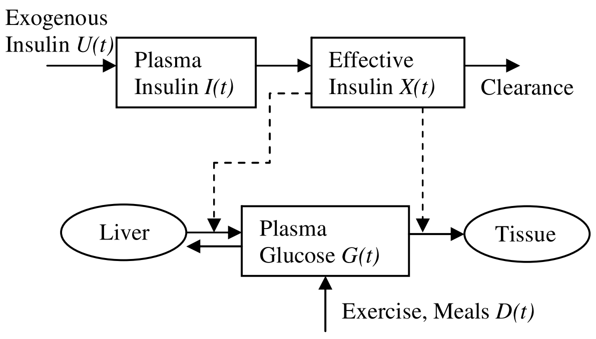


Model Based Control for Drug Delivery Systems, Figure 3
MPC performance for anesthesia

Note that MPC solves a quadratic program at regular time intervals. In the next section a parametric programming approach for control of blood glucose for type 1 diabetes is presented that does not require repetitively solving quadratic programs.

Blood Glucose Control for Type 1 Diabetes

Diabetes is a disease that affects the body's ability to regulate glucose. In Type 1 diabetes, the pancreas produces insufficient insulin, and exogenous insulin is required to be infused at an appropriate rate to maintain blood sugar levels within the range of 60–120 mg/dl [2]. If insulin is supplied in excess, the blood glucose level can go well below normal (< 60 mg/dl), a condition known as *hypoglycemia*. On the other hand, if insulin is not supplied sufficiently, the blood glucose level is elevated above normal (> 120 mg/dl), a condition known as *hyperglycemia*. Both hypo- and hyperglycemia can be harmful to an individual's health. Hence, it is very important to control the level of blood glucose in the body to within a reasonable range [9,12]. In the following sections, advanced model based controllers for regulating the blood glucose concentration for type 1 diabetes are presented.



Model Based Control for Drug Delivery Systems, Figure 4
Schematic representation of the Bergman model

Model for Type 1 Diabetes

The Bergman model [1] is used in this study, which presents a 'minimal' model comprising 3 equations to describe the dynamics of the system. The schematic representation of the model is shown in Fig. 4. The modeling equations are:

$$\frac{\mathrm{d}G}{\mathrm{d}t} = -P_1 G - X(G + G_b) + D(t) \qquad (4)$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = -n(I + I_b) + U(t)/V_1 \qquad (5)$$

$$\frac{\mathrm{d}X}{\mathrm{d}t} = -P_2 X + P_3 I\,. \qquad (6)$$

The states in this model are: G, plasma glucose concentration (mg/dl) relative to basal value, I, plasma insulin concentration (mU/l) relative to basal value, and X, proportional to I in remote compartment (min^{-1}). The inputs are: $D(t)$, meal glucose disturbance (mg/dl/min), $U(t)$, manipulated insulin infusion rate (mU/min) and G_b, I_b, nominal values of glucose and insulin concentration (81 mg/dl; 15 mU/l). The parameter values for a Type 1 diabetes are: $P_1 = 0\,\text{min}^{-1}$, $P_2 = 0.025\,\text{min}^{-1}$, $P_3 = 0.000013\,\text{l/mUmin}^2$, $V_1 = 12\,\text{l}$ and $n = 5/54\,\text{min}$ [4].

The model, (4)–(6) is linearized about the steady-state values of $G_b = 81$ mg/dl, $I_b = 15$ mU/l, $X_b = 0$ and $U_b = 16.66667$ mU/min to obtain the state space model of the form: $x_{t+1} = Ax_t + Bu_t + B_d d_t$ where the term d_t represents the input disturbance glucose meal. The sampling time considered is 5 minutes, which is reasonable for the current glucose sensor technology. The discrete state-space matrices A, B, C and B_d

are as follows:

$$A = \begin{bmatrix} 1 & -0.000604 & -21.1506 \\ 0 & 0.6294 & 0 \\ 0 & 0.00004875 & 0.8825 \end{bmatrix}$$
$$B = \begin{bmatrix} -0.000088 \\ 0.3335 \\ 0.0000112 \end{bmatrix},$$
$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \quad B_d = \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix}$$

The constraints imposed are $60 \leq G + G_b \leq 180$ and $0 \leq U + U_b \leq 100$.

Parametric Controller

Parametric programming can be used in the MPC framework to obtain U as a function of x_t by treating U as optimization variables and x_t as parameters as described next [3,13]. For simplicity in presentation assume that $N_y = N_u = N_c$, the theory presented is however valid for the case when N_y, N_u and N_c are not equal. The equalities in formulation (3) are eliminated by making the following substitution:

$$x_{t+k|t} = A^k x_t + \sum_{j=0}^{k-1} A^j B u_{t+k-1-j} \tag{7}$$

to obtain the following Quadratic Program (QP):

$$\begin{aligned} &\min_U \frac{1}{2} U^T H U + x_t^T F U + \frac{1}{2} x_t^T Y x_t \\ &s.t. GU \leq W + E x_t , \end{aligned} \tag{8}$$

where, $U = [u_t^T, \ldots, u_{t+N_u-1}^T]^T \in R^s$, is the vector of optimization variables, $s = mN_u$, H is a constant, symmetric and positive definite matrix and H, F, Y, G, W, E are obtained from Q, R and (1) and (2).

The QP problem in (8) can now be reformulated as a multi-parametric quadratic program (mp-QP):

$$\begin{aligned} &V_z(x) = \min_z \frac{1}{2} z^T H z \\ &s.t. Gz \leq W + S x_t , \end{aligned} \tag{9}$$

where, $z = U + H^{-1} F^T x_t, z \in R^s$, and $S = E + GH^{-1}F^T$.

This mp-QP is solved by treating z as the vector of optimization variables and x_t as the vector of parameters to obtain z as an explicit function of x_t. U is then obtained as an explicit function of x_t by using $U = z - H^{-1} F^T x_t$.

Results

A prediction horizon $N_y = 5$ and Q/R ratio of 1000 is considered for deriving the control law – this results in partitioning of the state-space into 31 polyhedral regions. These regions are known as Critical Regions (CR). Associated with each CR is a control law that is an affine function of the state of the patient. For example, one of the CRs is given by the following state inequalities:

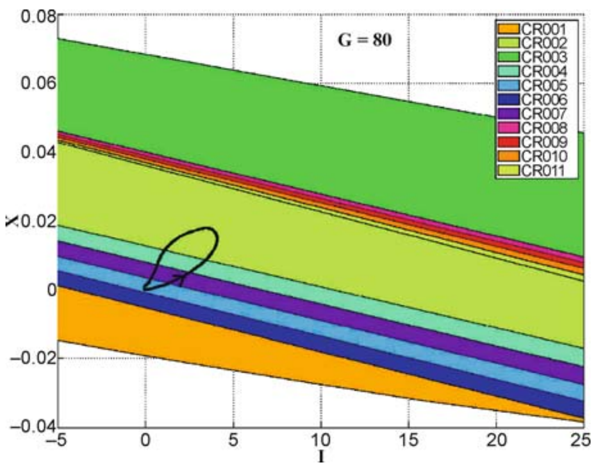
$$\begin{aligned} &-5 \leq I \leq 25 \\ &0.0478972G - 0.0002712I - X \leq 0.104055 \\ &0.0261386G - 0.0004641I - X \leq 0.0576751 \\ &-0.00808846G + 0.00119685I + X \leq 0 \\ &-0.00660123G + 0.00130239I + X \leq 0 \\ &0.00609435G - 0.00134362I - X \leq 0 \end{aligned} \tag{10}$$

where the insulin infusion rate as a function of the state variables for the next five time intervals is given as follows:

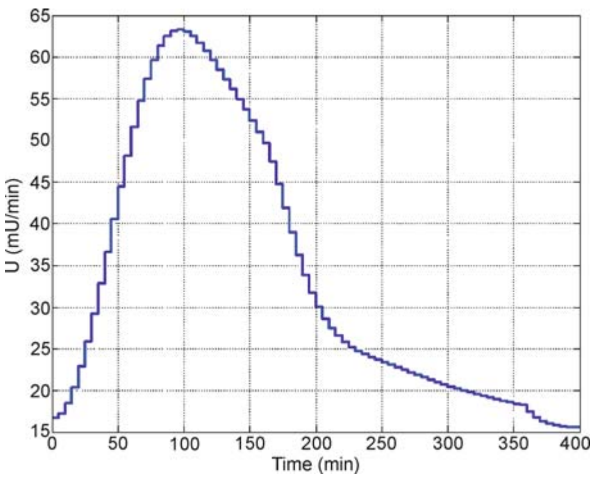
$$\begin{aligned} U(1) &= 30.139G - 0.44597I - 3726.2X \\ U(2) &= 24.874G - 0.40326I - 3280.4X \\ U(3) &= 20.16G - 0.35946I - 2842.8X \\ U(4) &= 16.002G - 0.31571I - 2424.1X \\ U(5) &= 0 \end{aligned} \tag{11}$$

The complete partitioning of the state-space for G = 80 mg/dl into CRs is shown in Fig. 5. The performance of the parametric controller for a 50 mg meal disturbance [8] is as shown in Figs. 6 and 7. The corresponding trajectory of the state variables is also shown in Fig. 5.

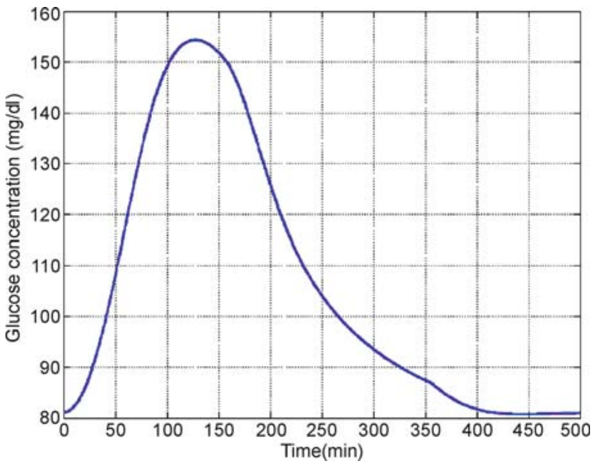
The model based parametric controller of the form given in (10) and (11) can be stored and implemented on a simple computational hardware and therefore can provide effective therapy at low on-line computational costs.



Model Based Control for Drug Delivery Systems, Figure 5
Critical regions for type 1 diabetes



Model Based Control for Drug Delivery Systems, Figure 7
Insulin infusion vs. time



Model Based Control for Drug Delivery Systems, Figure 6
Glucose concentration vs. time

Concluding Remarks

Automation of drug delivery systems aims at reducing patient inconvenience by providing better and personalized healthcare. The automation can be achieved by developing detailed models and by deriving advanced controllers that can take into account the model as well as the constraints on state and control variables. In this chapter, a compartmental model incorporating pharmacokinetic and pharmacodynamic aspects for delivery of anesthetic agents has been presented. This model was then used for the derivation of model predictive controller. For type 1 diabetes, implementation of advanced model based controllers through a simple computational hardware was demonstrated by deriving insulin delivery rate as an explicit function of the state of patient. The developments presented in this chapter highlight the importance of modeling and control techniques for biomedical systems.

See also

► Nondifferentiable Optimization: Parametric Programming

References

1. Bergman RN, Phillips LS, Cobelli C (1981) Physiologic evaluation of factors controlling glucose tolerance in man. J Clin Invest 68:1456–1467
2. DCCT-The Diabetes Control and Complications Trial Research Group (1993) The effect of intensive treatment of diabetes on the development and progression of long-term complications in insulin-dependent diabetes mellitus. New Engl J Med 329:977–986
3. Dua P, Doyle FJ III, Pistikopoulos EN (2006) Model based glucose control for type 1 diabetes via parametric programming. IEEE Trans Biomed Eng 53:1478–1491
4. Fisher ME (1991) A semiclosed loop algorithm for the control of blood glucose levels in diabetics. IEEE Trans Biomed Eng 38:57–61
5. Garcia CE, Prett DM, Morari M (1989) Model predictive control: theory and practice – a survey. Automatica 25: 335–348
6. Gentilini A, Frei CW, Glattfelder AH, Morari M, Sieber TJ, Wymann R, Schnider TW, Zbinden AM (2001) Multitasked closed-loop control in anesthesia. IEEE Eng Med Biol 20:39–53

7. gPROMS (2003) Introductory user guide, Release 2.2. Process Systems Enterprise Ltd, London
8. Lehmann ED, Deutsch T (1992) A physiological model of glucose-insulin interaction in type 1 diabetes mellitus. J Biomed Eng 14:235–242
9. Lynch SM, Bequette BW (2002) Model predictive control of blood glucose in type I diabetics using subcutaneous glucose measurements. In: Proc. American Control Conf., Anchorage, AK, pp 4039–4043
10. Mahfouf M, Asbury AJ, Linkens DA (2003) Unconstrained and constrained generalized predictive control of depth of anaesthesia during surgery. Control Eng Pract 11: 1501–1515
11. MATLAB (1998) MPC Toolbox manual. The MathWorks, Natick
12. Parker RS, Doyle FJ III, Peppas NA (2001) The intravenous route to blood glucose control. IEEE Eng Med Biol 20: 65–73
13. Pistikopoulos EN, Dua V, Bozinis NA, Bemporad A, Morari M (2002) On line optimization via off-line parametric optimization tools. Comput Chem Eng 26:175–185
14. Rao RR, Palerm CC, Aufderhiede B, Bequette BW (2001) Automated regulation of hemodynamic variables. IEEE Eng Med Biol 20:24–38
15. Yasuda N, Lockhart SH, Eger EI, Weiskopf RB, Laster M, Taheri S, Peterson NA (1991) Comparison of kinetics of sevoflurane and isoflurane in humans. Anesthesia Analg 72:316–324
16. Yu C, Roy RJ, Kaufman H (1990) A circulatory model for combined nitroprusside-dopamine therapy in acute heart failure. Med Prog Technol 16:77–88
17. Zwart AN, Smith NT, Beneken JEW (1972) Multiple model approach to uptake and distribution of halothane: the use of an analog computer. Comput Biomed Res 5:228–238

Modeling Difficult Optimization Problems

JOSEF KALLRATH[1,2]

[1] GVC/S (Scientific Computing) - B009, BASF Aktiengesellschaft, Ludwigshafen, Germany

[2] Astronomy Department, University of Florida, Gainesville, USA

MSC2000: 90C06, 90C10, 90C11, 90C30, 90C57, 90C90

Article Outline

Introduction

We define difficult optimization problems as problems that cannot be solved to optimality or to any guaranteed bound by any standard solver within a reasonable time limit. The problem class we have in mind are mixed-integer programming (MIP) problems. Optimization, and especially MIP, is often appropriate and frequently used to model real-world optimization problems. While it started in the 1950s, models have become larger and more complicated.

A reasonable general framework is mixed-integer nonlinear programming (MINLP) problems. They are specified by the augmented vector $\mathbf{x}_\oplus^\mathrm{T} = \mathbf{x}^\mathrm{T} \oplus \mathbf{y}^\mathrm{T}$ established by the vectors $\mathbf{x}^\mathrm{T} = (x_1, \ldots, x_{n_c})$ and $\mathbf{y}^\mathrm{T} = (y_1, \ldots, y_{n_d})$ of n_c continuous and n_d discrete variables, an objective function $f(\mathbf{x}, \mathbf{y})$, n_e equality constraints $\mathbf{h}(\mathbf{x}, \mathbf{y})$, and n_i inequality constraints $\mathbf{g}(\mathbf{x}, \mathbf{y})$. The problem

$$\min \left\{ f(\mathbf{x},\mathbf{y}) \;\middle|\; \begin{array}{l} \mathbf{h}(\mathbf{x},\mathbf{y}) = 0,\ \mathbf{h}: X \times U \to \mathbb{R}^{n_e}, \\ \mathbf{x} \in X \subseteq \mathbb{R}^{n_c} \\ \mathbf{g}(\mathbf{x},\mathbf{y}) \ge 0,\ \mathbf{g}: X \times U \to \mathbb{R}^{n_i}, \\ \mathbf{y} \in U \subseteq \mathbb{Z}^{n_d} \end{array} \right\} \tag{1}$$

is called a *mixed-integer nonlinear programming* (MINLP) problem if at least one of the functions $f(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$, or $\mathbf{h}(\mathbf{x}, \mathbf{y})$ is nonlinear. The vector inequality, $\mathbf{g}(\mathbf{x}, \mathbf{y}) \ge 0$, is to be read componentwise. Any vector $\mathbf{x}_\oplus^\mathrm{T}$ satisfying the constraints of (1) is called a *feasible point* of (1). Any feasible point whose objective function value is less than or equal to that of all other feasible points is called an *optimal solution*. From this definition it follows that the problem might not have a unique optimal solution.

Depending on the functions $f(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$, and $\mathbf{h}(\mathbf{x}, \mathbf{y})$ in (1) we get the following structured problems known as

Acronym	*Type of optimization*	$f(\mathbf{x}, \mathbf{y})$	$\mathbf{h}(\mathbf{x}, \mathbf{y})$	$\mathbf{g}(\mathbf{x}, \mathbf{y})$	n_d
LP	Linear programming	$\mathbf{c}^T\mathbf{x}$	$A\mathbf{x} - \mathbf{b}$	$\mathbf{x}$	0
QP	Quadratic programming	$\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x}$	$A\mathbf{x} - \mathbf{b}$	$\mathbf{x}$	0
NLP	Nonlinear programming				0
MILP	Mixed-integer LP	$\mathbf{c}^T\mathbf{x}_\oplus$	$A\mathbf{x}_\oplus - \mathbf{b}$	$\mathbf{x}_\oplus$	≥ 1
MIQP	Mixed-integer QP	$\mathbf{x}_\oplus^T Q\mathbf{x}_\oplus + \mathbf{c}^T\mathbf{x}_\oplus$	$A\mathbf{x}_\oplus - \mathbf{b}$	$\mathbf{x}_\oplus$	≥ 1
MINLP	Mixed-integer NLP				≥ 1

with a matrix A of m rows and n columns, i. e., $A \in \mathcal{M}(m \times n, \mathbb{R}), \mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^n$, and $n = n_c + n_d$. Real-world problems lead much more frequently to LP and MILP than to NLP or MINLP problems. QP refers to quadratic programming problems. They have a quadratic objective function but only linear constraints. QP and MIQP problems often occur in applications of the financial services industry.

While LP problems as described in [31] or [1] can be solved relatively easily (the number of iterations, and thus the effort to solve LP problems with m constraints, grows approximately linearly in m), the computational complexity of MILP and MINLP grows exponentially with n_d but depends strongly on the structure of the problem. Numerical methods to solve NLP problems work iteratively, and the computational problems are related to questions of convergence, getting stuck in bad local optima and availability of good initial solutions. Global optimization techniques can be applied to both NLP and MINLP problems, and its complexity increases exponentially in the number of all variables entering nonlinearly into the model.

While the word *optimization*, in nontechnical or colloquial language, is often used in the sense of *improving*, the mathematical optimization community sticks to the original meaning of the word related to finding the *best value* either globally or at least in a local neighborhood. For an algorithm being considered as a (mathematical, strict, or exact) optimization algorithm in the mathematical optimization community there is consensus that such an algorithm computes feasible points proven globally (or locally) optimal for linear (nonlinear) optimization problems. Note that this is a definition of a mathematical optimization algorithm and *not* a statement saying that computing a local optimum is sufficient for nonlinear optimization problems. In the context of mixed-integer linear problems an optimization algorithm [12] and [13] is expected to compute a proven optimal solution or to generate feasible points and, for a maximization problem, to derive a reasonably tight, nontrivial upper bound. The quality of such bounds is quantified by the integrality gap – the difference between the upper and lower bound. What one considers to be a good-quality solution depends on the problem, the purpose of the model, and the accuracy of the data. A few percent, say 2 to 3%, might be acceptable for the example discussed by Kallrath (2007, Encyclopedia: Planning). However, discussion based on percentage gaps become complicated when the objective function includes penalty terms containing coefficients without a strict economic interpretation. In such cases scaling is problematic. Goal programming as discussed in ([23], p. 294) might help in such situations to avoid penalty terms in the model. The problem is first solved with respect to the highest-priority goal, then one is concerned with the next level goal, and so on.

For practical purposes it is also relevant to observe that solving mixed-integer linear problems and the problem of finding appropriate bounds is often $\mathcal{NP}$-complete, which makes these problems hard to solve. A consequence of this structural property is that these problems scale badly. If the problem can be solved to optimality for a given instance, this might not be so if the size is increased slightly. While tailor-made optimization algorithms such as column generation and branch-and-price techniques can often cope with this situation for individual problems, it is very difficult for standard software.

We define difficult optimization problems as problems that cannot be solved to optimality or within a reasonable integrality gap by any standard MIP solver within a reasonable time limit. Problem structure, size, or both could lead to such behavior. However, in many cases these problems (typically MIP or nonconvex optimization problems fall into this class) can be solved if they are individually treated, and we resort to the art of modeling.

The art of modeling includes choosing the right level of detail implemented in the model. On the one hand, this needs to satisfy the expectations of the owner of the real-world problem. On the other hand, we are limited by the available computational resources. We give reasons why strict optimality or at least safe bounds are essential when dealing with real-world problems and why we do not accept methods that do not generate both upper and lower bounds.

Mapping the reality also forces us to discuss whether deterministic optimization is sufficient or whether we need to resort to optimization under uncertainty. Another issue is to check whether one objective function suffices or whether multiple-criterion optimization techniques need to be applied.

Instead of solving such difficult problems directly as, for example, a standalone MILP problem, we discuss how problems can be solved equivalently by solving a sequence of models.

Efficient approaches are as follows:

- Column generation with a master and subproblem structure,
- Branch-and-price,
- Exploiting a decomposition structure with a rolling time horizon,
- Exploiting auxiliary problems to generate safe bounds for the original problem, which then makes the original problems more tractable,
- Exhaustion approaches,
- Hybrid methods, i. e., constructive heuristics and local search on subsets of the difficult discrete variables leaving the remaining variables and constraints in tractable MILP or MINLP problems that can be solved.

We illustrate various ideas using real-world planning, scheduling, and cutting-stock problems.

Models and the Art of Modeling

We are here concerned with two aspects of modeling and models. The first one is to obtain a reasonable representation of the reality and mapping it onto a mathematical model, i. e., an optimization problem in the form of (1). The second one is to reformulate the model or problem in such equivalent forms that is is numerically tractable.

Models The terms *modeling* and *model building* are derived from the word *model*. Its etymological roots are the Latin word *modellus* (scale, [diminutive of modus, measure]) and what was to be in the 16th century the new word *modello*. Nowadays, in a scientific context the term is used to refer to a simplified, abstract, or well-structured part of the reality one is interested in. The idea itself and the associated concept is, however, much older. Classical geometry, and especially Pythagoras around 600 B.C., distinguish between wheel and circle and field and rectangle. Around A.D. 1100 a wooden model of the later Speyer cathedral was produced; the model served to build the real cathedral. Astrolabs and celestial globes have been used as models to visualize the movement of the moon, planets, and stars on the celestial sphere and to compute the times of rises and settings. Until the 19th century mechanical models were understood as pictures of reality. Following the principles of classical mechanics the key idea was to reduce all phenomena to the movement of small particles. Nowadays, in physics and other mathematical sciences one will talk about models if

- For reasons of simplification, one restricts oneself to certain aspects of the problem (*example*: if we consider the movement of the planets, in a first approximation the planets are treated as point masses);
- For reasons of didactic presentation, one develops a simplified picture for more complicated reality (*example*: the planetary model is used to explain the situation inside atoms);
- One uses the properties in one area to study the situation in an analogous problem.

A model is referred to as a mathematical model of a process or a problem if it contains typical mathematical objects (variables, terms, relations). Thus, a (mathematical) model represents a real-world problem in the language of mathematics using mathematical symbols, variables, equations, inequalities, and other relations.

It is very important when building a model to define and state precisely the purpose of the model. In science, we often encounter epistemological arguments. In engineering, a model might be used to construct some machines. In operations research and optimization, models are often used to support strategic or operative decisions. All models enable us to

- Learn and understand situations that do not allow easy access (very slow or fast processes, processes involving a very small or very large region);
- Avoid difficult, expensive, or dangerous experiments; and
- Analyze case studies and *what-if-when scenarios*.

Tailored optimization models can be used to support decisions (that is, the overall purpose of the model). It is essential to have a clear objective describing what a good decision is. The optimization model should produce, for instance, optimal solutions in the following sense:

- To avoid unwanted byproducts as much as possible,
- To minimize costs, or
- to maximize profit, earnings before interest and taxes (EBIT), or contribution margin.

The purpose of a model may change over time.

To solve a real-world problem by mathematical optimization, at first we need to represent our problem by a *mathematical model*, that is, a set of mathematical relationships (e. g., equalities, inequalities, logical conditions) representing an abstraction of our real-world problem. This translation is part of the model-building phase (which is part of the whole modeling process) and is not trivial at all because there is nothing we could consider an exact model. Each model is an acceptable candidate as long as it fulfills its purpose and approximates the real world accurately enough. Usually, a model in mathematical optimization consists of four key objects:

- Data, also called the *constants* of a model;
- Variables (continuous, semicontinuous, binary, integer), also called decision variables;
- Constraints (equalities, inequalities), also called *restrictions*; and
- Objective function (sometimes even several of them).

The data may represent costs or demands, fixed operation conditions of a reactor, capacities of plants, and so on. The variables represent the degrees of freedom, i. e., what we want to decide: how much of a certain product is to be produced, whether a depot is closed or not, or how much material we will store in the inventory for later use. Classical optimization (calculus, variational calculus, optimal control) treats those cases in which the variables represent continuous degrees of freedom, e. g., the temperature in a chemical reactor or the amount of a product to be produced. Mixed-integer optimization involves variables restricted to integer values, for example counts (numbers of containers, ships), decisions (yes-no), or logical relations (if product A is produced, then product B also needs to be produced). The constraints can be a wide range of mathematical relationships: algebraic, analytic, differential, or integral. They may represent mass balances, quality relations, capacity limits, and so on. The objective function expresses our goal: minimize costs, maximize utilization rate, minimize waste, and so on. Mathematical models for optimization usually lead to structured problems such as:

- *Linear programming* (LP) problems,
- *Mixed-integer linear programming* (MILP) problems,
- *Quadratic* (QP) and *mixed-integer quadratic programming* (MIQP),
- *Nonlinear programming* (NLP) problems, and
- *Mixed-integer nonlinear programming* (MINLP) problems.

The Art of Modeling How do we get from a given problem to its mathematical representation? This is a difficult, nonunique process. It is a compromise between the degree of detail required to model a problem and the complexity, which is tractable. However, simplifications should not only be seen as an unavoidable evil. They could be useful for developing understanding or serve as a platform with the client, as the following three examples show.

1. At the beginning of the modeling process it can be useful to start with a "down-scaled" version to develop a feeling for the structure and dependencies of the model. This enable a constructive dialog between the modeler and the client. A vehicle fleet with 100 vehicles and 12 depots could be analyzed with

only 10 vehicles and 2 depots to let the *model world* and the *real world* find each other in a sequence of discussions.

2. In partial or submodels the modeler can develop a deep understanding of certain aspects of the problem which can be relevant to solve the whole problem.
3. Some aspects of the real world problem could be too complicated to model them complete or exactly. During the modeling process it can be clarified, using a smaller version, whether partial aspects of the model could be neglected or whether they are essential.

In any case it is essential that the simplifications be well understood and documented.

Tricks of the Trade for Monolithic Models

Using state-of-the-art commercial solvers, e. g., `XPressMP` [`XPressMP` is by Dash Optimization, http://www.dashoptimization.com] or `CPLEX` [`CPLEX` is by ILOG, http://www.ilog.com], MILP problems can be solved quite efficiently. In the case of MINLP and using global optimization techniques, the solution efficiency depends strongly on the individual problem and the model formulation. However, as stressed in [21] for both MILP and MINLP problem, it is recommended that the full mathematical structure of a problem be exploited, that appropriate reformulations of models be made, and that problem-specific valid inequalities or cuts be used. Software packages may also differ with respect to the ability of *presolving techniques*, *default strategies* for the branch-and-bound algorithm, *cut generation* within the branch-and-cut algorithm, and, last but not least, *diagnosing and tracing infeasibilities*, which is an important issue in practice.

Here we collect a list of recommendation tricks that help to improve the solution procedure of monolithic MIP problems, i. e., standalone models that are solved by one call to a MILP or MINLP solver. Among them are:

- Use bounds instead of constraints if the dual values are not necessarily required.
- Apply one's own presolving techniques. Consider, for instance, a set of inequalities

$$B_{ijk}\delta_{ijk} \le A_{ijk}\ ; \quad \forall\{i,j,k\} \tag{2}$$

on binary variables δ_{ijk}. They can be replaced by the bounds

$$\delta_{ijk} = 0\ ; \quad \forall\{(i,j,k)\,|A_{ijk} < B_{ijk}\}$$

or, if one does not trust the $<$ in a modeling language, the bounds

$$\delta_{ijk} = 0\ ; \quad \forall\{(i,j,k)\,|A_{ijk} \le B_{ijk} - \varepsilon\}$$

where $\varepsilon > 0$ is a small number, say, of the order of 10^{-6}. If $A_{ijk} \ge B_{ijk}$, then (2) is redundant. Note that, due to the fact that we have three indices, the number of inequalities can be very large.
- Exploit the *presolving techniques* embedded in the solver; cf. [28].
- Exploit or eliminate symmetry: sometimes, symmetry can lead to degenerate scenarios. There are situations, for instance, in scheduling where orders can be allocated to identical production units. Another example is the capacity design problem of a set of production units to be added to a production network. In that case, symmetry can be broken by requesting that the capacities of the units be sorted in descending order, i. e., $c_u \ge c_{u+1}$. [29] exploit symmetry in order allocation for stock cutting in the paper industry; this is a very enjoyable paper to read.
- Use special types of variables for which tailor-made branching rules exist (this applies to semicontinuous and partial-integer variables as well as special ordered sets).
- Experiment with the various *strategies* offered by the commercial branch-and-bound solvers for the branch-and-bound algorithm.
- Experiment with the *cut generation* within the commercial branch-and-cut algorithm, among them Gomory cuts, knapsack cuts, or flow cuts; cf. [28].
- Construct one's own valid inequalities for certain substructures of problems at hand. Those inequalities may be added a priori to a model, and in the extreme case they would describe the complete convex hull. As an example we consider the mixed-integer inequality

$$x \le C\lambda\ , \quad 0 \le x \le X\ ; \quad x \in \mathbb{R}_0^+\ , \quad \lambda \in \mathbb{N} \tag{3}$$

which has the valid inequality

$$\begin{gathered} x \le X - G(K-\lambda) \quad \text{where} \\ K := \left\lceil \frac{X}{C} \right\rceil \quad \text{and} \quad G := X - C\,(K-1)\ . \end{gathered} \tag{4}$$

This valid inequality (4) is the more useful, the more K and X/C deviate. A special case arising is often the situation $\lambda \in \{0,1\}$. Another example, taken from [39], p. 129 is

$$A_1\alpha_1 + A_2\alpha_2 \le B + x \qquad x \in \mathbb{R}_0^+ \qquad \alpha_1, \alpha_2 \in \mathbb{N} \tag{5}$$

which for $B \notin \mathbb{N}$ leads to the valid inequality

$$\lfloor A_1 \rfloor \alpha_1 + \left(\lfloor A_2 \rfloor \alpha_2 + \frac{f_2 - f}{1-f} \right) \le \lfloor B \rfloor + \frac{x}{1-f} \tag{6}$$

where the following abbreviations are used:

$$f := B - \lfloor B \rfloor \,, \qquad f_1 := A_1 - \lfloor A_1 \rfloor \,, \quad f_2 := A_2 - \lfloor A_2 \rfloor \,. \tag{7}$$

The dynamic counterpart of valid inequalities added a priori to a model leads to cutting-plane algorithms that avoid adding a large number of inequalities a priori to the model (note, this can be equivalent to finding the complete convex hull). Instead, only those useful in the vicinity of the optimal solution are added dynamically. For the topics of valid inequalities and cutting-plane algorithms the reader is referred to books by Nemhauser and Wolsey [30], Wolsey [39], and Pochet and Wolsey [32].

- Try disaggregation in MINLP problems. Global optimization techniques are often based on convex underestimators. Univariate functions can be treated easier than multivariate terms. Therefore, it helps to represent bilinear or multilinear terms by their disaggregated equivalences. As an example we consider x_1x_2 with given lower and upper bounds X_i^- and X_i^+ for x_i; $i = 1,2$. Wherever we encounter x_1x_2 in our model we can replace it by

$$x_1x_2 = \frac{1}{2}(x_{12}^2 - x_1^2 - x_2^2)$$

and

$$x_{12} = x_1 + x_2 \,.$$

The auxiliary variable is subject to the bounds $X_{12}^- := X_1^- + X_2^-$ and

$$X_{12}^- \le x_{12} \le X_{12}^+ \,, \qquad X_{12}^- := X_1^- + X_2^- \,, \quad X_{12}^+ := X_1^+ + X_2^+ \,.$$

This formulation has another advantage. It allows us to construct easily a relaxed problem which can be used to derive a useful lower bound. Imagine a problem $\mathcal{P}$ with the inequality

$$x_1x_2 \le A \,. \tag{8}$$

Then

$$x_{12}^2 - X_1^- x_1 - X_2^- x_2 \le 2A \tag{9}$$

is a relaxation of $\mathcal{P}$ as each point (x_1, x_2) satisfying (8) also fulfills (9). Note that an alternative disaggregation avoiding an additional variable is given by

$$x_1x_2 = \tfrac{1}{4}\left[(x_1 + x_2)^2 - (x_1 - x_2)^2\right] \,.$$

However, all of the creative attempts listed above may not suffice to solve the MIP using one monolithic model. That is when we should start looking at solving the problem by a sequence of problems. We have to keep in mind that to solve a MIP problem we need to derive tight lower and upper bounds with the gap between them approaching zero.

Decomposition Techniques

Decomposition techniques decompose a problem into a set of smaller problems that can be solved in sequence or in any combination. Ideally, the approach can still compute the global optimum. There are standardized techniques such as Benders Decomposition [cf. Floudas ([9], Chap. 6). But often one should exploit the structure of an optimization to construct tailor-made decompositions. This is outlined in the following subsections.

Column Generation

In linear programming parlance, the term *column* usually refers to variables. In the context of column-generation techniques it has wider meaning and stands for any kind of objects involved in an optimization problem. In vehicle routing problems a column might, for instance, represent a subset of orders assigned to a vehicle. In network flow problems a column might represent a feasible path through the network. Finally, in cutting-stock problems [10,11] a column represents a pattern to be cut.

The basic idea of column generation is to decompose a given problem into a master and subproblem. Problems that might otherwise be nonlinear can be completely solved by solving only linear problems. The critical issue is to generate master and subproblems that can both be solved efficiently. One of the most famous examples is the elegant column-generation approach of Gilmore and Gomory [10] for computing the minimal number of rolls to satisfy a requested demand for smaller sized rolls. This problem, if formulated as one monolithic problem, leads to a MINLP problem with a large number of integer variables. In simple cases, such as those described by Schrage ([35], Sect. 11.7), it is possible to generate all columns explicitly, even within a modeling language. Often the decomposition has a natural interpretation. If not all columns can be generated, the columns are added dynamically to the problem. Barnhart et al. [2] give a good overview on such techniques. A more recent review focusing on selected topics of column generation is [25]. In the context of vehicle routing problems, feasible tours contain additional columns as needed by solving a shortest-path problem with time windows and capacity constraints using dynamic programming [7].

More generally, column-generation techniques are used to solve well-structured MILP problems involving a huge number, say, several hundred thousand or millions, of variables, i. e., columns. Such problems lead to large LP problems if the integrality constraints of the integer variables are relaxed. If the LP problem contains so many variables (columns) that it cannot be solved with a direct LP solver (revised simplex, interior point method), one starts solving this so-called *master problem* with a small subset of variables yielding the *restricted master problem*. After the restricted master problem has been solved, a pricing problem is solved to identify new variables. This step corresponds to the identification of a nonbasic variable to be taken into the basis of the simplex algorithm and the term *column generation*. The restricted master problem is solved with the new number of variables. The method terminates when the pricing problems cannot identify any new variables. The simplest version of column generation is found in the Dantzig–Wolfe decomposition [6].

Gilmore and Gomory [10,11] were the first to generalize the idea of dynamic column generation to an integer programming (IP) problem: the cutting-stock problem. In this case, the pricing problem, i. e., the subproblem, is an IP problem itself – and one refers to this as a *column-generationalgorithm*. This problem is special as the columns generated when solving the relaxed master problem are sufficient to get the optimal integer feasible solution of the overall problem. In general this is not so. If not only the subproblem, but also the master problem involves integer variables, then the column-generation part is embedded into a branch-and-bound method; this is called *branch-and-price*. Thus, branch-and-price is integer programming with column generation. Note that during the branching process new columns are generated; therefore the name *branch-and-price*.

Column Generation in cutting-stock Problems This section describes the mathematical model for minimizing the number of roles or trimloss and illustrates the idea of column generation.

Indices used in this model:

$p \in \mathcal{P} := \{p_1, \ldots, p_{N^{\text{P}}}\}$ for cutting patterns (formats).
Either the patterns are directly generated according to a complete enumeration or they are generated by column generation.
$i \in \mathcal{I} := \{i_1, \ldots, i_{N^{\text{I}}}\}$ given orders or widths.

Input Data We arrange the relevant input data size here:

B	[L] width of the rolls (raw material roles)
D_i	[-] number of orders for the width i
W_i	[L] width of order type i

Integer Variables used in the different model variants:

$\mu_p \in \mathbb{N}_0 := \{0, 1, 2, 3, \ldots\}$ [−] indicates how often pattern p is used.
If cutting pattern p is not used, then we have $\mu_p = 0$.
$\alpha_{ip} \in \mathbb{N}_0$ [−] indicates how often width i is contained in pattern p.
This variable can take values between 0 and D_i depending on the order situation.

Model The model contains a suitable object function

$$\min f(\alpha_{ip}, \mu_p) ,$$

as well as the boundary condition (fulfillment of the demand)

$$\sum_p \alpha_{ip}\mu_p = D_i , \quad \forall i \tag{10}$$

and the integrality constraints

$$\begin{aligned} &\alpha_{ip} \in \mathbb{N}_0 , \quad \forall\{ip\} , \\ &\mu_p \in \mathbb{N}_0 , \quad \forall\{p\} . \end{aligned} \tag{11}$$

General Structure of the Problem In this form it is a mixed-integer nonlinear optimization problem (MINLP). This problem class is difficult in itself. More serious is the fact that we may easily encounter several million variables α_{ip}. Therefore the problem cannot be solved in this form.

Solution Method The idea of dynamic column generation is based on the fact that one must decide in a master problem for a predefined set of patterns how often every pattern must be used as well as calculate suitable input data for a subproblem. In this subproblem new patterns are calculated.

The master problem solves for the multiplicities of existing patterns and has the shape

$$\min \sum_p \mu_p ,$$

with the demand-fulfill inequality (note that it is allowed to produce more than requested)

$$\sum_i N_{ip}\mu_p \geq D_i , \quad \forall i \tag{12}$$

and the integrality constraints

$$\mu_p \in \mathbb{N}_0 , \quad \forall\{p\} . \tag{13}$$

The subproblem generates new patterns. Structurally it is a knapsack problem with object function

$$\min_{\alpha_i} 1 - \sum_p P_i\alpha_i ,$$

where P_i are the dual values (pricing information) of the master problem (pricing problem) associated with (12) and α_i is an integer variable specifying how often width i occurs in the new pattern. We add the knapsack constraint with respect to the width of the rolls

$$\sum_i W_i\alpha_i \leq B , \quad \forall i \tag{14}$$

and the integrality constraints

$$\alpha_i \in \mathbb{N}_0 , \quad \forall\{i\} . \tag{15}$$

In some cases, α_i could be additionally bounded by the number, K, of knives.

Implementation Issues The critical issues in this method, in which we alternate in solving the master problem and the subproblem, are the initialization of the procedure (a feasible starting point is to have one requested width in each initial pattern, but this is not necessarily a good one), excluding the generation of the existing pattern by applying integer cuts, and the termination.

Column Enumeration

Column enumeration is a special variant of column generation and is applicable when a small number of columns is sufficient. This is, for instance, the case in real-world cutting-stock problems when it is known that the optimal solution has only a small amount of trimloss. This usually eliminates most of the pattern. Column enumeration naturally leads to a type of selecting columns or partitioning models. A collection of illustrative examples contained in ([35], Sect. 11.7) covers several problems of grouping, matching, covering, partitioning, and packing in which a set of given objects has to be grouped into subsets to maximize or minimize some objective function. Despite the limitations with respect to the number of columns, column enumeration has some advantages:

- No pricing problem,
- Easily applied to MIP problems,
- Column enumeration is much easier to implement.

In the online version of the vehicle routing problem described in [22] it is possible to generate the complete set, C_r, of all columns, i. e., subsets of orders $i \in \mathcal{O}$, $r = |\mathcal{O}|$, assigned to a fleet of n vehicles, $v \in \mathcal{V}$. Let C_r be the union of the sets, C_{rv}, i. e., $C_r = \cup_{v=1\ldots n} C_{rv}$ with $C_r = |C_r| = 2^r n$, where C_{rv}

contains the subsets of orders assigned to vehicle v. Note that C_{rv} contains all subsets containing 1, 2, or r orders assigned to vehicle v. The relevant steps of the algorithm are:

1. Explicitly generate all columns C_{rv}, followed by a simple feasibility test *w.r.t.* the availability of the cars.
2. Solve the routing-scheduling problem for all columns C_{rv} using a tailor-made branch-and-bound approach (the optimal objective function values, $Z(\tau_c)$ or $Z(\tau_{cv})$, respectively, and the associated routing-scheduling plan are stored).
3. Solve the partitioning model:

$$\min_{\gamma_{cv}} \sum_{c=1}^{C_{rv}} \sum_{v=1}^{N^V} Z(\tau_{cv})\gamma_{cv} \,, \tag{16}$$

s.t.

$$\sum_{c=1}^{C_r} \sum_{v=1}^{N^V} I_i(\tau_{cv})\gamma_{cv} = 1 \,, \quad \forall i = 1,\dots,r \tag{17}$$

ensures that each order is contained exactly once, the inequality

$$\sum_{c=1}^{C_r} \gamma_{cv} \le 1 \,, \quad \forall v \in \mathcal{V} \,, \tag{18}$$

ensuring that at most one column can exist for each vehicle, and the integrality conditions

$$\gamma_{cv} \in \{0,1\} \,, \quad \forall c = 1,\dots,C_r \,. \tag{19}$$

Note that not all combinations of index pairs $\{c, v\}$ exist; each c corresponds to exactly one v, and vice versa. This formulation allows us to find optimal solutions with the defined columns for a smaller number of vehicles. The objective function and the partitioning constraints are just modified by substituting

$$\sum_{v=1|v\in\mathcal{V}}^{N^V} \longrightarrow \sum_{v=1|v\in\mathcal{V}_*}^{N^V} \,,$$

the equations

$$\sum_{c=1}^{C_{rv}} \sum_{v=1|v\in\mathcal{V}_*}^{N^V} I_i(\tau_{cv})\gamma_{cv} = 1 \,, \quad \forall i = 1,\dots,r \,,$$

and the inequality

$$\sum_{c=1}^{C_{rv}} \gamma_{cv} \le 1 \,, \quad \forall v \in \mathcal{V}_* \,,$$

where $\mathcal{V}_* \subset \mathcal{V}$ is a subset of the set $\mathcal{V}$ of all vehicles. Alternatively, if it is not prespecified which vehicles should be used but it is only required that not more than N_*^V vehicles be used, then the inequality

$$\sum_{c=1}^{C_r} \sum_{v=1|v\in\mathcal{V}}^{N^V} \gamma_{cv} \le N_*^V \tag{20}$$

is imposed.

4. Reconstruct the complete solution and extract the complete solution from the stored optimal solutions for the individual columns.

Branch-and-Price

Branch-and-price (often coupled with branch-and-cut) refers to a tailor-made algorithm exploiting the decomposition structure of the problem to be solved. This efficient method for solving MIP problems with column generation has been well described by Barnhart et al. [2] and has been covered by Savelsbergh [34] in the first edition of the *Encyclopedia of Optimization*. Here, we give a list of more recent successful applications in various fields.

- *Cutting stock*: [3,38]
- *Engine routing and industrial in-plant railroads*: [26]
- *Network design*: [16]
- *Lot sizing*: [38]
- *Scheduling (staff planning)*: [8]
- *Scheduling of switching engines*: [24]
- *Supply chain optimization* (pulp industry): [5]
- *Vehicle routing*: [7,15]

Rolling Time Decomposition

The overall methodology for solving the medium-range production scheduling problem is to decompose the large and complex problem into smaller short-term scheduling subproblems in successive time horizons, i. e., we decompose according to time. Large-scale industrial problems have been solved by Janak et al. [18,19]. A decomposition model is formulated and solved to determine the current horizon and corresponding products that should be included in the current subproblem. According to the solution of the decomposition model, a short-term scheduling model is formulated using the information on customer orders, inventory levels, and processing recipes. The resulting MILP problem is a large-scale complex problem that requires a large computational effort for its solution. When a satisfactory solution is determined, the relevant data are output and the next time horizon is considered. The above procedure is applied iteratively in an automatic fashion until the whole scheduling period under consideration is finished.

Note that the decomposition model determines automatically how many days and products to consider in the small scheduling horizon subject to an upper limit on the complexity of the resulting mathematical model.

An Exhaustion Method

This method combines aspects of a constructive heuristics and of exact model solving. We illustrate the exhausting method by the cutting-stock problem described in Sect. "Column Generation in cutting-stock Problems"; assigning orders in a scheduling problem would be another example. The elegant column generation approach by Gilmore and Gomory [10] is known for producing minimal trimloss solutions with *many* patterns. Often this corresponds to setup changes on the machine and therefore is not desirable. A solution with a minimal number of patterns minimizes the machine setup costs of the cutter. Minimizing simultaneously trimloss and the number of patterns is possible for a small case of a few orders only exploiting the MILP model by Johnston and Salinlija [20]. It contains two conflicting objective functions. Therefore one could resort to goal programming. Alternatively, we could produce several parameterized solutions leading to different numbers of rolls to be used and patterns to be cut from which the user would extract the one he likes best.

As the table above indicates, we compute tight lower bounds on both trimloss and the number of patterns. Even for up to 50 feasible orders, near-optimal solutions are constructed in less than a minute.

Note that it would be possible to use the branch-and-price algorithm described in [38] or [3] to solve the one-dimensional cutting-stock problem with minimal numbers of patterns. However, these methods are not easy to implement. Therefore, we use the following approaches, which are much easier to program:

- V1: Direct usage of the model by Johnston and Salinlija [20] for a small number, say, $N^I \leq 14$, of orders and $D_{\max} \leq 10$. In a preprocessing step we compute valid inequalities as well as tight lower and upper bounds on the variables.
- V2: Exhaustion procedure in which we generate successively new patterns with maximal multiplicities. This method is parameterized by the permissible percentage waste $W_{\max}$, $1 \leq W_{\max} \leq 99$. After a few patterns have been generated with this parameterization, it could happen that is is not possible to generate any more patterns with waste restriction. In this case the remaining unsatisfied orders are generated by V1 without the $W_{\max}$ restriction.

Indices and Sets

In this model we use the indices listed in Johnston and Salinlija [20]:

$i \in \mathcal{I} := \{i_1, \ldots, i_{N^I}\}$ denotes the sets of width.

$j \in | := \{j_1, \ldots, j_{N^P}\}$ denotes the pattern; $N^J \leq N^I$.
The patterns are generated by V1, or dynamically by maximizing the multiplicities of a used pattern.

$k \in \mathcal{K} := \{k_1, \ldots, k_{N^P}\}$ denotes the multiplicity index to indicate how often a width is used in a pattern.
The multiplicity index can be restricted by the ratio of the width of the orders and the width of the given rolls.

Variables

The following integer or binary variables are used:

$a_{ijk} \in \mathbb{N}$ [−] specifies the multiplicity of pattern j.
The multiplicity can vary between 0 and $D_{\max} := \max\{D_i\}$. If pattern j is not used, we have $r_j = p_j = 0$.

$p_j \in \{0, 1\}$ [−] indicates whether pattern j is used at all.

$r_j \in \mathbb{N}$ [−] specifies how often pattern j is used.
The multiplicity can vary between 0 and $D_{\max} := \max\{D_i\}$. If pattern j is not used, we have $r_j = p_j = 0$.

$\alpha_{ip} \in \mathbb{N}$ [−] specifies how often width i occurs in pattern p.

```
 # of   #   output file   flag Wmax            comment
rolls pat
---------------------------------------------------------------------
  0    5                 8   99   lower bound: minimal # of patterns
  30  10   pat00.out     9   99   lower bound: minimal # of rolls
  34   7   pat01.out     0   20
  31   9   pat02.out     1   15
  30   8   pat03.out     0   10   minimal number of rolls
  32   9   pat04.out     1    8
  30   8   pat05.out     0    6   minimal number of rolls
  31   8   pat06.out     1    4

 The best solution found contains 7 patterns!
 The solution with minimal trimloss contain 30 rolls!

 Improvement in the lower bound of pattern: 6!
 Solutions with 6 patterns are minimal w.r.t.
 to the number of patterns.

 A new solution was found with only 6 patterns and 36 rolls: patnew.out
  36   6   patnew.out   0   99
```

This width-multiplicity variable can take all values between 0 and D_i.

$x_{ijk} \in \{0,1\}$ [−] indicates whether width i appears in pattern j at level k.
Note that $x_{ijk} = 0$ implies $a_{ijk} = 0$.

The Idea of the Exhaustion Method

In each iteration we generate m at most two or three new patterns by maximizing the multiplicities of these patterns, allowing no more than a maximum waste, $W_{\max}$. The solution generated in iteration m is preserved in iteration $m+1$ by fixing the appropriate variables. If the problem turns out to be infeasible (this may happen if $W_{\max}$ turns out to be restrictive), then we switch to a model variant in which we minimize the number of patterns subject to satisfying the remaining unsatisfied orders.

The model is based on the inequalities (2,3,5,6,7,8,9) in [20], but we add a few more additional ones or modify the existing ones. We exploit two objective functions: maximizing the multiplicities of the patterns generated

$$\max \sum_{j=1}^{\pi_u} r_j \,,$$

where π_u specifies the maximal number of patterns (π_u could be taken from the solution of the column-generation approach, for instance), or minimizing the number of patterns generated

$$\min \sum_{j=1}^{\pi_u} p_j \,.$$

The model is completed by the integrality conditions

$$r_j, a_{ijk} \in \{0,1,2,3,\ldots\} \tag{21}$$

$$p_j, x_{ijk}, y_{jk} \in \{0,1\} \,. \tag{22}$$

The model is applied several times with $a_{ijk} \le \tilde{D}_i$, where $\tilde{D}_i$ is the number of remaining orders of width i. In particular, the model has to fulfill the relationships

$$ka_{ijk} > \tilde{D}_i \quad \Longrightarrow \quad a_{ijk} = 0 \quad , \quad x_{ijk} = 0$$

and

$$a_{ijk} \le \left\lceil \frac{\tilde{D}_i}{k} \right\rceil \quad \text{or} \quad a_{ijk} \le \left\lceil \frac{\tilde{D}_i + S_i}{k} \right\rceil ,$$

where S_i denotes the permissible overproduction.

The constructive method described so far provides an improved upper bound, π'_u, on the number of pattern.

Computing Lower Bounds

To compute a lower bound we apply two methods. The first method is to solve a bin-packing problem, which is equivalent to minimizing the number of rolls in the original cutting-stock problem described in the Sect. "Column Generation in Cutting-Stock Problems" for equal demands $D_i = 1$. If solved with the column-generation approach, this method is fast and cheap, but the lower bound, π'_l, is often weak. The second method is to exploit the upper bound, π'_u, on the number of patterns obtained and to call the exact model as in V1. It is impressive how quickly the commercial solvers CPLEX and XpressMP improve the lower bound yielding π''_l. For most examples with up to 50 orders we obtain $\pi'_u - \pi'_l \leq 2$, but in many cases $\pi'_u - \pi'_l = 1$ or even $\pi'_u = \pi''_l$.

Primal Feasible Solutions and Hybrid Methods

We define hybrid methods as methods based on any combination of exact MIP methods with constructive heuristics, local search, metaheuristics, or constraint programming that produces primal feasible solutions. Dive-and-fix, near-integer-fix, and fix-and-relax are such hybrid methods. They are user-developed heuristics exploiting the problem structure. In their kernel they use a declarative model solved, for instance, by CPLEX and XpressMP.

In constructive heuristics we exploit the structure of the problem and compute a feasible point. Once we have a feasible point we can derive safe bounds on the optimum and assign initial values to the critical discrete variable, which could be exploited by the GAMS/CPLEX *mipstart* option. Feasible points can sometimes be generated by appropriate sequences of relaxed models. For instance, in a scheduling problem $\mathcal{P}$ with due times one might relax these due times obtaining the relaxed model $\mathcal{R}$. The optimal solution, or even any feasible point of $\mathcal{R}$, is a feasible point of $\mathcal{P}$ if the due times are models with appropriate unbounded slack variables.

Constructive heuristics can also be established by systematic approaches of fixing critical discrete variables. Such approaches are *dive-and-fix* and *relax-and-fix*. In dive-and-fix the LP relaxation of an integer problem is to be solved followed by fixing a subset of fractional variables to suitable bounds. *Near-integer-fix* is a variant of dive-and-fix that fixes variables with fractional values to the nearest integer point. Note that these heuristics are subject to the risk of becoming infeasible.

The probability of becoming infeasible is less likely in relax-and-fix. In relax-and-fix, following Pochet and Wolsey ([32], pp. 109) we suppose that the binary variables δ of a MIP problem $\mathcal{P}$ can be partitioned into R disjoint sets $S^1; \ldots ; S^R$ of decreasing importance. Within these subsets U^r with $U \subseteq \cup_{u=r+1}^{R} S^u$ for $r = 1; \ldots ; R-1$ can be chosen to allow for somewhat more generality. Based on these partitions, R MIP problems are solved, denoted $\mathcal{P}^r$ with $1 \leq r \leq R$ to find a heuristic solution to $\mathcal{P}$. For instance in a production planning problem, S^1 might be all the δ variables associated with time periods in $\{1, \ldots, t_1\}$, S^u those associated with periods in $\{t_u + 1, \ldots, t_{u+1}\}$, whereas U^r would would be the δ variables associated with the periods in some set $\{t_r + 1, \ldots, u_r\}$.

In the first problem, $\mathcal{P}^1$, one only imposes the integrality of the important variables in $S^1 \cup U^1$ and relaxes the integrality on all the other variables in S. As $\mathcal{P}^1$ is a relaxation of $\mathcal{P}$, for a minimization problem, the solution of $\mathcal{P}^1$ provides a lower bound of $\mathcal{P}$. The solution values, δ^1, of the discrete variables are kept fixed when solving $\mathcal{P}^r$. This continues and in the subsequent $\mathcal{P}^r$, for $2 \leq r \leq R$, we additionally fix the values of the δ variables with index in S^{r-1} at their optimal values from $\mathcal{P}^{r-1}$ and add the integrality restriction for the variables in $S^r \cup U^r$.

Either $\mathcal{P}^r$ is infeasible for some $r \in \{1, \ldots, R\}$, and the heuristic failed, or else (x^R, δ^R) is a relax-and-fix solution. To avoid infeasibilities one might apply a smoothed form of this heuristic that allows for some overlap of U^{r-1} and U^r. Additional free binary variables in horizon $r-1$ allow one to link the current horizon r with the previous one. Usually this suffices to ensure feasibility. Relax-and-fix comes in various flavors exploiting time-decomposition or time-partitioning structures. Other decompositions, for instance plants, products, or customers, are possible as well.

A local search can be used to improve the solution obtained by the relax-and-fix heuristic. The main idea is to solve repeatedly the subproblem on a small number of binary variables reoptimizing, for instance, the production of some products. The binary variables for resolving could be chosen randomly or by a metaheuristic such as simulated annealing. All binary variables related to them are released; the others are fixed to the previous best values.

Another class of MIP hybrid method is established by algorithms that combine a MIP solver with another algorithmic method. A hybrid method obtained by the combination of mixed-integer and constraint logic programming strategies has been developed and applied by Harjunkoski et al. [14] as well as Jain and Grossmann [17] for solving scheduling and combinatorial optimization problems. Timpe [37] solved mixed planning and scheduling problems with mixed MILP branch-and-bound and constraint programming. Maravelias and Grossmann [27] proposed a hybrid/decomposiiton algorithm for the short-term scheduling of batch plants, and Roe et al. [33] presented a hybrid MILP/CLP algorithm for multipurpose batch process scheduling in which MILP is used to solve an aggregated planning problem while CP is used to solve a sequencing problem. Other hybrid algorithms combine evolutionary and mathematical programming methods; see, for instance, the heuristics by Till et al. [36] for stochastic scheduling problems and by Borisovsky et al. [4] for supply management problems.

Finally, one should not forget to add some algorithmic component that, for the minimization problem at hand, would generate some reasonable bounds to be provided in addition to the hybrid method. The hybrid methods discussed above provide upper bounds by constructing feasible points. In favorite cases, the MIP part of the hybrid solver provides lower bounds. In other case, lower bounds can be derived from auxiliary problems, which are relaxations of the original problem, and which are easier to solve.

Summary

If a given MIP problem cannot be solved by an available MIP solver exploiting all its internal presolving techniques, one might reformulate the problem and obtain an equivalent or closely related representation of reality. Another approach is to construct MIP solutions and bounds by solving a sequence of models. Alternatively, individual tailor-made exact decomposition techniques could help as well as primal heuristics such as relax-and-fix or local search techniques on top of a MIP model.

References

1. Anstreicher KM (2001) Linear Programming: Interior Point Methods. In: Floudas CA, Pardalos P (eds) Encyclopedia of Optimization, vol 3. Kluwer, Dordrecht, pp 189–191
2. Barnhart C, Johnson EL, Nemhauser GL, Savelsberg MWP, Vance PH (1998) Branch-and-price: column generation for solving huge integer programs. Oper Res 46(3):316–329
3. Belov G, Scheithauer G (2006) A Branch-and-Crice Algorithm for One-Dimensional Stock Cutting and Two-Dimensional Two-Stage Cutting. Eur J Oper Res 171:85–106
4. Borisovsky P, Dolgui A, Eremeev A (2006) Genetic Algorithms for Supply Management Problem with Lower-bounded Demands. In: Dolgui A, Morel G, Pereira C (eds) Information Control Problems in Manufacturing 2006: A Proceedings volume from the 12th IFAC International Symposium. vol. 3., St Etienne, France, 17-19 May 2006. North-Holland, Dordrecht, pp 521–526
5. Bredström D, Lundgren JT, Rñnqvist M, Carlsson D, Mason A (2004) Supply Chain Optimization in the Pulp Mill Industry – IP models, Column Generation and Novel Constraint Branches. Eur J Oper Res 156:2–22
6. Dantzig B, Wolfe P (1960) The decomposition algorithm for linear programming. Oper Res 8:101–111
7. Desrochers M, Desrosiers J, Solomon MM (1992) A New Optimization Algorithm for the Vehicle Routing Problem with time Windows. Oper Res 40(2):342–354
8. Eveborn P, Ronnqvist M (2004) Scheduler – A System for Staff Planning. Ann Oper Res 128:21–45
9. Floudas CA (1995) Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. Oxford University Press, Oxford
10. Gilmore PC, Gomory RE (1961) A Linear Programming Approach to the Cutting Stock Problem. Oper Res 9:849–859
11. Gilmore PC, Gomory RE (1963) A Linear Programming Approach to the Cutting Stock Problem, Part II. Oper Res 11:863–888
12. Grötschel M (2004) Mathematische Optimierung im industriellen Einsatz. In: Lecture at Siemens AG, Munich, Germany, 7 Dec 2004
13. Grötschel M (2005) Private communication
14. Harjunkoski I, Jain V, Grossmann IE (2000) Hybrid Mixed-integer constraint Logic Programming Strategies for Solving Scheduling and Combinatorial Optimization Problems. Comput Chem Eng 24:337–343
15. Irnich S (2000) A Multi-Depot Pickup and Delivery Problem with a Single Hub and Heterogeneous Vehicles. Eur J Oper Res 122:310–328
16. Irnich S (2002) Netzwerk-Design für zweistufige Transportsysteme und ein Branch-and-Price-Verfahren für das gemischte Direkt- und Hubflugproblem. Dissertation, Fakultät für Wirtschaftswissenschaften, RWTH Aachen, Aachen, Germany

17. Jain V, Grossmann IE (2001) Algorithms for hybrid MILP/CP models for a class of optimization problems. IFORMS J Comput 13:258–276
18. Janak SL, Floudas CA, Kallrath J, Vormbrock N (2006) Production Scheduling of a Large-Scale Industrial Batch Plant: I. Short-Term and Medium-Term Scheduling. Ind Eng Chem Res 45:8234–8252
19. Janak SL, Floudas CA, Kallrath J, Vormbrock N (2006) Production Scheduling of a Large-Scale Industrial Batch Plant: II. Reactive Scheduling. Ind Eng Chem Res 45:8253–8269
20. Johnston RE, Sadinlija E (2004) A New Model for Complete Solutions to One-Dimensional Cutting Stock Problems. Eur J Oper Res 153:176–183
21. Kallrath J (2000) Mixed Integer Optimization in the Chemical Process Industry: Experience, Potential and Future Perspectives. Chem Eng Res Des 78(6):809–822
22. Kallrath J (2004) Online Storage Systems and Transportation Problems with Applications: Optimization Models and Mathematical Solutions, vol 91 of Applied Optimization. Kluwer, Dordrecht
23. Kallrath J, Maindl TI (2006) Real Optimization with SAP-APO. Springer, Berlin
24. Lübbecke M, Zimmermann U (2003) Computer aided scheduling of switching engines. In: Jäger W, Krebs HJ (eds) Mathematics – Key Technology for the Future: Joint Projects Between Universities and Industry. Springer, Berlin, pp 690–702
25. Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. Oper Res 53(6):1007–1023
26. Lübbecke ME, Zimmermann UT (2003) Engine routing and scheduling at industrial in-plant railroads. Transp Sci 37(2):183–197
27. Maravelias CT, Grossmann IE (2004) A Hybrid MILP/CP Decomposition Approach for the Continuous Time Scheduling of Multipurpose Batch Plants. Comput Chem Eng 28:1921–1949
28. Martin A (2001) General Mixed Integer Programming: Computational Issues for Branch-and-Cut Algorithms. In: Naddef D, Juenger M (eds) Computational Combinatorial Optimization. Springer, Berlin, pp 1–25
29. Menon S, Schrage L (2002) Order Allocation for Stock Cutting in the Paper Industry. Oper Res 50(2):324–332
30. Nemhauser GL, Wolsey LA (1988) Integer and Combinatorial Optimization. Wiley, New York
31. Pardalos PM (2001) Linear Programming. In: Floudas CA, Pardalos P (eds) Encyclopedia of Optimization, vol 3. Kluwer, Dordrecht, pp 186–188
32. Pochet Y, Wolsey LA (2006) Production Planning by Mixed Integer Programming. Springer, Berlin
33. Roe B, Papageorgiou LG, Shah N (2005) A hybrid MILP/CLP Algorithm for Multipurpose Batch Process schedulin. Comput Chem Eng 29:1277–1291
34. Savelsbergh MWP (2001) Branch-and-Price: Integer Programming with Column Generation. In: Floudas CA, Pardalos P (eds) Encyclopedia of Optimization. Kluwer, Dordrecht, pp 218–221
35. Schrage L (2006) Optimization Modeling with LINGO. LINDO Systems, Chicago
36. Till J, Sand G, Engell S, Emmerich M, Schönemann L (2005) A New Hybrid Algorithm for Solving Two-Stage Stochastic Problems by Combining Evolutionary and Mathematical Programming Methods. In: Puigjaner L, Espuña A (eds) Proc. European Symposium on Computer Aided Process Engineering (ESCAPE) - 15. Dordrecht, North-Holland, pp 187–192
37. Timpe C (2002) Solving mixed planning and scheduling problems with mixed branch and bound and constraint programming. OR Spectrum 24:431–448
38. Vanderbeck F (2000) Exact algorithm for minimising the number of setups in the one-dimensional cutting stock problem. Oper Res 48(5):915–926
39. Wolsey LA (1998) Integer Programming. Wiley, New York

Modeling Languages in Optimization: A New Paradigm

TONY HÜRLIMANN
Institute Informatics, University Fribourg,
Fribourg, Switzerland

MSC2000: 90C10, 90C30

Article Outline

Keywords

Algorithmic language; Declarative language; Modeling language; Solver

In this paper, modeling languages are identified as a new computer language paradigm and their applications for representing optimization problems is illustrated by examples.

Programming languages can be classified into three paradigms: imperative, functional, and logic programming [14]. The *imperative programming paradigm* is closely related to the physical way of how (the von Neumann) computer works: Given a set of memory locations, a program is a sequence of well defined instructions on retrieving, storing and transforming the content of these locations. The *functional paradigm* of computation is based on the evaluation of functions. Every program can be viewed as a function which translates an input into a unique output. Functions are first-class values, that is, they must be viewed as values themselves. The computational model is based on the λ-calculus invented by A. Church (1936) as a mathematical formalism for expressing the concept of a computation. The *paradigm of logic programming* is based on the insight that a computation can be viewed as a kind of (constructive) proof. Hence, a program is a notation for writing logical statements together with specified algorithms for implementing inference rules.

All three programming paradigms concentrate on problem representation as a *computation*, that is, the problem is stated in a way that describes the process of solving it. The computation on how to solve a problem 'is' its representation. One may call such a notational system an *algorithmic language*.

Definition 1 An *algorithmic language* describes (explicitly or implicitly) the computation of solving a problem, that is, 'how' a problem can be processed using a machine. The computation consists of a sequence of well-defined instructions which can be executed in a finite time by a Turing machine. The information of a problem which is captured by an algorithmic language is called *algorithmic knowledge* of the problem.

Algorithmic knowledge to describe a problem is very common in our everyday life – one only need to look at cookery-books, or technical maintenance manuals – that one may ask whether the human brain is 'predisposed' to preferably present a problem in describing its solution recipe.

However, there exists at least one different way to capture knowledge about a problem; it is the method which describes 'what' the problem is by defining its properties, rather than saying 'how' to solve it. Mathematically, this can be expressed by a set $\{x \in X \colon R(x)\}$, where X is a continuous or discrete state space and $R(x)$ is a Boolean relation, defining the properties or the *constraints* of the problem; x is called the *variable*(s). A notational system that represents a problem in this way is called a *declarative language*.

Definition 2 A *declarative language* describes the problem as a set using mathematical variables and constraints defined over a given state space. This space can be finite or infinite, countable or noncountable. The information of a problem which is captured by a declarative language is called *declarative knowledge* of the problem.

The declarative representation, in general, does not give any indication on how to solve the problem. It only states what the problem is. Of course, there exists a trivial algorithm to solve a declaratively stated problem, which is to enumerate the state space and to check whether a given $x \in X$ violates the constraint $R(x)$. The algorithm breaks down, however, whenever the state space is infinite. But even if the state space is finite, it is – for most nontrivial problems – so large that a full enumeration is practically impossible.

Algorithmic and declarative representations are two fundamentally different kinds of modeling and representing knowledge. Declarative knowledge answers the question 'what is?', whereas algorithmic knowledge asks 'how to?' [4]. An algorithm gives an exact recipe of how to solve a problem. A mathematical model, i. e. its declarative representation, on the other hand, (only) defines the problem as a subspace of the state space. No algorithm is given to find all or a single element of the feasible subspace.

Why Declarative Representation

The question arises, therefore, why to present a problem using a declarative way, since one must solve it anyway and, hence, represent as an algorithm? The reasons are, first of all, *conciseness*, *insight*, and *documentation*. Many problems can be represented declaratively in a very concise way, while the representation of their computation is long and complex. Concise writings favor also the insight of a problem. Furthermore, in many scientific papers a problem is stated in a declarative way using mathematical equations and inequalities for documentational purposes. This gives a clear statement of the problem and is an efficient way to communicate it to other scientists. However, documentation is by no means limited to human beings. One can imagine declarative languages implemented on a computer like algorithmic languages, which are parsed and interpreted by a compiler. In this way, an interpretative system can *analyse the structure* of a declarative program, can *pretty-print* it on a printer or a screen, can *classify* it, or *symbolically transform* it in order to view it as a diagram or in another textual form.

Of course, the most interesting question is whether the declarative way of representing a problem could be of any help in *solving* the problem.

Indeed, for certain classes of problems the computation can be obtained directly from a declarative formulation. This is true for all recursive definitions. A classical example is the algorithm of Euclid to find the greatest common divisor (gcd) of two integers. One can proof that

$$\gcd(a, b) = \begin{cases} \gcd(b, a \bmod b), & b > 0 \\ a, & b = 0, \end{cases}$$

which is clearly a declarative statement of the problem. In *Scheme*, a functional language, this formula can be implemented directly as a function in the following way:

```
(define (gcd a b)
    (if(= b 0) a
        (gcd b (remainder a b))))
```

Similar formulations can be given for any other language which includes recursion as a basic control structure. This class of problems is surprisingly rich. The whole paradigm of dynamic programming can be subsumed under this class.

A class of problems of a very different kind are linear programs, which can be represented declaratively in the following way:

$$\{\min cx\colon\ Ax \geq b\}$$

From this formulation – in contrast to the class of recursive definitions – nothing can be deduced that would be useful in solving the problem. However, there exists well-known methods, for example the simplex method, which solves almost all instances in a very efficient way. Hence, to make the declarative formulation of a linear program useful for solving it, one only needs to translate it into a form, the simplex algorithm accepts as input. The translation from the declarative formulation $\{\min cx\colon Ax \geq b\}$ to such an input-form can be automated. This concept can be extended to nonlinear and discrete problems.

Algebraic Modeling Languages

The idea to state the mathematical problem in a declarative way and to translate it into an 'algorithmic' form by a standard procedure led to a new language paradigm emerged basically in the community of operations research at the end of the 1980s, the *algebraic modeling languages* (AIMMS [1], AMPL [7], GAMS [2], LINGO [18], and LPL [12] and others). These languages are becoming increasingly popular even outside the community of operations research. Algebraic modeling languages represent a problem in a purely declarative way, although most of them include computational facilities to manipulate the data as well as certain control structures.

One of their strength is the complete separation of the problem formulation as a declarative model from finding a solution, which is supposed to be computed by an external program called a *solver*. This allows the modeler not only to separate the two main tasks of model formulation and model solution, but also to switch easily between several solvers. This is an invaluable benefit for many difficult problems, since it is not uncommon that a model instance can be solved using one method, and another instance is solvable only using another method. Another advantage of such languages is to separate clearly between model structure, which only contains parameters (place-holder for data) but no data, and model instance, in which the parameters are replaced by a specific data set. This leads to a natural separation between model formulation and data gathering stored in databases. Hence, the main features of these algebraic modeling languages are:

- purely declarative representation of the problem;
- clear separation between formulation and solution;
- clear separation between model structure and model data.

It is, however, naive to think that one only needs to formulate a problem in a concise declarative form and to link it somehow to a solver in order to solve it. First of all, the 'linking process' is not so straightforward as it seems initially. Second, a solver may not exist which could solve the problem at hand in an efficient way. One only needs to look at Fermat's last conjecture which can be stated in a declarative way as $\{a, b, c, n \in \mathbf{N}^+ \colon a^n + b^n = c^n, a, b, c \geq 1, n > 2\}$ to convince oneself of this fact. Even worse, one can state a problem declaratively for which no solver can exist. This is true already for the rather limited declarative language of first order logic, for which no algorithm exists which decides whether a formula is true or false in general (see [5]).

In this sense, efforts are under way actually in the design of such languages which focus on flexibly linking the declarative formulation to a specific solver to make this paradigm of purely declarative formulation more powerful. This language-solver-interface problem has different aspects and research goes in many directions. A main effort is to integrate symbolic model transformation rules into the declarative language in order to generate formulations which are more useful for a solver. AMPL, for example, automatically detects partially separable structure and computes second derivatives [8]. This information are also handed over to a nonlinear solver. LPL, to cite a very different undertaking, has integrated a set of rules to translate *symbolically* logical constraints into 0–1 constraint [11]. To do this in an intelligent way is all but easy, because the resulting 0–1 formulation should be as sharp as possible. This translation is useful for large mathematical models which must be extended by a few logical conditions. For many applications the original model becomes straightforward while the transformed is complicated but still relatively easy to solve (examples were given in [11]). Even if the resulting formulation is not solvable efficiently, the modeler can gain more insights into the structure of the model from such a symbolic translation procedure, and eventually modify the original formulation.

Second Generation Modeling Languages

Another research activity, actually under way, goes in the direction of extending the algebraic modeling languages in order to express also algorithmic knowledge. This is necessary, because even if one could link an purely declarative language to any solver, it remains doubtful of whether this can be done efficiently in all cases. Furthermore, for many problems it is not useful to formulate them in a declarative way: the algorithmic way is more straightforward and easier to understand. For still other problems a mixture of declarative and algorithmic knowledge leads to a superior formulation in terms of understandability as well as in terms of efficiency, (examples are given below to confirm this findings).

Therefore, AIMMS integrates control structures and procedure definitions. GAMS, AMPL and LPL also allow the modeler to write algorithms powerful enough to solves models repeatedly.

A theoretical effort was undertaken in [10] to specify a modeling language which allows the modeler (or the programmer) to combine algorithmic and declarative knowledge within the same language framework without intermingle them. The overall syntax structure of a model (or a program) in this framework is as follows:

```
MODEL ModelName
    ⟨declarative part of the model⟩
BEGIN
    ⟨algorithmic part of the model⟩
END ModelName.
```

Declarative and algorithmic knowledge are clearly separated. Either part can be empty, meaning that the problem is represented in a purely declarative or in a purely algorithmic form. The declarative part consists of the basic building blocks of declarative knowledge: variables, parameters, constraints, model checking facilities, and sets (that is a way to 'multiply' basic building blocks). This part may also contain 'ordinary declarations' of an algorithmic language (e. g., type and function declarations). Furthermore, one can declare whole models within this part, leading to nested model structures, which is very useful in decomposing a complex problem into smaller parts. The algorithmic part, on the other hand, consists of all control structures which make the language Turing complete. One may imagine his or her favorite programming language being implemented in this part. A language which combines declarative and algorithmic knowledge in this way is called *modeling language*.

Definition 3 A *modeling language* is a notational system which allows one to combine (not to merge) declarative and algorithmic knowledge in the same language framework. The content captured by such a notation is called a *model*.

Such a language framework is very flexible. Purely declarative models are linked to external solvers to be solved; purely algorithmic models are programs, that is algorithms + data structures, in the ordinary sense.

Modeling Language and Constraint Logic Programming

Merging declarative and algorithmic knowledge is not new, although it is not very common in language design. The only existing language paradigm doing it is *constraint logic programming* (CLP), a refinement of logic programming [13]. There are, however, important differences between the CLP paradigm and the paradigm of modeling language as defined above.

1) In CLP the algorithmic part – normally a search mechanism – is behind the scene and the computation is intrinsically coupled with the declarative language itself. This could be a strength because the programmer does not have to be aware of how the computation is taking place, he or she only writes the rules in a descriptive, that is declarative, way and triggers the computation by a request. In reality, however, it is an important drawback, because – for most nontrivial problem – the programmer 'must' be aware on how the computation is taking place. Therefore, to guide the computation in CLP, the declarative program is interspersed with additional rules which have nothing to do with the description of the original problem. In a modeling language, the user either links the declarative part to an external solver or writes the solver within the language. In either case, both parts are strictly separated. Why is this separation so important? Because it allows the modeler to 'plug in' different solvers without touching the overall model formulation.
2) The second difference is that the modeling language paradigm lead automatically to modular design. This is probably to hottest topic in software engineering: building components. Software engineering teaches us that a complex structure can be only managed efficiently by break it down into many relatively independent components. The CLP approach leads more likely to programs that are difficult to survey and hard to debug and to maintain, because such considerations are entirely absent within the CLP paradigm.
3) On the other hand, the community of CLP has developed methods to solve specific classes of combinatorial problems which seems to be superior to other methods. This is because they rely on propagation, simplification of constraints, and various consistency techniques. In this sense, CLP solvers could be used and linked with modeling languages. Such a project is actually under way between the AMPL language and the ILOG solver [6,17].

Hence, while the *representation of models* is probably best done in the language framework of modeling languages, the solution process can taken place in a CLP solver for certain problems.

Modeling Examples

Five modeling examples are chosen from very different problem domains to illustrate the highlights of the presented paradigm of modeling language. The first two examples show that certain problems are best formulated using algorithmic knowledge, the next two examples show the power of a declarative formulation, and a last example indicates that mixing both paradigms is sometimes more advantageous.

Sorting

Sorting is a problem which is preferably expressed in an algorithmic way. Declaratively, the problem could be formulated as follows: Find a permutation π such that $A_{\pi\, i} \leq A_{\pi\, i+1}$ for all $i \in \{1, \ldots, n-1\}$ where $A_1, \ldots, n$ is an array of objects on which an order is defined. It is difficult to imagine a 'solver' that could solve this problem as efficiently as the best known sorting algorithms such as Quicksort, of which the implementation is straightforward.

The reason why the sorting problem is best formulated as an algorithm is probably that the state space is exponential in the number of items, however, the best algorithm only has complexity $O(n \log n)$.

The *n*-Queens Problem

The n-queens problem is to place n queens on a chessboard of dimension $n \times n$ in such a way, that they cannot beat each other. This problem can be formulated declarative as follows: $\{x_i, x_j \in \{1, \dots, n\}: x_i \neq x_j, x_i + i \neq x_j + j, x_i - i \neq x_j - j\}$, where x_i is the column position of the ith queen (i. e. the queen in row i).

Using the LPL [12] formulation:

```
MODEL nQueens;
   PARAMETER n; SET i ALIAS j::={1, . . . , n};
   DISTINCT VARIABLE x{i}[1, . . . , n];
   CONSTRAINT S{i, j : i < j}:
      x[i] + i <> x[j] + j AND x[i] − i <> x[j] − j;
END
```

the author was able to solve problems for $n \leq 8$ using a general MIP solver. The problem is automatically translated into a 0–1 problem by LPL. Replacing the MIP-solver by a tabu search heuristic, problems with $n \leq 50$ were solvable within the LPL framework. Using the constraint language OZ [19] problems of $n \leq 200$ are efficiently solvable using techniques of propagation and variable domain reductions. However, the success of all these methods seems to be limited compared to the best we can attain. In [20,21], Sosic Rok and Gu Jun presented a polynomial time local heuristic that can solve problems of $n \leq 3\,000\,000$ in less than one minute. The presented algorithm is very simple. The conclusion seems to be for the n-queens problem that an algorithmic formulation is advantageous.

A Two-Person Game

Two players choose at random a positive number and note it on a piece of paper. They then compare them. If both numbers are equal, then neither player gets a payoff. If the difference between the two numbers is one, then the player who has chosen the higher number obtains the sum of both; otherwise the player who has chosen the smaller number obtains the sum of both. What is the optimal strategy for a player, i. e. which numbers should be chosen with what frequencies to get the maximal payoff? This problem was presented in [9] and is a typical two-person zero-sum game. In LPL, it can be formulated as follows:

```
ODEL Game ‘finite two-person zero-sum game’;
   SET i ALIAS j := /1 : 50/;
   PARAMETER p{i, j} := IF(j > i, IF(j = i + 1,
   −i − j, MIN(i, j)), IF(j < i, −p[j, i], 0));
   VARIABLE x{i};
   CONSTRAINT R : SUM{i} x[i] = 1;
   MAXIMIZE gain: MIN{j}(SUM{i}p[j, i] ∗ x[i]);
END Game.
```

This is an very compact way to declaratively formulate the problem and it is difficult to imagine how this could be achieved using algorithmic knowledge alone. It is also an efficient way to state the problem, because large instances can be solved by an linear programming solver. LPL automatically transforms it into an linear program. (By the way, the problem has an interesting solution: Each player should only choose number smaller than six.)

Equal Circles in a Square

The problem is to find the maximum diameter of n equal mutually disjoint circles packed inside a unit square.

In LPL, this problem can be compactly formulated as follows:

```
MODEL circles ‘pack equal circles in a square’;
PARAMETER n ‘number of circles’;
SET i ALIAS j = 1, . . . , n;
VARIABLE
     t ‘diameter of the circles’;
     x{i}[0, 1] ‘x-position of the center’;
     y{i}[0, 1] ‘y-position of the center’;
CONSTRAINT
     R{i, j : i < j} ‘circles must be disjoint’:
          (x[i] − x[j])^2 + (y[i] − y[j])^2 ≥ t;
MAXIMIZE obj ‘maximize diameter’: t;
END
```

C.D. Maranas et al. [15] obtained the best known solutions for all $n \leq 30$ and, for $n = 15$, an even better one using an equivalent formulation in GAMS and linking it to MINOS [16], an well-known nonlinear solver.

The (Fractional) Cutting-Stock Problem

Paper is manufactured in rolls of width B. A set of customers W orders d_w rolls of width b_w(with $w \in W$). Rolls can be cut in many ways, every subset $P' \subseteq W$ such that $\sum_{i \in P'} y_i \, b_i \leq B$ is a possible cut-pattern, where y_i is a positive integer. The question is how the initial roll of width B should be cut, that is, which patterns should be used, in order to minimize the overall paper waste. A straightforward formulation of this problem is to enumerate all patterns, each giving a variable, then to minimize the number of used patterns while fulfilling the demands. The resulting model is a very large linear program which cannot be solved.

A well-known method in operations research to solve such kind of problems is to use a column generation method (see [3] for details), that is, a small instance with only a few patterns is solved and a rewarding column – a pattern – is added repeatedly to the problem. The new problem is then solved again. This process is repeated, until no pattern can be added. To find a rewarding pattern, another problem – named a knapsack problem – must be solved.

The problem can be formulated partially be algorithmic partially by declarative knowledge. It consists of two declaratively formulated problems (a linear program and an knapsack problem), which are both repeatedly solved. In a pseudocode one could formulate the algorithmic knowledge as follows:

```
SOLVE the small cutting-stock problem
SOLVE the knapsack problem
WHILE a rewarding pattern was found DO
    add pattern to the cutting-stock problem
    SOLVE the cutting-stock problem again
    SOLVE the knapsack problem again
ENDWHILE
```

The two models (the cutting-stock problem and the knapsack problem) can be formulated declaratively. In the proposed framework of modeling language, the complete problem can now be expressed as in the program below.

```
MODEL CuttingStock;
  MODEL Knapsack(i, w, p, K, x, obj);
    SET i;
    PARAMETER w{i}; p{i}; K;
    INTEGER VARIABLE x{i};
    CONSTRAINT R: SUM{i} w * x ≤ K;
    MAXIMIZE obj: SUM{i} p * x;
  END Knapsack.
  SET
    w 'rolls ordered'; p 'possible patterns';
  PARAMETER
    a{w, p} 'pattern table';
    d{w} 'demands';
    b{w} 'widths of ordered rolls';
    B 'initial width';
    INTEGER y{w} 'new added pattern';
    C 'contribution of a cut';
  VARIABLE
    X{p} 'number of rolls cut according to p';
  CONSTRAINT
    Dem{w}: SUM{p} a * X ≥ d;
  MINIMIZE obj: SUM{p} X;
BEGIN
  SOLVE;
  SOLVE Knapsack(w, b, Dem.dual, B, y, C);
  WHILE (C > 1) DO
    p := p + {'pattern_' + str(#p)};
    a{w, #p} := y[w];
    SOLVE;
    SOLVE Knapsack(w, b, Dem.dual, B, y, C);
  END;
END CuttingStock.
```

This formulation has several remarkable properties:

1) It is short and readable. The declarative part consists of the (small) linear cutting-stock problem, it also contains, as a submodel, a knapsack problem. The algorithmic part implements thecolumn generation method. Both parts are entirely separated.
2) It is a complete formulation, except from the data. No other code is needed; both models can be solved using a standard MIP solver (since the knapsack problem is small in general).
3) It has a modular structure. The knapsack problem is an independent component with its own name space; there is no interference with the surrounding model. It could even be declared outside the cutting-stock problem.

4) The cutting-stock problem is only one problem of a large class of relevant problems which are solved using a column generation or, alternatively, a row-cut generation.

Conclusion

It has been shown that certain problems are best formulated as algorithms, others in a declarative way, still others need both paradigms to be stated concisely. Computer science made available many algorithmic languages; they can be contrasted to the algebraic modeling languages which are purely declarative. A language, called modeling language, which combines both paradigms was defined in this paper and examples were given showing clear advantages of doing so. Its is more powerful than both paradigms separated.

However, the integration of algorithmic and declarative knowledge cannot be done in an arbitrary way. The language design must follow certain criteria well-known in computer science. The main criteria are: *reliability* and *transparency*. Reliability can be achieved by a unique notation to code models, that is, by a modeling language, and by various checking mechanisms (type checking, unit checking, data integrity checking and others). Transparency can be obtained by flexible decomposition techniques, like modular structure as well as access and protection mechanisms of these structure, well-known techniques in language design and software engineering.

Solving efficiently and relevant optimization problems using present desktop machine not only asks for fast machines and sophisticated solvers, but also for formulation techniques that allow the modeler to communicate the model easily and to build it in a readable and maintainable way.

See also

- ► Continuous Global Optimization: Models, Algorithms and Software
- ► Large Scale Unconstrained Optimization
- ► Optimization Software

References

1. Bisschop J (1998) AIMMS, the modeling system. Paragon Decision Techn, Haarlem, www.paragon.nl
2. Brooke A, Kendrick D, Meeraus A (1988) GAMS. A user's guide. Sci Press, Marrickville
3. Chvátal V (1973) Linear programming. Freeman, New York
4. Feigenbaum EA (1996) How the 'what' becomes the 'how'. Comm ACM 39(5):97–104
5. Floyd RW, Beigel R (1994) The language of machines, an introduction to computability and formal languages. Computer Sci Press, Rockville
6. Fourer R (1998) Extending a general-purpose algebraic modeling language to combinatorial optimization: A logic programming approach. In: Woodruff DL (ed) Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search: Interfaces in Computer Sci and Oper Res. Kluwer, Dordrecht, pp 31–74
7. Fourer R, Gay DM, Kernighan BW (1993) AMPL, a modeling language for mathematical programming. Sci Press, Marrickville
8. GAY DM (1996) Automatically finding and exploiting partially separable structure in nonlinear programming problems. AT&T Bell Lab Murray Hill, New Jersey
9. Hofstadter DR (1988) Metamagicum, Fragen nach der Essenz von Geist und Struktur. Klett-Cotta, Stuttgart
10. Hürlimann T (1997) Computer-based mathematical modeling. Habilitations Script. Fac Economic and Social Sci, Inst Informatics, Univ Fribourg
11. Hürlimann T (1998) An efficient logic-to-IP translation procedure. Working Paper, Inst Informatics, Univ Fribourg, ftp://ftp-iiuf.unifr.ch/pub/lpl/doc/APMOD1.pdf
12. Hürlimann T (1998) Reference manual for the LPL modeling language. Working Paper, version 4.30. Inst Informatics, Univ. Fribourg, Fribourg, ftp://ftp-iiuf.unifr.ch/pub/lpl/doc/Manual.ps
13. Jaffar J, Maher MJ (1995) Constraint logic programming: A survey. Handbook Artificial Intelligence and Logic Programming. Oxford Univ Press, Oxford
14. Louden KC (1993) Programming languages – Principles and practice. PWS/Kent Publ, Boston
15. Maranas CD, Floudas CA, Pardalos PM (1993) New results in the packing of equal circles in a square. Dept Chemical Engin, Princeton Univ, Princeton
16. Murtagh BA, Saunders MA (1987) MINOS 5.0, user guide. Systems Optim Lab, Dept Oper Res, Stanford Univ, Stanford
17. ILOG SA (1997) ILOG solver 4.0 user's manual; ILOG solver 4.0 reference manual. ILOG, Mountain View
18. Schrage L (1998) Optimization modeling with LINGO. Lindo Systems, Chicago, www.lindo.com
19. Smolka G (1995) The Oz programming model. In: van Leeuwen J (ed) Computer Sci Today, 1000 of Computer Sci. Springer, Berlin, pp 324–343
20. Sosic R, Gu J (1990) A polynomial time algorithm for the n-queens problem. SIGART Bull 1(3):7–11
21. Sosic R, Gu J (1991) 3,000,000 queens in less than one minute. SIGART Bull 2(1):22–24

Molecular Distance Geometry Problem

Carlile Lavor[1], Leo Liberti[2], Nelson Maculan[3]
[1] State University of Campinas (IMECC-UNICAMP), Campinas, Brazil
[2] École Polytechnique, LIX, Palaiseau, France
[3] Federal University of Rio de Janeiro (COPPE-UFRJ), Rio de Janeiro, Brazil

MSC2000: 46N60

Article Outline

Introduction

This article presents a general overview of some of the most recent approaches for solving the molecular distance geometry problem, namely, the ABBIE algorithm, the Global Continuation Algorithm, d.c. optimization algorithms, the geometric build-up algorithm, and the BP algorithm.

The determination of the three-dimensional structure of a molecule, especially in the protein folding framework, is one of the most important problems in computational biology. That structure is very important because it is associated to the chemical and biological properties of the molecule [7,11,46]. Basically, this problem can be tackled in two ways: experimentally, via nuclear magnetic resonance (NMR) spectroscopy and X-ray crystallography [8], or theoretically, through potential energy minimization [19].

The Molecular Distance Geometry Problem (MDGP) arises in NMR analysis. This experimental technique provides a set of inter-atomic distances d_{ij} for certain pairs of atoms (i,j) of a given protein [23,24,33,56,57]. The MDGP can be formulated as follows:

Given a set S of atom pairs (i,j) on a set of m atoms and distances d_{ij} defined over S, find positions $x_1, \ldots, x_m \in \mathbb{R}^3$ of the atoms in the molecule such that

$$||x_i - x_j|| = d_{ij}\,, \quad \forall (i,j) \in S. \tag{1}$$

When the distances between all pairs of atoms of a molecule are given, a unique three-dimensional structure can be determined by a linear time algorithm [16]. However, because of errors in the given distances, a solution may not exist or may not be unique. In addition to this, because of the large scale of problems that arise in practice, the MDGP becomes very hard to solve in general. Saxe [51] showed that the MDGP is NP-complete even in one spatial dimension.

The exact MDGP can be naturally formulated as a nonlinear global minimization problem, where the objective function is given by

$$f(x_1, \ldots, x_m) = \sum_{(i,j)\in S} (||x_i - x_j||^2 - d_{ij}^2)^2\,. \tag{2}$$

This function is everywhere infinitely differentiable and has an exponential number of local minimizers. Assuming that all the distances are correctly given, $x \in \mathbb{R}^{3m}$ solves the problem if and only if $f(x) = 0$.

Formulations (1) and (2) correspond to the exact MDGP. Since experimental errors may prevent solution existence (e. g. when the triangle inequality

$$d_{ij} \le d_{ik} + d_{kj}$$

is violated for atoms i, j, k), we sometimes consider an ϵ-optimum solution of (1), i. e. a solution $x_1, \ldots, x_m$ satisfying

$$\left| ||x_i - x_j|| - d_{ij} \right| \le \epsilon\,, \quad \forall (i,j) \in S\,. \tag{3}$$

Moré and Wu [41] showed that even obtaining such an ϵ-optimum solution is NP-hard for ϵ small enough.

In practice, it is often just possible to obtain lower and upper bounds on the distances [4]. Hence a more practical definition of the MDGP is to find positions $x_1, \ldots, x_m \in \mathbb{R}^3$ such that

$$l_{ij} \le ||x_i - x_j|| \le u_{ij}\,, \quad \forall (i,j) \in S\,, \tag{4}$$

where l_{ij} and u_{ij} are lower and upper bounds on the distance constraints, respectively.

The MDGP is a particular case of a more general problem, called the distance geometry problem [6,13,14,15], which is intimately related to the Euclidean distance matrix completion problem [1,28,38].

Several methods have been developed to solve the MDGP, including the EMBED algorithm by Crippen and Havel [12,25], the alternating projection algorithm by Glunt et al. [20], spectrial gradient methods by Glunt et al. [21,22], the multi-scaling algorithm by Trosset et al. [29,52], a stochastic/perturbation algorithm by Zou, Byrd, and Schnabel [58], variable neighborhood search-based algorithms by Liberti, Lavor, and Maculan [35,39], the ABBIE algorithm by Hendrickson [26,27], the Global Continuation Algorithm by Moré and Wu [41,42,43,44,45], the d.c. optimization algorithms by An and Tao [2,3], the geometric build-up algorithm by Dong, Wu, and Wu [16,17,54], and the BP algorithm by Lavor, Liberti, and Maculan [37]. Two completely different approaches for solving the MDGP are given in [34] (based on quantum computation) and [53] (based on algebraic geometry).

The wireless network sensor positioning problem is closely related to the MDGP, the main difference being the presence of fixed anchor points with known positions: results derived for this problem can often be applied to the MDGP. Amongst the most notable, [18] shows that the MDGP associated to a trilateration graph (a graph with an order on the vertices such that each vertex is adjacent to the preceding 4 vertices) can be solved in polynomial time; [40] provides a detailed study of Semi Definite Programming (SDP) relaxations applied to distance geometry problems.

ABBIE Algorithm

In [26,27], Hendrickson describes an approach to the exact MDGP that replaces a large optimization problem, given by (2), by a sequence of smaller ones. He exploits some combinatorial structure inherent in the MDGP, which allows him to develop a divide-and-conquer algorithm based on a graph-theoretic viewpoint.

If the atoms and the distances are considered as nodes and edges of a graph, respectively, the MDGP can be described by a distance graph and the solution to the problem is an embedding of the distance graph in an Euclidean space. When some of the atoms can be moved without violating any distance constraints, there may be many embeddings. The graph is then called flexible or otherwise rigid.

If the graph is rigid or does not have partial reflections, for example, then the graph has a unique embedding. These necessary conditions can be used to find subgraphs that have unique embeddings. The problem can then be solved by decomposing the graph into such subgraphs, in which the minimization problems associated to the function (2) are solved. The solutions found for the subgraphs can then be combined into a solution for the whole graph.

This approach to the MDGP has been implemented in a code named ABBIE and tested on simulated data provided by the bovine pancreatic ribonuclease A, a typical small protein consisting of 124 amino acids, whose three-dimensional structure is known [47]. The data set consists of all distances between pairs of atoms in the same amino acid, along with 1167 additional distances corresponding to pairs of hydrogen atoms that were within 3.5 Å of each other. It was used fragments of the protein consisting of the first $20, 40, 60, 80$ and 100 amino acids as well as the full protein, with two sets of distance constraints for each size corresponding to the largest unique subgraphs and the reduced graphs. These problems have from 63 up to 777 atoms.

Global Continuation Algorithm

In [43], Moré and Wu formulated the exact MDGP in terms of finding the global minimum of a similar function to (2),

$$f(x_1, \dots, x_m) = \sum_{(i,j)\in S} w_{ij}(||x_i - x_j||^2 - d_{ij}^2)^2 , \quad (5)$$

where w_{ij} are positive weights (in numerical results $w_{ij} = 1$ was used).

Following the ideas described in [55], Moré and Wu proposed an algorithm, called Global Continuation Algorithm, based on a continuation approach for global optimization. The idea is gradually transform the function (5) into a smoother function with fewer local minimizers, where an optimization algorithm is then applied to the transformed function, tracing their minimizers back to the original function. For other works based on continuation approach, see [9,10,30,31,32,49].

The transformed function $\langle f\rangle_\lambda$, called the Gaussian transform, of a function $f\colon \mathbb{R}^n \to \mathbb{R}$ is defined by

$$\langle f\rangle_\lambda(x) = \frac{1}{\pi^{n/2}\lambda^n}\int_{\mathbb{R}^n} f(y)\exp\left(-\frac{||y-x||^2}{\lambda^2}\right)\mathrm{d}y\,, \tag{6}$$

where the parameter λ controls the degree of smoothing. The value $\langle f\rangle_\lambda(x)$ is a weighted average of $f(x)$ in a neighborhood of x, where the size of the neighborhood decreases as λ decreases: as $\lambda \to 0$, the average is carried out on the singleton set $\{x\}$, thus recovering the original function in the limit. Smoother functions are obtained as λ increases.

This approach to the MDGP has been implemented and tested on two artificial models of problems, where the molecule has $m = s^3$ atoms located in the three-dimensional lattice

$$\{(i_1, i_2, i_3)\colon 0 \le i_1 < s, 0 \le i_2 < s, 0 \le i_3 < s\}$$

for an integer $s \ge 1$. In numerical results, it was considered $m = 27, 64, 125, 216$.

In the first model, the ordering for the atoms is specified by letting i be the atom at the position (i_1,i_2,i_3),

$$i = 1 + i_1 + s i_2 + s^2 i_3,$$

and the set of atom pairs whose distances are known, S, is given by

$$S = \{(i,j) : |i-j| \le r\}\,, \tag{7}$$

where $r = s^2$. In the second model, the set S is specified by

$$S = \{(i,j) : ||x_i - x_j|| \le \sqrt{r}\}\,, \tag{8}$$

where $x_i = (i_1, i_2, i_3)$ and $r = s^2$. For both models, s is considered in the interval $3 \le s \le 6$.

In (7), S includes all nearby atoms, while in (8), S includes some of nearby atoms and some relatively distant atoms.

It was shown that the Global Continuation Algorithm usually finds a solution from any given starting point, whereas the local minimization algorithm used in the multistart methods is unreliable as a method for determining global solutions. It was also showed that the continuation approach determines a global solution with less computational effort that is required by the multistart approach.

D.C. Optimization Algorithms

In [2,3], An and Tao proposed an approach for solving the exact MDGP, based on the d.c. (difference of convex functions) optimization algorithms. They worked in $\mathcal{M}_{m,3}(\mathbb{R})$, the space of real matrices of order $m \times 3$, where for $X \in \mathcal{M}_{m,3}(\mathbb{R})$, X_i (resp., X^i) is its ith row (resp., ith column). By identifying a set of positions of atoms $x_1, \dots, x_m$ with the matrix X, $X_i^T = x_i$ for $i = 1, \dots, m$, they expressed the MDGP by

$$0 = \min\left\{\sigma(X) := \frac{1}{2}\sum_{(i,j)\in S, i<j} w_{ij}\theta_{ij}(X)\colon X \in \mathcal{M}_{m,3}(\mathbb{R})\right\}\,, \tag{9}$$

where $w_{ij} > 0$ for $i \ne j$ and $w_{ii} = 0$ for all i. The pairwise potential $\theta_{ij}\colon \mathcal{M}_{m,3}(\mathbb{R}) \to \mathbb{R}$ is defined for problem (1) by either

$$\theta_{ij}(X) = \left(d_{ij}^2 - ||X_i^T - X_j^T||^2\right)^2 \tag{10}$$

or

$$\theta_{ij}(X) = \left(d_{ij} - ||X_i^T - X_j^T||\right)^2\,, \tag{11}$$

and for problem (4) by

$$\theta_{ij}(X) = \min{}^2\left\{\frac{||X_i^T - X_j^T||^2 - l_{ij}^2}{l_{ij}^2}, 0\right\} + \max{}^2\left\{\frac{||X_i^T - X_j^T||^2 - u_{ij}^2}{u_{ij}^2}, 0\right\}\,. \tag{12}$$

Similarly to (2), X is a solution if and only if it is a global minimizer of problem (9) and $\sigma(X) = 0$.

While the problem (9) with θ_{ij} given by (9) or (12) is a nondifferentiable optimization problem, it is a d.c. optimization problem.

An and Tao demonstrated that the d.c. algorithms can be adapted for developing efficient algorithms for solving large-scale exact MDGPs. They proposed various versions of d.c. algorithms that are based on different formulations for the problem. Due its local character, the global optimality cannot be guaranteed for a general d.c. problem. However, the fact that the global optimality can be obtained with a suitable starting point motivated them to investigate a technique for computing good starting points for the d.c. algorithms in the solution of (9), with θ_{ij} defined by (11).

The algorithms have been tested on three sets of data: the artificial data from Moré and Wu [43] (with up to 4096 atoms), 16 proteins in the PDB [5] (from 146 up to 4189 atoms), and the data from Hendrickson [27] (from 63 up to 777 atoms). Using these data, they showed that the d.c. algorithms can efficiently solve large-scale exact MDGPs.

Geometric Build-up Algorithm

In [17], Dong and Wu proposed the solution of the exact MDGP by an algorithm, called the geometric build-up algorithm, based on a geometric relationship between coordinates and distances associated to the atoms of a molecule. It is assumed that it is possible to determine the coordinates of at least four atoms, which are marked as fixed; the remaining ones are non-fixed. The coordinates of a non-fixed atom a can be calculated by using the coordinates of four non-coplanar fixed atoms such that the distances between any of these four atoms and the atom a are known. If such four atoms are found, the atom a changes its status to fixed. More specifically, let b_1, b_2, b_3, b_4 be the four fixed atoms whose Cartesian coordinates are already known. Now suppose that the Euclidean distances among the atom a and the atoms b_1, b_2, b_3, b_4, namely $d_{a,bi}$, for $i = 1, 2, 3, 4$, are known. That is,

$$\begin{aligned}
||a - b_1|| &= d_{a,b_1}, \\
||a - b_2|| &= d_{a,b_2}, \\
||a - b_3|| &= d_{a,b_3}, \\
||a - b_4|| &= d_{a,b_4}.
\end{aligned}$$

Squaring both sides of these equations, we have:

$$\begin{aligned}
||a||^2 - 2a^T b_1 + ||b_1||^2 &= d_{a,b_1}^2, \\
||a||^2 - 2a^T b_2 + ||b_2||^2 &= d_{a,b_2}^2, \\
||a||^2 - 2a^T b_3 + ||b_3||^2 &= d_{a,b_3}^2, \\
||a||^2 - 2a^T b_4 + ||b_4||^2 &= d_{a,b_4}^2.
\end{aligned}$$

By subtracting one of these equations from the others, it is obtained a linear system that can be used to determine the coordinates of the atom a. For example, subtracting the first equation from the others, we obtain

$$Ax = b\,, \tag{13}$$

where

$$A = -2 \begin{bmatrix} (b_1 - b_2)^T \\ (b_1 - b_3)^T \\ (b_1 - b_3)^T \end{bmatrix}, \quad x = a\,,$$

and

$$b = \begin{bmatrix} \left(d_{a,b_1}^2 - d_{a,b_2}^2\right) - \left(||b_1||^2 - ||b_2||^2\right) \\ \left(d_{a,b_1}^2 - d_{a,b_3}^2\right) - \left(||b_1||^2 - ||b_3||^2\right) \\ \left(d_{a,b_1}^2 - d_{a,b_4}^2\right) - \left(||b_1||^2 - ||b_4||^2\right) \end{bmatrix}.$$

Since b_1, b_2, b_3, b_4 are non-coplanar atoms, the system (13) has a unique solution. If the exact distances between all pairs of atoms are given, this approach can determine the coordinates of all atoms of the molecule in linear time [16].

Dong and Wu implemented such an algorithm, but they verified that it is very sensitive to the numerical errors introduced in calculating the coordinates of the atoms. In [54], Wu and Wu proposed the updated geometric build-up algorithm showing that, in this algorithm, the accumulation of the errors in calculating the coordinates of the atoms can be controlled and prevented. They have been tested the algorithm with a set of problems generated using the known structures of 10 proteins downloaded from the PDB data bank [5], with problems from 404 up to 4201 atoms.

BP Algorithm

In [37], Lavor, Liberti, and Maculan propose an algorithm, called branch-and-prune (BP), based on a discrete formulation of the exact MDGP. They observe that the particular structures of proteins makes it possible to formulate the MDGP applied to protein backbones as a discrete search problem. They formalize this by introducing the discretizable molecular distance geometry problem (DMDGP), which consists of a certain subset of MDGP instances (to which most protein backbones belong) for which a discrete formulation can be supplied. This approach requires that the bond lengths and angles, as well as the distances between atoms separated by three consecutive bond lengths are known.

In order to describe a backbone of a protein with m atoms, in addition to the bond lengths $d_{i-1,i}$, for $i = 2, \ldots, m$, and the bond angles $\theta_{i-2,i}$, for $i = 3, \ldots, m$, it is necessary to consider the torsion angles $\omega_{i-3,i}$, for $i = 4, \ldots, m$, which are the angles between the normals through the planes defined by the atoms $i-3$, $i-2$, $i-1$ and $i-2$, $i-1$, i.

It is known that [48], given all the bond lengths $d_{1,2}, \ldots, d_{m-1,m}$, bond angles $\theta_{13}, \ldots, \theta_{m-2,m}$, and torsion angles $\omega_{1,4}, \ldots, \omega_{m-3,m}$ of a molecule with m atoms, the Cartesian coordinates $(x_{i_1}, x_{i_2}, x_{i_3})$ for each atom i in the molecule can be obtained using the following formulae:

$$\begin{bmatrix} x_{i_1} \\ x_{i_2} \\ x_{i_3} \\ 1 \end{bmatrix} = B_1 B_2 \ldots B_i \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \forall i = 1, \ldots, m,$$

where

$$B_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$B_2 = \begin{bmatrix} -1 & 0 & 0 & -d_{1,2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$B_3 = \begin{bmatrix} -\cos\theta_{1,3} & -\sin\theta_{1,3} & 0 & -d_{2,3}\cos\theta_{1,3} \\ \sin\theta_{1,3} & -\cos\theta_{1,3} & 0 & d_{2,3}\sin\theta_{1,3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

and

$$B_i = \begin{bmatrix} -\cos\theta_{i-2,i} & -\sin\theta_{i-2,i} & 0 & -d_{i-1,i}\cos\theta_{i-2,i} \\ \sin\theta_{i-2,i}\cos\omega_{i-3,i} & -\cos\theta_{i-2,i}\cos\omega_{i-3,i} & -\sin\omega_{i-3,i} & d_{i-1,i}\sin\theta_{i-2,i}\cos\omega_{i-3,i} \\ \sin\theta_{i-2,i}\sin\omega_{i-3,i} & -\cos\theta_{i-2,i}\sin\omega_{i-3,i} & \cos\omega_{i-3,i} & d_{i-1,i}\sin\theta_{i-2,i}\sin\omega_{i-3,i} \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

for $i = 4, \ldots, m$.

Since all the bond lengths and bond angles are assumed to be given in the instance, the Cartesian coordinates of all atoms of a molecule can be completely determined by using the values of $\cos\omega_{i-3,i}$ and $\sin\omega_{i-3,i}$, for $i = 4, \ldots, m$.

For instances of the DMDGP class, for all $i = 4, \ldots, m$, the value of $\cos\omega_{i-3,i}$ can be computed by the formula

$$\begin{aligned} \cos\omega_{i-3,i} &= a/b \\ \text{where}\quad a &= d^2_{i-3,i-2} + d^2_{i-2,i} - 2d_{i-3,i-2}d_{i-2,i} \\ &\quad \cdot \cos\theta_{i-2,i}\cos\theta_{i-1,i+1} - d^2_{i-3,i} \\ \text{and}\quad b &= 2d_{i-3,i-2}d_{i-2,i}\sin\theta_{i-2,i}\sin\theta_{i-1,i+1}, \end{aligned} \tag{14}$$

which is just a rearrangement of the cosine law for torsion angles [50] (p. 278), and all the values in the expression (14) are given in the instance. This allows to express the position of the i-th atom in terms of the preceding three, giving 2^{m-3} possible conformations, which characterizes the discretization of the problem.

The idea of the BP algorithm is that at each step the ith atom can be placed in two possible positions. However, either of both of these positions may be infeasible with respect to some constraints. The search is branched on all atomic positions which are feasible with respect to all constraints; by contrast, if a position is not feasible the search scope is pruned.

The algorithm has been tested on the artificial data from Moré and Wu [43] (with up to 216 atoms) and on the artificial data from Lavor [36] (a selection from 10 up to 100 atoms).

Conclusion

This paper surveys some of the methods to solve the Molecular Distance Geometry Problem, with particular reference to five existing algorithms: ABBIE algorithm, global continuation algorithm, d.c. optimization algorithms, the geometric build-up algorithm and the BP algorithm.

Acknowledgements

The authors would like to thank CNPq, FAPESP and FAPERJ for their financial support.

References

1. Alfakih AY, Khandani A, Wolkowicz H (1999) Solving Euclidean distance matrix completion problems via semidefinite programming. Comput Optim Appl 12:13–30
2. An LTH (2003) Solving large-scale molecular distance geometry problems by a smoothing technique via the Gaussian transform and d.c. programming. J Global Optim 27:375–397

3. An LTH, Tao PD (2003) Large-scale molecular optimization from distance matrices by a d.c. optimization approach. SIAM J Optim 14:77–114
4. Berger B, Kleinberg J, Leighton T (1999) Reconstructing a three-dimensional model with arbitrary errors. J ACM 46:212–235
5. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE (2000) The protein data bank. Nucl Acids Res 28:235–242
6. Blumenthal LM (1953) Theory and Applications of Distance Geometry. Oxford University Press, London
7. Brooks III CL, Karplus M, Pettitt BM (1988) Proteins: a theoretical perspective of dynamics, structure, and thermodynamics. Wiley, New York
8. Brünger AT, Nilges M (1993) Computational challenges for macromolecular structure determination by X-ray crystallography and solution NMR-spectroscopy. Q Rev Biophys 26:49–125
9. Coleman TF, Shalloway D, Wu Z (1993) Isotropic effective energy simulated annealing searches for low energy molecular cluster states. Comput Optim Appl 2:145–170
10. Coleman TF, Shalloway D, Wu Z (1994) A parallel build-up algorithm for global energy minimizations of molecular clusters using effective energy simulated annealing. J Global Optim 4:171–185
11. Creighton TE (1993) Proteins: structures and molecular properties. Freeman and Company, New York
12. Crippen GM, Havel TF (1988) Distance geometry and molecular conformation. Wiley, New York
13. Dattorro J (2005) Convex optimization and euclidean distance geometry. Meboo Publishing USA, Palo Alto
14. De Leeuw J (1977) Applications of convex analysis to multidimensional scaling. In: Barra JR, Brodeau F, Romier G, van Cutsem B (eds) Recent developments in statistics. North-Holland, Amsterdam, pp 133–145
15. De Leeuw J (1988) Convergence of the majorization method for multidimensional scaling. J Classif 5:163–180
16. Dong Q, Wu Z (2002) A linear-time algorithm for solving the molecular distance geometry problem with exact inter-atomic distances. J Global Optim 22:365–375
17. Dong Q, Wu Z (2003) A geometric build-up algorithm for solving the molecular distance geometry problem with sparse distance data. J Global Optim 26:321–333
18. Eren T, Goldenberg DK, Whiteley W, Yang YR, Morse AS, Anderson BDO, Belhumeur PN (2004) Rigidity, computation, and randomization in network localization. In: Proc IEEE Infocom 2673–2684, Hong Kong
19. Floudas CA, Pardalos PM (eds)(2000) Optimization in computational chemistry and molecular biology. Nonconvex optimization and its applications, vol 40. Kluwer, The Netherlands
20. Glunt W, Hayden TL, Hong S, Wells J (1990) An alternating projection algorithm for computing the nearest euclidean distance matrix. SIAM J Matrix Anal Appl 11:589–600
21. Glunt W, Hayden TL, Raydan M (1993) Molecular conformations from distance matrices. J Comput Chem 14:114–120
22. Glunt W, Hayden TL, Raydan M (1994) Preconditioners for distance matrix algorithms. J Comput Chem 15:227–232
23. Gunther H (1995) NMR Spectroscopy: basic principles, concepts, and applications in chemistry. Wiley, New York
24. Havel TF (1991) An evaluation of computational strategies for use in the determination of protein structure from distance geometry constraints obtained by nuclear magnetic resonance. Prog Biophys Mol Biol 56:43–78
25. Havel TF (1995) Distance geometry. In: Grant DM, Harris RK (eds) Encyclopedia of nuclear magnetic resonance. Wiley, New York, pp 1701–1710
26. Hendrickson BA (1991) The molecule problem: determining conformation from pairwise distances. Ph.D. thesis. Cornell University, Ithaca
27. Hendrickson BA (1995) The molecule problem: exploiting structure in global optimization. SIAM J Optim 5:835–857
28. Huang HX, Liang ZA (2003) Pardalos PM Some properties for the euclidean distance matrix and positive semidefinite matrix completion problems. J Global Optim 25:3–21
29. Kearsley AJ, Tapia RA, Trosset MW (1998) The solution of the metric STRESS and SSTRESS problems in multidimensional scaling by Newton's method. Comput Stat 13:369–396
30. Kostrowicki J, Piela L (1991) Diffusion equation method of global minimization: performance for standard functions. J Optim Theor Appl 69:269–284
31. Kostrowicki J, Piela L, Cherayil BJ, Scheraga HA (1991) Performance of the diffusion equation method in searches for optimum structures of clusters of Lennard-Jones atoms. J Phys Chem 95:4113–4119
32. Kostrowicki J, Scheraga HA (1992) Application of the diffusion equation method for global optimization to oligopeptides. J Phys Chem 96:7442–7449
33. Kuntz ID, Thomason JF, Oshiro CM (1993) Distance geometry. In: Oppenheimer NJ, James TL (eds) Methods in Enzymology, vol 177. Academic Press, New York, pp 159–204
34. Lavor C, Liberti L, Maculan N (2005) Grover's algorithm applied to the molecular distance geometry problem. In: Proc. of VII Brazilian Congress of Neural Networks, Natal, Brazil
35. Lavor C, Liberti L, Maculan N (2006) Computational experience with the molecular distance geometry problem. In: Pintér J (ed) Global optimization: scientific and engineering case studies. Springer, New York, pp 213–225
36. Lavor C (2006) On generating instances for the molecular distance geometry problem. In: Liberti L, Maculan N (eds) Global optimization: from theory to implementation. Springer, Berlin, pp 405–414
37. Lavor C, Liberti L, Maculan N (2006) The discretizable molecular distance geometry problem. arXiv:q-bio/0608012

38. Laurent M (1997) Cuts, matrix completions and a graph rigidity. Math Program 79:255–283
39. Liberti L, Lavor C, Maculan N (2005) Double VNS for the molecular distance geometry problem. In: Proc. of MECVNS Conference, Puerto de la Cruz, Spain
40. Man-Cho So A, Ye Y (2007) Theory of semidefinite programming for sensor network localization. Math Program 109:367–384
41. Moré JJ, Wu Z (1996) ϵ-Optimal solutions to distance geometry problems via global continuation. In: Pardalos PM, Shalloway D, Xue G (eds) Global minimization of non-convex energy functions: molecular conformation and protein folding. American Mathematical Society, Providence, IR, pp 151–168
42. Moré JJ, Wu Z (1996) Smoothing techniques for macromolecular global optimization. In: Di Pillo G, Gianessi F (eds) Nonlinear Optimization and Applications. Plenum Press, New York, pp 297–312
43. Moré JJ, Wu Z (1997) Global continuation for distance geometry problems. SIAM J Optim 7:814–836
44. Moré JJ, Wu Z (1997) Issues in large scale global molecular optimization. In: Biegler L, Coleman T, Conn A, Santosa F (eds) Large scale optimization with applications. Springer, Berlin, pp 99–122
45. Moré JJ, Wu Z (1999) Distance geometry optimization for protein structures. J Global Optim 15:219–234
46. Neumaier A (1997) Molecular modeling of proteins and mathematical prediction of protein structure. SIAM Rev 39:407–460
47. Palmer KA, Scheraga HA (1992) Standard-geometry chains fitted to X-ray derived structures: validation of the rigid-geometry approximation. II. Systematic searches for short loops in proteins: applications to bovine pancreatic ribonuclease A and human lysozyme. J Comput Chem 13:329–350
48. Phillips AT, Rosen JB, Walke VH (1996) Molecular structure determination by convex underestimation of local energy minima. In: Pardalos PM, Shalloway D, Xue G (eds) Global minimization of non-convex energy functions: molecular conformation and protein folding. American Mathematical Society, Providence, IR, pp 181–198
49. Piela L, Kostrowicki J, Scheraga HA (1989) The multiple-minima problem in the conformational analysis of molecules: deformation of the protein energy hypersurface by the diffusion equation method. J Phys Chem 93:3339–3346
50. Pogorelov A (1987) Geometry. Mir Publishers, Moscow
51. Saxe JB (1979) Embeddability of weighted graphs in k-space is strongly NP-hard. In: Proc. of 17th Allerton Conference in Communications, Control, and Computing, 480–489, Allerton, USA
52. Trosset M (1998) Applications of multidimensional scaling to molecular conformation. Comput Sci Stat 29:148–152
53. Wang L, Mettu RR, Donald BR (2005) An algebraic geometry approach to protein structure determination from NMR data. In: Proc. of the 2005 IEEE Computational Systems Bioinformatics Conference, Stanford, USA
54. Wu D, Wu Z (2007) An updated geometric build-up algorithm for solving the molecular distance geometry problem with sparse distance data. J Global Optim 37:661–673
55. Wu Z (1996) The effective energy transformation scheme as a special continuation approach to global optimization with application to molecular conformation. SIAM J Optim 6:748–768
56. Wütrich K (1989) The development of nuclear magnetic resonance spectroscopy as a technique for protein structure determination. Acc Chem Res 22:36–44
57. Wütrich K (1989) Protein structure determination in solution by nuclear magnetic resonance spectroscopy. Science 243:45–50
58. Zou Z, Byrd RH, Schnabel RB (1997) A stochastic/perturbation global optimization algorithm for distance geometry problems. J Global Optim 11:91–105

Molecular Structure Determination: Convex Global Underestimation

ANDREW T. PHILLIPS
Computer Sci. Department, University Wisconsin–Eau Claire, Eau Claire, USA

MSC2000: 65K05, 90C26

Article Outline

Keywords

Protein folding; Molecular structure determination; Convex global underestimation

An important class of difficult global minimization problems arise as an essential feature of molecular structure calculations. The determination of a stable molecular structure can often be formulated in terms of calculating the global (or approximate global) minimum of a potential energy function (see [6]). Computing the global minimum of this function is very difficult because it typically has a very large number of local minima which may grow exponentially with molecule size.

One such application is the well known protein folding problem. It is widely accepted that the folded state of a protein is completely dependent on the one-dimensional linear sequence (i. e., 'primary' sequence) of amino acids from which the protein is constructed: external factors, such as enzymes, present at the time of folding have no effect on the final, or native, state of the protein. This led to the formulation of the *protein folding problem*: given a known primary sequence of amino acids, what would be its native, or folded, state in three-dimensional space.

Several successful predictions of folded protein structures have been made and announced before the experimental structures were known (see [3,9]). While most of these have been made with a blend of a human expert's abilities and computer assistance, fully automated methods have shown promise for producing previously unattainable accuracy [2].

These machine based prediction strategies attempt to lessen the reliance on experts by developing a completely computational method. Such approaches are generally based on two assumptions. First, that there exists a potential energy function for the protein; and second that the folded state corresponds to the structure with the lowest potential energy (minimum of the potential energy function) and is thus in a state of thermodynamic equilibrium. This view is supported by in vitro observations that proteins can successfully refold from a variety of denatured states. Evolutionary theory also supports a folded state at a global energy minimum. Protein sequences have evolved under pressure to perform certain functions, which for most known occurrences requires a stable, unique, and compact structure. Unless specifically required for a certain function, there was no biochemical need for proteins to hide their global minimum behind a large kinetic energy barrier. While kinetic blocks may occur, they should be limited to special proteins developed for certain functions (see [1]).

Molecular Model

Unfortunately, finding the 'true' energy function of a molecular structure, if one even exists, is virtually impossible. For example, with proteins ranging in size up to 1, 053 amino acids (a collagen found in tendons), exhaustive conformational searches will never be tractable. Practical search strategies for the protein folding problem currently require a simplified, yet sufficiently realistic, molecular model with an associated potential energy function representing the dominant forces involved in protein folding [4]. In a one such simplified model, each residue in the primary sequence of a protein is characterized by its backbone components $NH - C_\alpha H - \mathbf{C}'O$ and one of 20 possible amino acid sidechains attached to the central C_α atom. The three-dimensional structure of the chain is determined by internal molecular coordinates consisting of bond lengths l, bond angles θ, sidechain torsion angles χ, and the backbone dihedral angles ϕ, ψ, and ω. Fortunately, these $10r - 6$ parameters (for an r-residue structure) do not all vary independently. Some of these ($7r - 4$ of them) are regarded as fixed since they are found to vary within only a very small neighborhood of an experimentally determined value. Among these are the $3r - 1$ backbone bond lengths l, the $3r - 2$ backbone bond angles θ, and the $r - 1$ peptide bond dihedral angles ω (fixed in the trans conformation). This leaves only the r sidechain torsion angles χ, and the $r - 1$ backbone dihedral angle pairs (ϕ, ψ). In the reduced representation model presented here, the sidechain angles χ are also fixed since sidechains are treated as united atoms (see below) with their respective torsion angles χ fixed at an 'average' value taken from the Brookhaven Protein Databank. Remaining are the $r - 1$ backbone dihedral angles pairs. These also are not completely independent; they are severely constrained by known chemical data (the Ramachandran plot) for each of the 20 amino acid residues. Furthermore, since the atoms from one C_α to the next C_α along the backbone can be grouped into rigid planar peptide units, there are no extra parameters required to express the three-dimensional position of the attached O and H peptide atoms. Hence, these bond lengths and bond angles are also known and fixed.

Another key element of this simplified polypeptide model is that each sidechain is classified as either hydrophobic or polar, and is represented by only a single 'virtual' center of mass atom. Since each sidechain is represented by only the single center of mass 'virtual atom' C_s, no extra parameters are needed to define the position of each sidechain with respect to the backbone mainchain. The twenty amino acids are thus classified into two groups, hydrophobic and polar, according to the scale given by S. Miyazawa and R.L. Jernigan [7].

Corresponding to this simplified polypeptide model is a simple energy function. This function includes four components: a contact energy term favoring pairwise hydrophobic residues, a second contact term favoring hydrogen bond formation between donor NH and acceptor $C' = O$ pairs, a steric repulsive term which rejects any conformation that would permit unreasonably small interatomic distances, and a main chain torsional term that allows only certain preset values for the backbone dihedral angle pairs (ϕ, ψ). Since the residues in this model come in only two forms, hydrophobic and polar, where the hydrophobic monomers exhibit a strong pairwise attraction, the lowest free energy state involves those conformations with the greatest number of hydrophobic 'contacts' [4] and intrastrand hydrogen bonds. Simplified potential functions have been successful in [10,11], and [12]. Here we use a simple modification of the energy function from [11].

The Convex Global Underestimator

One practical means for finding the global minimum of the polypeptide's potential energy function is to use a convex global underestimator to localize the search in the region of the global minimum. The idea is to fit all known local minima with a convex function which underestimates all of them, but which differs from them by the minimum possible amount in the discrete L1 norm. The minimum of this underestimator is used to predict the global minimum for the function, allowing a more localized conformer search to be performed based on the predicted minimum.

More precisely, given an r-residue structure with $n = 2r - 2$ backbone dihedral angles, denote a conformation of this simplified model by $\phi \in \mathbf{R}^n$, and the corresponding simplified potential energy function value by $F(\phi)$. Then, assuming that $k \geq 2n + 1$ local minimum conformations $\phi^{(j)}$, for $j = 1, \dots, k$, have been computed, a convex quadratic underestimating function $U(\phi)$ is fitted to these local minima so that it underestimates all the local minima, and normally interpolates $F(\phi^{(j)})$ at $2n + 1$ points. This is accomplished by determining the coefficients in the function $U(\phi)$ so that

$$\delta_j = F(\phi^{(j)}) - U(\phi^{(j)}) \geq 0 \tag{1}$$

for $j = 1, \dots, k$, and where $\sum_{j=1}^{n} \delta_j$ is minimized. That is, the difference between $F(\phi)$ and $U(\phi)$ is minimized in the discrete L_1 norm over the set of k local minima $\phi^{(j)}$, $j = 1, \dots, k$. Of course, this 'underestimator' only underestimates known local minima. The specific underestimating function $U(\phi)$ used in this convex global underestimator (CGU) method is given by

$$U(\phi) = c_0 + \sum_{i=1}^{n} \left(c_i \phi_i + \frac{1}{2} d_i \phi_i^2 \right) . \tag{2}$$

Note that c_i and d_i appear linearly in the constraints of (1) for each local minimum $\phi^{(j)}$. Convexity of this quadratic function is guaranteed by requiring that $d_i \geq 0$ for $i = 1, \dots, n$. Other linear combinations of convex functions could also be used, but this quadratic function is the simplest.

Additionally, in order to guarantee that $U(\phi)$ attains its global minimum $U_{\min}$ in the hyperrectangle $H\phi = \{\phi_i : 0 \leq \underline{\phi}_i \leq \phi_i \leq \overline{\phi}_i \leq 2\pi\}$, an additional set of constraints are imposed on the coefficients of $U(\phi)$:

$$\begin{cases} c_i + \underline{\phi}_i d_i \leq 0, \\ c_i + \overline{\phi}_i d_i \geq 0, \end{cases} \qquad i = 1, \dots, n. \tag{3}$$

Note that the satisfaction of (3) implies that $c_i \leq 0$ and $d_i \geq 0$ for $i = 1, \dots, n$.

The unknown coefficients c_i, $i = 0, \dots, n$, and d_i, $i = 1, \dots, n$, can be determined by a linear program which may be considered to be in the dual form. For reasons of efficiency, the equivalent primal of this problem is actually solved, as described below. The solution to this primal linear program provides an optimal dual vector, which immediately gives the underestimating function coefficients c_i and d_i. Since the convex quadratic function $U(\phi)$ gives a global approximation to the local minima of $F(\phi)$, then its easily computed global minimum function value $U_{\min}$ is a good candidate for an approximation to the global minimum of the correct energy function $F(\phi)$.

An efficient linear programming formulation and solution satisfying (1)–(3) will now be summarized. Let $f^{(j)} = F(\phi^{(j)})$, for $j = 1, \dots, k$, and let $f \in \mathbf{R}^k$ be the vector with elements $f^{(j)}$. Also let $\omega^{(j)} \in \mathbf{R}^n$ be the vector with elements $\frac{1}{2}(\phi_i^{(j)})^2$, $i = 1, \dots, n$, and let $e_k \in \mathbf{R}^k$ be the vector of ones. Now define the following two matrices

$\varPhi \in \mathbf{R}^{(n+1)\times k}$ and $\varOmega \in \mathbf{R}^{n\times k}$:

$$\begin{cases} \varPhi = \begin{pmatrix} e_k^\top \\ \phi^{(1)} \cdots \phi^{(k)} \end{pmatrix}, \\ \varOmega = \left(\omega^{(1)} \cdots \omega^{(k)}\right). \end{cases} \tag{4}$$

Finally, let $c \in \mathbf{R}^{n+1}$, $d \in \mathbf{R}^n$, and $\delta \in \mathbf{R}^k$ be the vectors with elements c_i, d_i, and δ_i, respectively. Then (1)–(3) can be restated as the linear program (with free variables c, d, and δ):

- minimize $e_k^\top \delta$
- such that

$$\begin{pmatrix} \varPhi^\top & \varOmega^\top & 0 \\ -\varPhi^\top & -\varOmega^\top & -I_k \\ I'_n & \underline{D} & 0 \\ -I'_n & -\overline{D} & 0 \end{pmatrix} \begin{pmatrix} c \\ d \\ \delta \end{pmatrix} \leq \begin{pmatrix} f \\ -f \\ 0 \\ 0 \end{pmatrix}, \tag{5}$$

where $\underline{D} = \operatorname{diag}(\underline{\phi}_1, \dots, \underline{\phi}_n)$, $\overline{D} = \operatorname{diag}(\overline{\phi}_1, \dots, \overline{\phi}_n)$, I_k is the identity matrix of order k, and I'_n is the $n \times (n+1)$ 'augmented' matrix $(0 : I_n$ where I_n is the identity matrix of order n.

Since the matrix in (5) has more rows than columns ($2(k+n)$ rows and $k + 2n + 1$ columns, where $k \geq 2n + 1$), it is computationally more efficient to consider it as a dual problem, and to solve the equivalent primal. After some simple transformations, this primal problem reduces to:

$$\begin{cases} \min & f^\top y_1 - f^\top e_k \\ \text{s.t.} & \begin{pmatrix} \varPhi & I_n'^\top & -I_n'^\top \\ \varOmega & \underline{D} & -\overline{D} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \varPhi e_k \\ \varOmega e_k \end{pmatrix} \\ & y_1, y_2, y_3 \geq 0 \end{cases} \tag{6}$$

which has only $2n + 1$ rows and $k + 2n \geq 4n + 1$ columns, and the obvious initial feasible solution $y_1 = e_k$ and $y_2 = y_3 = 0$. Furthermore, since the first of the $2n + 1$ constraints in (6) in fact requires that $e_k^\top y_1 = 1$, then the function $f^\top y_1 - f^\top e_k$ is also bounded below, and so this primal linear program always has an optimal solution. This optimal solution gives the values of c, d, and δ via the dual vectors, and also determines which values of $f^{(j)}$ are interpolated by the potential function $U(\phi)$. That is, the basic columns in the optimal solution to (6) correspond to the conformations $\phi^{(j)}$ for which $F(\phi^{(j)}) = U(\phi^{(j)})$.

Note that once an optimal solution to (6) has been obtained, the addition of new local minima is very easy. It is done by simply adding new columns to $\varPhi$ and $\varOmega$, and therefore to the constraint matrix in (6). The number of primal rows remains fixed at $2n + 1$, independent of the number k of local minima.

The convex quadratic underestimating function $U(\phi)$ determined by the values $c \in \mathbf{R}^{n+1}$ and $d \in \mathbf{R}^n$ now provides a global approximation to the local minima of $F(\phi)$, and its easily computed global minimum point $\phi_{\min}$ is given by $(\phi_{\min})_i = -c_i/d_i$, $i = 1, \dots, n$, with corresponding function value $U_{\min}$ given by $U_{\min} = c_0 - \sum_{i=1}^n c_i^2/d_i$. The value $U_{\min}$ is a good candidate for an approximation to the global minimum of the correct energy function $F(\phi)$, and so $\phi_{\min}$ can be used as an initial starting point around which additional configurations (i. e., local minima) should be generated. These local minima are added to the constraint matrix in (6) and the process is repeated. Before each iteration of this process, it is necessary to reduce the volume of the hyperrectangle $H\phi$ over which the new configurations are produced so that a tighter fit of $U(\phi)$ to the local minima 'near' $\phi_{\min}$ is constructed.

The rate and method by which the hyperrectangle size is decreased, and the number of additional local minima computed at each iteration must be determined by computational testing. But clearly the method depends most heavily on computing local minima quickly and on solving the resulting linear program efficiently to determine the approximating function $U(\phi)$ over the current hyperrectangle.

If E_c is a cutoff energy, then one means for decreasing the size of the hyperrectangle $H\phi$ at any step is to let $H\phi = \{\phi \colon U(\phi) \leq E_c\}$. To get the bounds of $H\phi$, consider $U(\phi) \leq E_c$ where $U(\phi)$ satisfies (2). Then limiting ϕ_i requires that

$$\sum_{i=1}^{n} \left(c_i \phi_i + \frac{1}{2} d_i \phi_i^2 \right) \leq E_c - c_0. \tag{7}$$

As before, the minimum value of $U(\phi)$ is attained when $\phi_i = -c_i/d_i$, $i = 1, \dots, n$. Assigning this minimum value to each ϕ_i, except ϕ_k, then results in

$$c_k \phi_k + \frac{1}{2} d_k \phi_k^2 \leq E_c - c_0 + \frac{1}{2} \sum_{i \neq k} \frac{c_i^2}{d_i} \equiv \beta_k. \tag{8}$$

The lower and upper bounds on ϕ_k, $k = 1, \dots, n$, are given by the roots of the quadratic equation

$$c_k\phi_k + \frac{1}{2}d_k\phi_k^2 = \beta_k. \qquad (9)$$

Hence, these bounds can be used to define the new hyperrectangle $H\phi$ in which to generate new configurations.

Clearly, if E_c is reduced, the size of $H\phi$ is also reduced. At every iteration the predicted global minimum value $U_{\min}$ satisfies $U_{\min} \leq F(\phi^*)$, where ϕ^* is the smallest known local minimum conformation. Therefore, $E_c = F(\phi^*)$ is often a good choice. If at least one improved point ϕ, with $F(\phi) < F(\phi^*)$, is obtained in each iteration, then the search domain $H\phi$ will strictly decrease at each iteration, and may decrease substantially in some iterations.

The CGU Algorithm

Based on the preceding description, a general method for computing the global, or near global, energy minimum of the potential energy function $F(\phi)$ can now be described.

1) Compute $k \geq 2n + 1$ distinct local minima $\phi^{(j)}$, for $j = 1, \dots, k$, of the function $F(\phi)$.
2) Compute the convex quadratic underestimator function given in (2) by solving the linear program given in (6). The optimal solution to this linear program gives the values of c and d via the dual vectors.
3) Compute the predicted global minimum point $\phi_{\min}$ given by $(\phi_{\min})_i = -c_i/d_i$, $i = 1, \dots, n$, with corresponding function value $U_{\min}$ given by $U_{\min} = c_0 - \sum_{i=1}^{n} c_i^2/(2d_i)$.
4) If $\phi_{\min} = \phi^*$, where $\phi^* = \operatorname{argmin}\{F(\phi^{(j)})\colon j = 1, 2, \dots\}$ is the best local minimum found so far, then stop and report ϕ^* as the approximate global minimum conformation.
5) Reduce the volume of the hyperrectangle $H\phi$ over which the new configurations will be produced, and remove all columns from Φ and Ω which correspond to the conformations which are excluded from $H\phi$.
6) Use $\phi_{\min}$ as an initial starting point around which additional local minima $\phi^{(j)}$ of $F(\phi)$ (restricted to the hyperrectangle $H\phi$) are generated. Add these new local minimum conformations as columns to the matrices Φ and Ω.
7) Return to step 2.

The number of new local minima to be generated in step 6 is unspecified since there is currently no theory to guide this choice. In general, a value exceeding $2n + 1$ would be required for the construction of another convex quadratic underestimator in the next iteration (step 2). In addition, the means by which the volume of the hyperrectangle $H\phi$ is reduced in step 5 may vary. One could use the two roots of (7) to define the new bounds of $H\phi$. Another method would be simply to use $H\phi = \{\phi_i\colon (\phi_{\min})_i - \delta_i \leq \phi_i \leq (\phi_{\min})_i + \delta_i\}$ where $\delta_i = |(\phi_{\min})_i - (\phi^*)_i|$, $i = 1, \dots, n$.

For complete details of the CGU method and its computational results, see [5,8].

See also

- ▶ Adaptive Simulated Annealing and its Application to Protein Folding
- ▶ Genetic Algorithms
- ▶ Global Optimization in Lennard–Jones and Morse Clusters
- ▶ Global Optimization in Protein Folding
- ▶ Monte-Carlo Simulated Annealing in Protein Folding
- ▶ Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach
- ▶ Packet Annealing
- ▶ Phase Problem in X-ray Crystallography: Shake and Bake Approach
- ▶ Protein Folding: Generalized-ensemble Algorithms
- ▶ Simulated Annealing
- ▶ Simulated Annealing Methods in Protein Folding

References

1. Abagyan RA (1993) Towards protein folding by global energy optimization. Federation of Europ Biochemical Soc: Lett 325:17–22
2. Androulakis IR, Maranas CD, Floudas CA (1997) Prediction of oligopeptide conformations via deterministic global optimization. J Global Optim 11:1–34
3. Benner SA, Gerloff DL (1993) Predicting the conformation of proteins: man versus machine. Federation of Europ Biochemical Soc: Lett 325:29–33
4. Dill KA (1990) Dominant forces in protein folding. Biochemistry 29(31):7133–7155

5. Dill KA, Phillips AT, Rosen JB (1997) Protein structure and energy landscape dependence on sequence using a continuous energy function. J Comput Biol 4(3):227–239
6. Merz K, Grand S Le (1994) The protein folding problem and tertiary structure prediction. Birkhäuser, Basel
7. Miyazawa S, Jernigan RL (1993) A new substitution matrix for protein sequence searches based on contact frequencies in protein structures. Protein Eng 6:267–278
8. Phillips AT, Rosen JB, Walke VH (1995) Molecular structure determination by global optimization. In: Pardalos PM, Xue GL, Shalloway D (eds) DIMACS. Amer Math Soc, Providence, pp 181–198
9. Richards FM (1991) The protein folding problem. Scientif Amer:54–63
10. Srinivasan R, Rose GD (1995) LINUS: A hierarchic procedure to predict the fold of a protein. PROTEINS: Struct Funct Genet 22:81–99
11. Sun S, Thomas PD, Dill KA (1995) A simple protein folding algorithm using binary code and secondary structure constraints. Protein Eng 8(8):769–778
12. Yue K, Dill KA (1996) Folding proteins with a simple energy function and extensive conformational searching. Protein Sci 5:254–261

Monotonic Optimization

SAED ALIZAMIR
Department of Industrial and Systems Engineering, University of Florida, Gainesville, USA

MSC2000: 90C26, 65K05, 90C30

Article Outline

Introduction

The role of convexity in optimization theory has increased significantly over the last few decades. Despite this fact, a wide variety of global optimization problems are usually encountered in applications in which nonconvex models need to be tackled. For this reason, developing solution methods for specially structured nonconvex problems has become one of the most active areas in recent years. Although these problems are difficult by their nature, promising progress is achieved for some special mathematical structures. Among the solution methods developed for these special structures, *monotonic optimization*, first proposed by Tuy [9], is presented in this study.

Problems of optimizing monotonic functions of n variables under monotonic constraints arise in the mathematical modeling of a broad range of real-world systems, including in economics and engineering. The original difficulties of these problems can be reduced by a number of principles derived from their monotonicity properties. For example, in nonconvex problems in general, a solution which is known to be feasible or even locally optimal, does not provide any information about global optimality and the search should be continued on the entire feasible space, while for an increasing objective function, a feasible solution like z, would exclude the cone $z + R^n_+$ from the search procedure (for a minimization objective function). In a similar way, if $g(x)$ in a constraint like $g(x) \leq 0$ is increasing, then by knowing that z is infeasible for this constraint, the whole cone $z + R^n_+$ can be discarded from further consideration. This kind of information would obviously restrict the search space and may result in more efficient solution methods.

To formally present the general framework of the monotonic optimization problem, consider two vectors $x, x' \in R^n$. We say $x' \geq x$ (x' dominates x) if $x'_i \geq x_i \; \forall i = 1, \ldots, n$. We say $x' > x$ (x' strictly dominates x) if $x'_i > x_i \; \forall i = 1, \ldots, n$. Let $R^n_+ = \{x \in R^n | x \geq 0\}$ and $R^n_{++} = \{x \in R^n | x > 0\}$. If $a, b \in R^n$ and $a \leq b$, we define the box $[a, b]$ as the set of all $x \in R^n$ such that $a \leq x \leq b$. Similarly, let $[a, b) = \{x | a \leq x < b\}$ and $(a, b] = \{x | a < x \leq b\}$. A function $f : R^n \to R$ is called *increasing* on a box $[a, b] \in R^n$ if $f(x) \leq f(x')$ for $a \leq x \leq x' \leq b$. A function f is called *decreasing* if $-f$ is increasing. Any increasing or decreasing function is referred to as *monotonic*. It can be easily shown that the pointwise supremum of a bounded-above family of increasing functions and the pointwise infimum of a bounded-below family of increasing functions are increasing.

In monotonic optimization, the following problem is considered:

$$\begin{aligned} \text{Maximize (minimize)} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \le 1 \quad \forall i = 1, \dots, m_1 , \\ & h_j(x) \ge 1 \quad \forall j = 1, \dots, m_2 , \\ & x \in R^n_+ , \end{aligned} \tag{1}$$

in which $f(x)$, $g_i(x)$, and $h_j(x)$ are increasing functions on R^n. A more general definition of this problem is presented in Sect. "Normal Sets and Polyblocks". Heuristically, $f(x)$ may be a cost function (profit function for the maximize problem), $g_i(x)$ may express some resource availability constraints, while $h_j(x)$ may be a family of utility functions which have to take a value at least as big as a goal.

The remainder of this article is organized as follows. We first describe the theory of normal sets and polyblocks in Sect. "Normal Sets and Polyblocks". Monotonic optimization algorithms are presented in Sect. "Solution Method". Section "Generalizations" contains two generalizations of monotonic optimization. Different class of applications for which monotonic optimization is adapted are discussed in Sect. 5 and finally conclusions are made in Sect. "Conclusions".

Normal Sets and Polyblocks

The theory of normal sets and polyblocks is the underlying principle for monotonic optimization. In this section, the definitions are presented as well as the main concepts and properties to help the reader to understand the upcoming algorithms. For more details and proofs see [5,9,10].

Normal Sets

A set $G \subset R^n_+$ is called *normal* if for any two points $x, x' \in R^n_+$ such that $x \le x' > x' \in G$ implies $x \in G$. Given any set $D \subset R^n_+$, the set $N[D]$, which is called the *normal hull* of D, is the smallest normal set containing D. In other words, $N[D]$ can be interpreted as the intersection of all normal sets that contain D. The intersection and the union of a family of normal sets are normal. If the normal set contains a point $u \in R^n_{++}$ we say it has a *nonempty interior*. Suppose that $g(x)$ is an increasing function over R^n_+. Define the *level set* of $g(x)$ as the set $G = \{x \in R^n_+ | g(x) \le 1\}$. It can be shown that the level set of an increasing function is a normal set and it is closed if the function is lower semicontinuous.

Define $I(x) = \{i | x_i = 0\}$, $K_x = \{x' \in R^n_+ | x'_i > x_i \ \forall i \notin I(x)\}$, and $\text{cl}K_x = \{x' \in R^n_+ | x' \ge x\}$. Then a point $y \in R^n_+$ is called an *upper boundary point* of a bounded normal set G if $y \in \text{cl}G$ while $K_y \subset R^n_+ \setminus G$. The set of upper boundary points of G is called the *upper boundary* of G and is denoted by $\partial^+ G$.

For a compact normal set $G \subset [0, b]$ with nonempty interior and for every point $z \in R^n_+ \setminus \{0\}$, the half line from 0 through z meets $\partial^+ G$ at a unique point denoted by $\pi_G(z)$, which is defined as $\pi_G(z) = \lambda z$, $\lambda = \max\{\alpha > 0 | \alpha z \in G\}$.

A set $H \subset R^n_+$ is called a *reverse normal set* (also known as *conormal*) if $x' \ge x$ and $x \in H$ implies $x' \in H$. A reverse normal set in a box $[0, b]$ is defined as a set like $H \in R^n_+$ for which $0 \le x \le x' \le b$ and $x \in H$ implies $x' \in H$. As before, $rN[D]$ is the smallest reverse normal set containing $D \subset R^n_+$ and is called a *reverse normal hull* of set D. Define $H = \{x \in R^n_+ | h(x) \ge 1\}$ for the increasing function $h(x)$. Then it can be shown that H is reverse normal and it is closed if $h(x)$ is upper semicontinuous.

A point $y \in R^n_+$ is called a *lower boundary point* of a reverse normal set H if $y \in \text{cl}H$ and $x \notin H \ \forall x < y$. The set of lower boundary points of H is called the *lower boundary* of H and is denoted by $\partial^- H$.

For the closed reverse normal set H and $b \in \text{int}H$ and every point $z \in [0, b] \setminus H$, the half line from b through z meets $\partial^- H$ at a unique point $\rho_H(z)$, which is defined as $\rho_H(z) = b + \mu(z - b)$, $\mu = \max\{\alpha > 0 | b + \alpha(z - b) \in H\}$.

Now consider the set of constraints imposed by increasing functions $g_i(x)$ and $h_j(x)$ in problem (1). The feasible space characterized by these sets of constraints can properly be presented by normal sets and reverse normal sets. Define the sets $G, H \subset R^n_+$ as $G = \{x \in R^n_+ | g_i(x) \le 1 \ \forall i = 1, \dots, m_1\}$ and $H = \{x \in R^n_+ | h_j(x) \ge 1 \ \forall i = 1, \dots, m_2\}$. Then by the basic properties of normal and reverse normal sets which were described above, G is the intersection of a finite number of normal sets which is normal. In a similar way, H is the intersection of a finite number of reverse normal sets which is reverse normal. Now we can redefine the fundamental problem of monotonic optimization, also called the *canonical monotonic optimization* *problem*, as optimizing a monotonic function on the intersection of a family of normal and reverse normal sets as follows:

$$\begin{aligned} \text{Maximize (minimize)} \quad & f(x) \\ \text{subject to} \quad & x \in G \cap H\,, \end{aligned} \tag{2}$$

in which $G \subset [0,\ b] \subset R^n_+$ is a compact normal set, H is a close reverse normal set, and $f(x)$ is an increasing function on $[0, b]$. Tuy [9] proved that if G has a nonempty interior (if $b \in \text{int}H$), then the maximum (minimum) of $f(x)$ over $G \cap H$, if it exists, is attained on $\partial^+ G \cap H$ ($G \cap \partial^- H$). On the basis of this essential result, it can be shown that for every arbitrary compact set $D \subset R^n_+$, $\max\{f(x)|x \in D\} = \max\{f(x)|x \in N[D]\}$. Analogously, for the minimization version of the objective function, for any arbitrary set $E \subset R^n_+$, we have $\min\{f(x)|x \in E\} = \min\{f(x)|x \in rN[E]\}$.

It is worth mentioning that the minimization problem can be converted to the maximization case by making a simple set of transformations. So it can be either transformed to the maximization problem or treated separately.

Polyblocks

The role of *polyblocks* in monotonic optimization is the same as that of the polytope in convex optimization. As the polytope is the convex hull of finitely many points in R^n, a polyblock is the normal hull of finitely many points in $R^n{}_+$. A set $P \subset R^n_+$ is a polyblock in $[a,\ b] \subset R^n_+$ if it is the union of a finite number of boxes $[a, z]$, $z \in T \subset [a, b]$. The set T is called the *vertex set* of the polyblock. We call the vertex $z \in T$ a *proper* vertex if $z \notin [0,\ z']\ \forall z' \in T \setminus \{z\}$, i. e., by removing the vetex z from T, the new polyblock created by T is not equivalent to P. A vertex which is not proper is called an *improper* vertex. A polyblock can be defined by the set of its proper vertices.

A polyblock is a closed normal set and the intersection of a set of polyblocks is again a polyblock. Now suppose that $x \in [a,\ b]$ and consider the set $P = [a,\ b] \setminus (x,\ b]$. Then it is easy to verify that P is a polyblock with vertices $z^i = b + (x - b)e^i$, $\forall i = 1, \ldots, n$ in which e^i is the ith unit vector. Using this property, we can approximate an arbitrary compact normal set $\Omega \subset R^n_+$ (with any desired accuracy) by a nested sequence of polyblock approximation. At each iteration, a point $x \notin \Omega$ is found and a new polyblock is constructed based on that which is a subset of the previous polyblock but still contains the set Ω.

To present the main idea of the *polyblock approximation method* in monotonic optimization, we need one more result on optimizing an increasing function over a polyblock. Tuy [9] proved that the increasing function $f(x)$ achieves its maximum over a polyblock at a proper vertex.

Now consider the problem of maximizing the increasing function $f(x)$ over the arbitrary compact set $\Omega \subset R^n_+$. As mentioned before, we can substitute Ω by its normal hull. So without loss of generality, we assume that Ω is normal. The idea is to construct a nested sequence of polyblock outer approximation $P_1 \supset P_2 \supset \ldots \supset \Omega$ in such a way that $\max\{f(x)|x \in P_k\} \searrow \max\{f(x)|x \in \Omega\}$.

At iteration k, assume z^k is the proper vertex of P_k which maximizes $f(x)$, i. e., $z^k = \arg\max\{f(z)|z \in T_k\}$, where T_k is the set of proper vertices of P_k. Then if z^k is feasible in Ω, the initial feasible space, it also solves the problem. Otherwise, we are interested in a new polyblock $P_{k+1} \subset P_k \setminus \{z^k\}$ which still contains Ω as a subset.

To obtain P_{k+1} from P_k, the box $[0,\ z^k]$ is replaced by $[0,\ z^k] \setminus K_{x^k}$, in which x^k is defined as $\pi(z^k)$. Mathematically, $P_{k+1} = ([0, z^k] \setminus K_{x^k}) \bigcup_{z \in T_k \setminus \{z^k\}} [0, z]$, which clearly satisfies the desired property of $\Omega \subset P_{k+1} \subset P_k \setminus \{z^k\}$.

The vertex set of the established polyblock P_{k+1}, denoted by V_{k+1}, contains the proper vertices of P_k excluding z^k and a set of n new vertices, $z^{k,1}$, $z^{k,2}, \ldots, z^{k,n}$, defined as $z^{k,i} = z^k + (x^k_i - z^k_i)e^i$. This result is directly followed by the earlier-mentioned property of polyblocks about the vertices of $[a,\ b] \setminus (x,\ b]$. Finally, the proper vertex set of P_{k+1}, T_{k+1}, is obtained from V_{k+1} by removing its improper vertices [9,10].

A set $P \subset R^n_+$ is called a *reverse polyblock* in $[0, b]$ if it is the union of a finite number of boxes $[z,\ b]$, $z \in T$, $T \subset [0,\ b]$. The set T is called the *vertex set* of the reverse polyblock. As before, z is a *proper* vertex if by removing it from T, the new reverse polyblock created by T is not equivalent to P. A reverse polyblock can be defined by the set of its proper vertices. An increasing function $f(x)$ achieves its minimum over a reverse polyblock at a proper vertex. Similar results to what we had for polyblocks can be developed for reverse polyblocks in the very same way. For more details see [9,10].

Solution Method

Consider problem (2) (in the maximization form) as discussed in Sect. "Normal Sets and Polyblocks" with the additional assumptions that $f(x)$ is semicontinuous on H and $G \cap H \subset R^n_{++}$. The latter assumption implies the existence of a vector a such that $0 < a \leq x$, $\forall x \in G \cap H$. Let $H_a = \{x \in H | x \geq a\}$. For $\epsilon \geq 0$ as a given tolerance, the solution x' is called ϵ-optimal if $f(x') \geq f(x) - \epsilon$, $\forall x \in G \cap H$. We attempt to design an algorithm which is capable of finding an ϵ-optimal solution for any given ϵ.

Obviously, $b \in H$ because otherwise the problem is infeasible. Let $P_1 = [0, b]$ be the initial polyblock and $T_1 = \{b\}$ its corresponding proper vertex set. If we apply the polyblock approximation method described in Sect. "Normal Sets and Polyblocks" to this problem, at each iteration k, P_k and its proper vertex set, T_k, are obtained from the last iteration. We should notice that every vertex $z \in T_k \backslash H_a$ can be removed since they do not belong to the initial feasible space. Also suppose that $f(x^k)$ is the best value found for the objective function so far. Then any vertex z for which $f(z) \leq f(x^k) + \epsilon$ is discarded because no ϵ-optimal solution happens to be in box $[0, z]$. These two rules can be applied at each iteration to refine the proper vertex set T_k and delete some of the vertices from further consideration.

If $T_k = \emptyset$ in some iteration k, it means there is no solution x for which $f(x) > f(x^k) + \epsilon$. So, x^k, the best solution found so far, is ϵ-optimal and the procedure terminates. Otherwise, let $z^k = \arg\max\{f(z) | z \in T_k\}$. If z^k is feasible in $G \cap H$, it solves the problem. Since $z^k \in H$ is always true, it is feasible if it belongs to G and infeasible otherwise. In the case of infeasibility, we find $x^k = \pi_G(z^k)$ and construct the polyblock P_{k+1} as described in Sect. "Normal Sets and Polyblocks" which excludes z^k while still containing a global optimal solution of the problem. This procedure is repeated until the termination criteria are satisfied or the problem is known to be infeasible. This procedure, first proposed by Tuy [9], is called the *polyblock algorithm*. Tuy [9] discussed the convergence of this method and showed that as $k \to \infty$, the sequence x^k converges to a global optimal solution of the problem.

Now consider the minimization case of problem (2) in Sect. "Normal Sets and Polyblocks" with additional assumptions that $f(x)$ is semicontinuous on G and there exists a vector c such that $0 < c < b$ and $0 \leq x \leq c$, $\forall x \in G \cap H$. A nested sequence of reverse polyblock outer approximation of $G \cap H$ (or a subset of $G \cap H$ in which the existence of at least one optimal solution is guaranteed) is called the *reverse polyblock algorithm* (*copolyblock algorithm*) which is devised to solve this problem [9].

The polyblock approximation algorithm works properly for relatively small dimension n, typically $n = 10$. However, the algorithm converges slowly as it gets closer to the global optimal solution and needs a large number of iterations even for a value of n as small as 5. Tuy et al. [12] presented two main reasons for this drawback of the algorithm. First, the speed of convergence depends on the way in which we construct the current polyblock from the previous one. Obviously, we prefer to remove a larger portion of the previous polyblock to have a smaller search space and a higher speed of convergence. This goal is achieved by employing more complex rules of constructing the polyblocks, which imposes some additional computational effort. The second source of the slowness of the algorithm is how it selects the solution x_k in each iteration. These solutions are basically derived from the monotonicity properties of the problem, while sometimes there may exist some amount of convexity which can be used to speed up the algorithm.

Tuy and Al-Khayyal [11] introduced the concept of *reduced box* and *reduced polyblock*. It involves tightening the box in which we are interested to find the upper bound of $f(x)$, in such a way that the reduced box still contains an optimal solution of the problem. Then based on that, a new procedure is developed to produce tighter polyblocks. They also redefined the proper vertex set of polyblocks in the algorithm and suggested that instead of selecting x^k as the last point of G on the halfline from a through z^k, as the original algorithm does, a more complex way can be implemented by incorporating some of the convexity properties of the problem. This is by solving the convex relaxation of the problem $\max\{f(x) | x \in G \cap H, x \in [a, z^k]\}$ which gives us an upper bound of $f(x)$ over the feasible solution x in box $[a, x^k]$. Similar ideas were applied to the reverse polyblock algorithm as well. Using these two new modifications and improvements, they developed new algorithms and discussed their convergence properties, namely, the *revised polyblock algorithm* and the *revised reverse polyblock (copolyblock) algorithm*.

Most of the outer approximation procedures, including the polyblock algorithm, encounter storage and numerical problems while solving problems in high dimensions. By using branch-and-bound strategies, one can tackle these difficulties. Bounding is performed on the basis of the polyblock approximation. As before, monotonicity cuts and convex relaxation can be combined to enhance the quality of the bounds in the corresponding portion of the feasible space. In this branch-and-bound approach, branching is performed as partitioning the feasible space into cones pairwise having no common interior point. The logic behind using conical partitioning instead of rectangular partitioning is the fact that the optimal solution of the monotonic optimization problem, as discussed before, is always achieved on the upper boundary of the feasible normal set. Using conical partitioning is more efficient and less expensive in terms of the computational time.

The algorithm starts with initial cone R^n_+ and partitions it into subcones. For each of these subcones, an upper bound for the value of the objective function over the feasible solutions contained in it is derived. Those cones which are known to not contain an optimal solution are fathomed and the remaining ones are subdivided again and the procedure is repeated until the termination criteria are satisfied. Among the remaining cones, the one having the maximal bound is the first candidate for branching. This algorithm, suggested by Tuy and Al-Khayyal [11], is called the *conical algorithm*.

For those problems having partial monotonicity and partial convexity, this branch-and-bound scheme can be extended to devise a more general method. In this method, branching is performed on the nonconvex variables and bounds are computed by Lagrangian or convex relaxation [6].

To further exploit the monotonic structure of the problem, *reduction cuts* are combined with original monotonicity cuts and a more efficient method is developed [13]. This method creates branch-and-cut algorithms to solve monotonic optimization problems by systematic use of these cuts.

Finally, it is worth mentioning that a new concept of the *essential ϵ-optimal solution* can be applied to monotonic optimization problems. The advantage of the method developed on the basis of this concept is the finding of an approximate optimal solution which is more appropriate and more stable than that which is found by the ϵ-optimal method. For details see [8].

Generalizations

The essential approach used in monotonic optimization can be further generalized to cover a wider class of non-convex general optimization problems. Among these generalizations, optimization of the difference of monotonic functions and discrete monotonic optimization are presented here.

Optimization of the Difference of Monotonic Functions

The underlying idea of monotonic optimization can be extended to deal with problems including the *difference of monotonic functions*. A function $f : R^n_+ \to R$ is said to be a difference of monotonic functions if it is representable as the difference of two increasing functions: $f_1 : R^n_+ \to R$ and $f_2 : R^n_+ \to R$. Similar to functions presented as the difference of convex functions, the class of difference of monotonic functions is a linear space. The pointwise minimum and pointwise maximum of a family of difference of monotonic functions (difference of convex functions) is still a difference of monotonic functions (difference of convex functions). The linear combination of a set of difference of monotonic functions is a difference of monotonic functions. Obviously, any polynomial function can be presented as the difference of two increasing functions, the first one includes all terms having positive coefficients and the second one includes all terms having negative coefficients.

Consider the problem:

$$\begin{aligned} \text{Maximize (minimize)} \quad & f(x) - g(x) \\ \text{subject to} \quad & x \in G \cap H\,, \end{aligned} \tag{3}$$

in which G and H are as before and $f(x)$ and $g(x)$ are increasing functions on $[0, b]$. Tuy [9] extended the original polyblock algorithm to solve this problem. By introducing t as the difference between $g(b)$

and $g(x)$ for $x \in [0, b]$ and regarding the fact that t is always positive owing the function $g(x)$ being increasing, we rewrite the model as (maximization case) $\max\{f(x) + t - g(b) | x \in G \cap H, t = g(b) - g(x)\}$. Now $g(b)$ is a constant and can be removed from the objective function. In the resulting problem, $\max\{f(x) + t | x \in G \cap H, 0 \le t \le g(b) - g(x)\}$, consider the set of constraints. By incrementing the dimension of the problem by one, the feasible space can be presented as $D \cap E$ such that $D = \{(x, t) | x \in G, t + g(x) \le g(b), 0 \le t \le g(b) - g(0)\}$ and $E = \{(x, t) | x \in H, 0 \le t \le g(b) - g(0)\}$. It is easy to verify that D is a normal set and H is a reverse normal set in the box $[0, b] \times [0, g(b) - g(0)]$. Also the function $F(x, t) = f(x) + t$ is an increasing function on $[0, b] \times [0, g(b) - g(0)]$. So problem (3) is reduced to problem (2) in Sect. "Normal Sets and Polyblocks" and can be treated by the original polyblock algorithm. The additional cost that the presence of difference of monotonic functions has incurred is the dimension of the problem incremented by one.

For the minimization case of problem (3), a similar transformation can be applied to convert this problem to the minimization case of problem (2).

To make the problem even more general, suppose that all constraints are also difference of monotonic functions. Specifically, consider the problem:

$$\begin{aligned} \text{Maximize (minimize)} \quad & f_1(x) - f_2(x) \\ \text{subject to} \quad & g_i(x) - h_i(x) \le 0 \\ & \forall i = 1, \dots, m\,, \\ & x \in \Omega \subset [0, b] \subset R^n_+\,, \end{aligned} \tag{4}$$

in which $f_1(x)$, $f_2(x)$, $g_i(x)$, and $h_i(x)$ are increasing functions and Ω is a normal set. By the above argument, first we can make a proper transformation and convert the objective function to an increasing function. So without loss of generality, let us assume that $f_2(x) = 0$. Now consider the set of m constraints. This set of constraints can be rewritten as $\max_i \{g_i(x) - h_i(x)\} \le 0$. Since the pointwise maximum of a family of difference of monotonic functions is still a difference of monotonic functions, we can represent the space imposed by these constraints by $g(x) - h(x) \le 0$, where both $g(x)$ and $h(x)$ are increasing. By introducing the new variable $t \ge 0$ and assuming $g(b) \ge 0$ (this assumption is not restrictive), the set of the following two constraints fully defines the space mentioned: $g(x) + t \le g(b)$, $h(x) + t \ge g(b)$. The first constraint gives us the upper bound of $g(b) - g(0)$ for t.

Finally the problem reduces to (maximization case): $\max\{f_1(x) | g(x) + t \le g(b),\ h(x) + t \ge g(b),\ x \in \Omega, 0 \le t \le g(b) - g(0)\}$. This problem is the same as problem (2) by defining $G = \{(x, t) | x \in \Omega, g(x) + t \le g(b), 0 \le t \le g(b) - g(0)\}$, which is a subset of the box $[0, b] \times [0, g(b) - g(0)]$ and $H = \{(x, t) | h(x) + t \ge g(b)\}$ is defined in R^{n+1}_+.

Increasing the dimension of the problem is the main drawback of the above mentioned approach. Tuy and Al-Khayyal [11] presented a direct approach for the difference of monotonic functions optimization problem requiring no additional dimension. This method is referred to as the branch-reduce-and-bound (BRB) algorithm. As the name of the algorithm suggests, it contains three main steps, which are branching upon nonconvex variables, reducing any partition set before bounding, and bounding over each partition set.

The branching phase is performed by rectangular subdivision. Every box is divided into two subboxes by a hyperplane. The reduction phase contains a set of operations by which the box $[p, q]$ is tightened without losing any feasible solution. This is called a *proper reduction* of $[p,q]$. This approach takes advantage of the monotonicity properties of the problem and increases the rate of convergence in the algorithm. In the bounding phase, for a properly reduced box $[p, q]$, an upper bound like β is obtained such that $\beta \ge \max\{f_1(x) - f_2(x) | g_i(x) - h_i(x) \le 0, \forall i = 1, \dots, m; x \in [p, q]\}$. As mentioned before, stronger bounds are obtained by a sequence of polyblock approximations or by combining monotonicity with convexity present in the problem. Furthermore, more complex methods can be applied to improve the quality of the bounds in the bounding phase.

Discrete Monotonic Optimization

A class of monotonic optimization problems containing the additional discrete constraints are called *discrete monotonic optimization problems*. Specifically, given a finite set S of points in the box $[a,b]$, the constraint $x \in S$ is added to the model. So the problem can be represented as $\max\{f(x) | x \in G \cap H \cap S\}$ (all the assumptions are as in problem (2).

The original polyblock algorithm is not practical for these problems. Since the polyblock algorithm is an iterative procedure, it does not have the capability to produce the optimal solution in a finite number of iterations. However, by making suitable modifications, one can use this algorithm to obtain the exact optimal solution of the problem in a finite number of steps [1,14]. In the new method, monotonicity cuts are adjusted on the basis of a special procedure to cope with discrete requirements. This adjustment consists in updating the vertex of the monotonicity cut by pushing it deeper inside the polyblock to obtain a tighter space while keeping all discrete points which are not proven to be nonoptimal, unaffected.

The algorithm first constructs the normal hull of $G \cap S$, denoted by $\tilde{G}$, and then tries to solve the problem $\max \left\{ f(x) | x \in \tilde{G} \cap H \right\}$ in continuous space. This method is called the *discrete polyblock algorithm*. For large-scale instances, a similar BRB algorithm was developed by Tuy et al. [14].

Applications

Although monotonic optimization is a new approach in global optimization and there is not a broad literature on its applications, it can be applied to numerous problems. In most of these applications, first some transformations are performed and the problems are reformulated in the proper way. Then monotonic optimization is applied and other approaches are employed to enhance the quality of the bounds. Some of these applications are briefly introduced in this section.

Polynomial programming: The problem of minimizing or maximizing a polynomial function under a set of polynomial constraints, which is encountered in a multitude of applications, is called *polynomial programming*. Tuy [9] reformulated this problem as a difference of monotonic functions problem which can be solved by the methods described before. Tuy [7] proposed a robust solution approach for polynomial programming based on a monotonic optimization scheme. He developed a BRB procedure to tackle the polynomial optimization problems of higher dimensions.

Polynomial optimization contains nonconvex quadratic programming as a special case. So every polynomial optimization method can be applied to solve this important class of problems [4,16].

Fractional programming: In *fractional programming*, we are dealing with functions which are represented by ratios of other functions. Phuong and Tuy [3] considered a generalized linear fractional programming problem. In this problem, the objective function consists of an arbitrary continuous increasing function of m linear fractional functions and the feasible set is the polytope D. Linear fractional functions are defined as the ratio of two linear affine functions. They proposed a new unified approach which reformulates the problem and solves it as a monotonic optimization problem.

Tuy [17] considered a more general class of fractional programming problems which is optimizing a polynomial fractional function (the ratio of two polynomial functions) under polynomial constraints. His method to solve the problem is again based on reformulating the problem as a monotonic optimization problem. A branch-and-bound scheme was presented for problems of higher ranks. Clearly, polynomial programming is a special case of this class of problems.

Multiplicative programming: *Multiplicative programming* problems are optimization problems containing products of a number of convex or concave functions in the objective function or constraints. Tuy [9] showed that these classes of problems are essentially monotonic optimization problems. Tuy and Nghia [15] devised a new approach based on the reverse polyblock approximation method for a broad class of problems including generalized linear multiplicative and linear fractional programming as special cases.

For more applications, including *Lipschitz optimization*, *optimization under network constraints*, the *Fekete points problem*, and the *Lennard-Jones potential energy function*, see [9].

Conclusions

We have discussed the recently developed theory of monotonic optimization as well as its generalizations and applications. This noble scheme which is capable of solving a wide range of nonconvex problems is based on an polyblock outer approximation procedure.

The approach that monotonic optimization uses to deal with optimization problems is analogous to convex optimization in several respects. Just as we approximate convex sets by polyhedrons, normal sets, defined as the level sets of increasing functions, can be approximated by a set of polyblocks in monotonic optimization. As the difference of convex functions plays an essential role in convex analysis (because any arbitrary continuous function can be represented as the difference of two convex functions), optimization problems representable as the difference of monotonic functions can be treated in monotonic optimization.

The performance of this method can be significantly improved by incorporating some other techniques like convex relaxation to exploit other properties present in the problem. In high dimensions, branch-and-bound or branch-and-cut extensions of the algorithm can be applied to overcome storage difficulties and increase the convergence speed.

References

1. Minoux M, Tuy H (2001) Discrete Monotonic Global Optimization. preprint. Institute of of Mathematics, Hanoi
2. Pardalos PM, Romeijn HE, Tuy H (2000) Recent developments and trends in global optimization. J Comput Appl Math 124:209–228
3. Phuong NTH, Tuy H (2003) A Unified Monotonic Approach to Generalized Linear Fractional Programming. J Global Optim 26:229–259
4. Phuong NTH, Tuy H (2002) A Monotonicity Based Approach to Nonconvex Quadratic Minimization. Vietnam J Math 30:373–393
5. Rubinov A, Tuy H, Mays H (2001) An Algorithm for Monotonic Global Optimization Problems. Optimization 49:205–221
6. Tuy H (2005) Partly Convex and Convex-Monotonic Optimization Problems. preprint, Institute of Mathematics, Hanoi
7. Tuy H (2005) Polynomial Optimization: A Robust Approach. preprint, Institute of Mathematics, Hanoi
8. Tuy H (2005) Robust Solution of Nonconvex Global Optimization Problems. J Global Optim 32:307–323
9. Tuy H (2000) Monotonic Optimization: Problems and Solution Approaches. SIAM J Optim 11:464–494
10. Tuy H (1999) Normal sets, Polyblocks, and Monotonic Optimizatin. Vietnam J Math 27:277–300
11. Tuy H, Al-Khayyal F (2003) Monotonic Optimization Revisited. preprint, Institute of Mathematics, Hanoi
12. Tuy H, Al-Khayyal F, Ahmed S (2001) Polyblock Algorithms Revisited. preprint, Institute of Mathematics, Hanoi
13. Tuy H, Al-Khayyal F, Thach PT (2005) Monotonic Optimization: Branch and Cut Methods. In: Audet C, Hansen P, Savard G (eds) Essays and Surveys in Global Optimization. Springer US, pp 39–78
14. Tuy H, Minoux M, Phuong NTH (2006) Discrete Monotonic Optimization with Application to a Discrete Location Problem. SIAM J Optim 17:78–97
15. Tuy H, Nghia ND (2001) Reverse Polyblock Approximation for Generalized Multiplicative/Fractional Programming. preprint, Institute of Mathematics, Hanoi
16. Tuy H, Phuong NTH (2007) A robust algorithm for quadratic optimization under quadratic constraints. J Global Optim 37:557–569
17. Tuy H, Thach PT, Konno H (2004) Optimization of Polynomial Fractional Functions. J Global Optim 29:19–44

Monte-Carlo Simulated Annealing in Protein Folding

YUKO OKAMOTO
Department Theoret. Stud. Institute Molecular Sci.
and Department Functional Molecular Sci.,
Graduate University Adv. Stud., Okazaki, Japan

MSC2000: 92C40

Article Outline

Keywords

Simulated annealing; Protein folding; Tertiary structure prediction; α-helix; β-sheet

We review uses of Monte-Carlo simulated annealing in the protein folding problem. We will discuss the strategy for tackling the protein folding problem based on all-atom models. Our approach consists of two elements: the inclusion of accurate solvent effects and the development of powerful simulation algorithms that can avoid getting trapped in states of energy local minima. For the former, we discuss several models varying in nature from crude (distance-dependent dielectric function) to rigorous (reference interaction site model).

For the latter, we show the effectiveness of Monte-Carlo simulated annealing.

Introduction

Proteins under their native physiological conditions spontaneously fold into unique three-dimensional structures (tertiary structures) in the time scale of milliseconds to minutes. Although protein structures appear to be dependent on various environmental factors within the cell where they are synthesized, it was inferred by experiments 'in vitro' that the three-dimensional structure of a protein is determined solely by its amino-acid sequence information [12]. Hence, it has been hoped that once the correct Hamiltonian of the system is given, one can predict the native protein tertiary structure from the first principles by computer simulations. However, this has yet to be accomplished. There are two reasons for the difficulty. One reason is that the inclusion of accurate solvent effects is nontrivial, because the number of solvent molecules that have to be considered is very large. The other reason for the difficulty comes from the fact that the number of possible conformations for each protein is astronomically large [30,60]. Simulations by conventional methods such as Monte-Carlo or molecular dynamics algorithms in canonical ensemble will necessarily be trapped in one of many local-minimum states in the energy function. In this article, I will discuss a possible strategy to alleviate these difficulties. The outline of the article is as follows. In Sect. "Energy Functions of Protein Systems" we summarize the energy functions of protein systems that we used in our simulations. In Sect. "Methods" we briefly review our simulation methods. In Sect. "Results" we present the results of our protein folding simulations. Section "Conclusions" is devoted to conclusions.

Energy Functions of Protein Systems

The energy function for the protein systems is given by the sum of two terms: the conformational energy E_P for the protein molecule itself and the solvation free energy E_S for the interaction of protein with the surrounding solvent. The conformational energy function E_P (in kcal/mol) for the protein molecule that we used is one of the standard ones. Namely, it is given by the sum of the electrostatic term E_C, 12-6 Lennard–Jones term E_{LJ}, and hydrogen-bond term E_{HB} for all pairs of atoms in the molecule together with the torsion term E_{tor} for all torsion angles:

$$\begin{cases} E_P = E_C + E_{LJ} + E_{HB} + E_{tor}, \\ E_C = \sum_{(i,j)} \dfrac{332 q_i q_j}{\epsilon r_{ij}}, \\ E_{LJ} = \sum_{(i,j)} \left(\dfrac{A_{ij}}{r_{ij}^{12}} - \dfrac{B_{ij}}{r_{ij}^{6}} \right), \\ E_{HB} = \sum_{(i,j)} \left(\dfrac{C_{ij}}{r_{ij}^{12}} - \dfrac{D_{ij}}{r_{ij}^{10}} \right), \\ E_{tor} = \sum_{i} U_i \left(1 \pm \cos(n_i \chi^i)\right). \end{cases} \tag{1}$$

Here, r_{ij} is the distance (in Å) between atoms i and j, ϵ is the dielectric constant, and χ^i is the torsion angle for the chemical bond i. Each atom is expressed by a point at its center of mass, and the partial charge q_i (in units of electronic charges) is assumed to be concentrated at that point. The factor 332 in E_C is a constant to express energy in units of kcal/mol. These parameters in the energy function as well as the molecular geometry were adopted from ECEPP/2 [37,41,57]. The computer code KONF90 [23,46] was used for all the Monte-Carlo simulations. For gas phase simulations, we set the dielectric constant ϵ equal to 2. The peptide-bond dihedral angles ω were fixed at the value 180° for simplicity. So, the remaining dihedral angles ϕ and ψ in the main chain and χ in the side chains constitute the variables to be updated in the simulations. One Monte-Carlo (MC) sweep consists of updating all these angles once with Metropolis evaluation [36] for each update.

Solvation free energy of interactions between a solute molecule and solvent molecules, in general, can be divided into three contributions: hydrophobic term that corresponds to the work required to create a cavity of the shape of the solute molecule in solution (the term 'hydrophobic' used in this article is different from a more standard one; see [11] for clarification on various definitions), the electrostatic term (including the hydrogen-bond energy) between solute and solvent molecules, and the Lennard–Jones term between solute and solvent molecules.

One of the simplest ways to represent solvent effects is by the sigmoidal, distance-dependent dielectric function [20,54]. The explicit form of the function we used is given by [43]

$$\epsilon(r) = D - \frac{D-2}{2}\left[(sr)^2 + 2sr + 2\right] e^{-sr}, \tag{2}$$

which is a slight modification of the one used in [9]. Here, we use $s = 0.3$ and $D = 78$. It approaches 2 (the value inside a protein) in the limit the distance r going to zero and 78 (the value for bulk water) in the limit r going to infinity. The distance-dependent dielectric function is simple and also computationally only slightly more demanding than the gas-phase case. But it only involves the electrostatic interactions. Other solvent contributions are hydrophobic interactions and Lennard–Jones interactions between protein and solvent.

Another commonly used term that represents solvent contributions is the term proportional to the solvent-accessible surface area of protein molecule. The solvation free energy E_S in this approximation is given by

$$E_S = \sum_i \sigma_i A_i, \tag{3}$$

where A_i is the solvent-accessible surface area of ith functional group, and σ_i is the proportionality constant. There are several versions of the set of the proportionality constants and functional groups. Five parameter sets were compared for the systems of peptides and a small protein, and we found that the parameter sets of [52,59] are valid ones [33]. The term in (3) includes all the contributions from solvent (namely, hydrophobic, electrostatic, and Lennard–Jones interactions), and it is therefore more accurate than the distance-dependent dielectric function. It is, however, an empirical representation, and its validity has to be eventually tested with a rigorous solvation theory.

The most widely-used and rigorous method of inclusion of solvent effects is probably the one that deals with the explicit solvent molecules with all-atom representations. Many molecular dynamics simulations of protein systems now directly include these explicit solvent molecules (for a review, see, for instance, [4]). Another rigorous method is based on the statistical mechanical theory of liquid and solution and is called the reference interaction site model (RISM) [7,21]. The RISM integral equation for solute-solvent (p-s) correlation functions in Fourier k-space is given by

$$\widetilde{\mathbf{h}}^{\mathbf{ps}} = \widetilde{\mathbf{w}}^{\mathbf{pp}}\widetilde{\mathbf{c}}^{\mathbf{ps}}\left(\widetilde{\mathbf{w}}^{\mathbf{ss}} + \boldsymbol{\rho}\widetilde{\mathbf{h}}^{\mathbf{ss}}\right), \tag{4}$$

where $\widetilde{\mathbf{h}}^{\mathbf{ps}}$ and $\widetilde{\mathbf{h}}^{\mathbf{ss}}$ are the matrices of the solute-solvent and the solvent-solvent total correlation functions, respectively, $\widetilde{\mathbf{c}}^{\mathbf{ps}}$ is the matrix of the solute-solvent direct correlation functions, $\widetilde{\mathbf{w}}^{\mathbf{pp}}$ and $\widetilde{\mathbf{w}}^{\mathbf{ss}}$ are the intramolecular correlation matrices for solute and solvent, respectively, and $\boldsymbol{\rho}$ is the number density matrix of the solvent. The solvation free energy is given by

$$E_S = 4\pi\rho k_B T \int_0^\infty r^2 F(r)\,\mathrm{d}r, \tag{5}$$

where $F(r)$ is defined by

$$F(r) \equiv \sum_{a,b}\left\{\frac{1}{2}h_{ab}^{ps}(r)^2 - c_{ab}^{ps}(r) - \frac{1}{2}h_{ab}^{ps}(r)c_{ab}^{ps}(r)\right\}. \tag{6}$$

Here, the summation indices a and b run over the solute and the solvent sites, respectively. A robust and fast algorithm for solving RISM equations was recently (as of 1999) developed [24], which made folding simulations of peptides a feasible possibility [25]. Although this method is computationally much more time-consuming than the first two methods (terms with distance-dependent dielectric function and those proportional to surface area), it gives the most accurate representation of the solvation free energy.

Methods

Once the appropriate energy function of the protein system is given, we have to employ a simulation method that does not get trapped in states of energy local minima. We have been advocating the use of Monte-Carlo simulated annealing [27].

In the regular canonical ensemble with a given inverse temperature $\beta \equiv 1/k_{BT}$, the probability weight of each state with energy E is given by the Boltzmann factor:

$$W_B(E) = \exp(-\beta E). \tag{7}$$

The probability distribution in energy is then given by

$$\mathsf{P}_B(T, E) \propto n(E) W_B(E), \tag{8}$$

where $n(E)$ is the number of states with energy E. Since the number of states $n(E)$ is an increasing function of energy and the Boltzmann factor $W_B(E)$ decreases exponentially with E, the probability distribution $\mathsf{P}_B(T,E)$ has a bell-like shape in general. When the temperature is high, β is small, and $W_B(E)$ decreases slowly with E. So, $\mathsf{P}_B(T,E)$ has a wide bell-shape. On the other hand, at low temperature β is large, and $W_B(E)$ decreases rapidly with E. So, $\mathsf{P}_B(T,E)$ has a narrow bell-shape (and in the limit $T \to 0\ K$, $\mathsf{P}_B(T,E) \propto \delta(E - E_{\mathrm{GS}})$, where E_{GS} is the global-minimum energy). However, it is very difficult to obtain canonical distributions at low temperatures with conventional simulation methods. This is because the thermal fluctuations at low temperatures are small and the simulation will certainly get trapped in states of energy local minima. Simulated annealing [27] is based on the process of crystal making. Namely, by starting a simulation at a sufficiently high temperature (much above the melting temperature), one lowers the temperature gradually during the simulation until it reaches the global-minimum-energy state (crystal). If the rate of temperature decrease is sufficiently slow so that thermal equilibrium may be maintained throughout the simulation, only the state with the global energy minimum is obtained (when the final temperature is 0 K). However, if the temperature decrease is rapid (quenching), the simulation will get trapped in a state of energy local minimum in the vicinity of the initial state.

Simulated annealing was first successfully used to predict the global-minimum-energy conformations of polypeptides and proteins [22,61,63] and to refine protein structures from X-ray and NMR data [5,42] almost a decade ago. Since then this method has been extensively used in the protein folding and structure refinement problems (for reviews, see [45,62]). Our group has been testing the effectiveness of the method mainly in oligopeptide systems. The procedure of our approach is as follows. While the initial conformations in the protein simulations are usually taken from the structures inferred by the experiments, our initial conformations are *randomly generated*. Each Monte-Carlo sweep updates every dihedral angle (in both the main chain and side chains) once. Our annealing schedule is as follows: The temperature is lowered exponentially from $T_I = 1000$ K to $T_F = 250$ K (the final temperature T_F was sometimes set equal to 100 K, 50 K, or 1 K) [23,46]. The temperature for the nth MC sweep is given by

$$T_n = T_I \gamma^{n-1}, \tag{9}$$

where γ is a constant which is determined by T_I, T_F, and the total number of MC sweeps of the run. Each run consists of $10^4 \sim 10^6$ MC sweeps, and we usually made 10 to 20 runs from different initial conformations.

Results

We now present the results of our simulations based on Monte-Carlo simulated annealing. All the simulations were started from randomly-generated conformations.

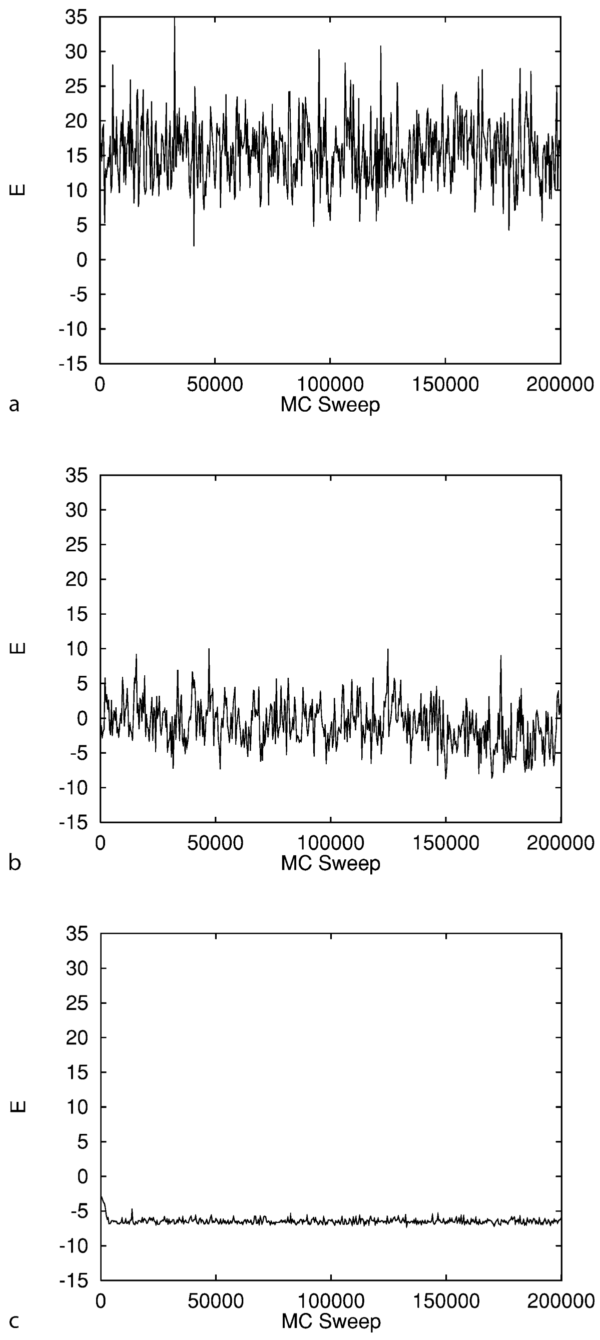
The first example is Met-enkephalin. This brain neuro peptide consists of 5 amino acids with the amino-acid sequence: Tyr-Gly-Gly-Phe-Met. Because it is one of the smallest peptides that have biological functions, it has served as a bench mark for testing a new simulation method. The global minimum conformation of this peptide for ECEPP/2 energy function in gas phase ($\epsilon = 2$) is known [31,49]. For KONF90 realization of ECEPP/2 energy, the peptide is essentially in the ground state for $E_{\mathrm{P}} \le -11$ kcal/mol [15,49] and the lowest value is -12.2 kcal/mol [16,17].

In Fig. 1, we show the 'time series' of the total conformational energy E_{P} (in (1)) obtained by conventional canonical Monte-Carlo simulations at $T = 1000$, 300, and 50 K.

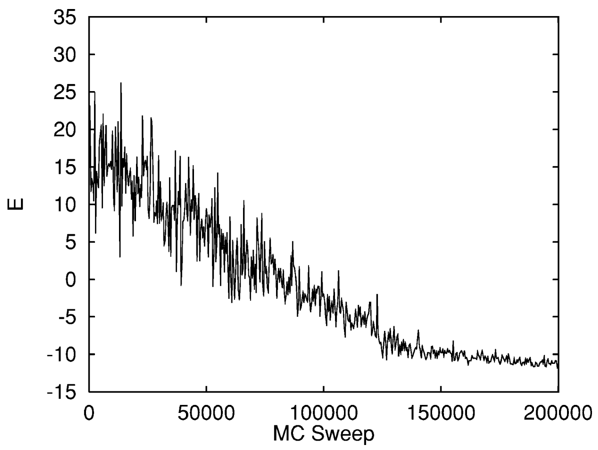
The thermal fluctuations for the run at $T = 50$ K in Fig. 1c are very small and this run has apparently gotten trapped in states of energy local minima (because the average energy at 50 K is about -11 kcal/mol [15,16]). In Fig. 2 we display the time series of energy obtained by a Monte-Carlo simulated annealing simulation.

This run reaches the global minimum region ($E_{\mathrm{P}} \le -11$ kcal/mol) as the temperature is decreased during the simulation from 1000 K to 50 K.

We have up to now presented the results in gas phase ($\epsilon = 2$). In Fig. 3 we compare the superposed structures of lowest-energy conformations from 8 Monte-Carlo simulated annealing runs in gas phase, simple-repulsive solvent, and water (the latter two contributions were calculated by the RISM theory) [26] with those of 5 structures inferred from NMR experiments ([13, Fig. 2]). The figures were created with RasMol [55].



Monte-Carlo Simulated Annealing in Protein Folding, Figure 1
Series of energy E_P (kcal/mol) of Met-enkephalin from conventional canonical Monte-Carlo runs at $T = 1000$ K (a), 300 K (b), and 50 K (c)

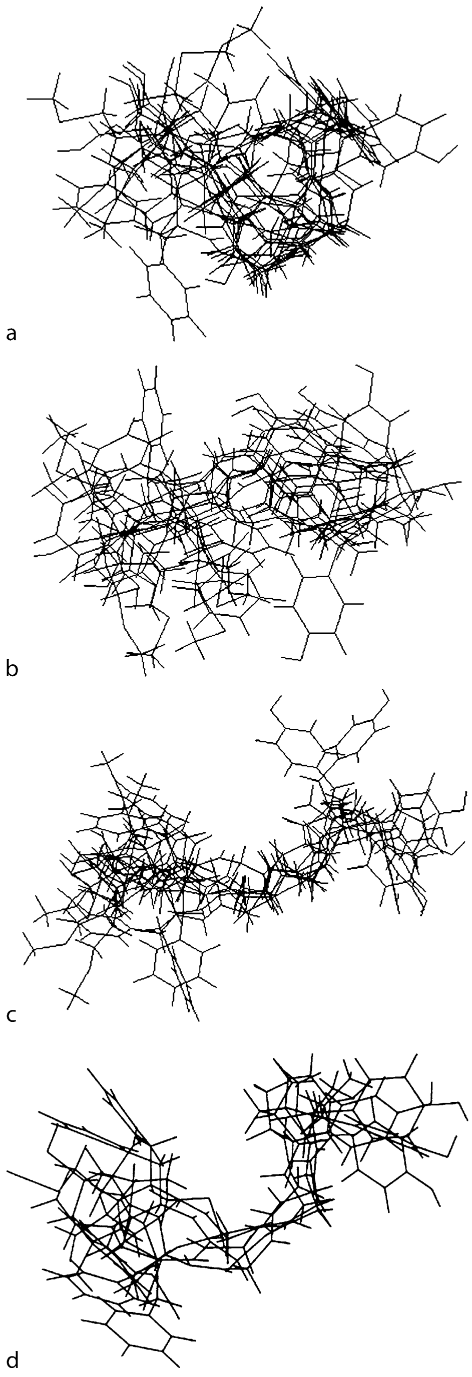


Monte-Carlo Simulated Annealing in Protein Folding, Figure 2
Time series of energy E_P (kcal/mol) of Met-enkephalin from a Monte-Carlo simulated annealing run

We see a striking similarity between simulation results in water Fig. 3c and those of NMR experiments (Fig. 3d). The simulation results in Fig. 3 are from the same number of MC sweeps. It seems that the presence of water speeds up the convergence of the backbone structures in the sense that it requires less number of MC sweeps for convergence [26].

The solvation free energy based on the RISM theory is very accurate, but it is also computationally very demanding. We are currently trying to solve this problem making the algorithm more efficient and robust [24]. Hereafter, we discuss how well other solvation theories can still describe the effects of solvent in the prediction of three-dimensional structures of oligopeptides and small proteins.

Next systems we discuss are those of homo-oligomers with length of 10 amino acids. From the structural data base of *X*-ray experiments of protein structures [8] and CD experiments [6], it is known that certain amino acids have more tendency of α-helix formation than others. For instance, alanine is a helix former and glycine is a helix breaker, while phenylalanine has intermediate helix-forming tendency. We have performed 20 Monte-Carlo simulated annealing runs of 10,000 MC sweeps in gas phase ($\epsilon = 2$) with each of $(\text{Ala})_{10}$, $(\text{Leu})_{10}$, $(\text{Met})_{10}$, $(\text{Phe})_{10}$, $(\text{Ile})_{10}$, $(\text{Val})_{10}$, and $(\text{Gly})_{10}$ [44]. These amino acids are nonpolar and we can avoid the complications of electrostatic and



Monte-Carlo Simulated Annealing in Protein Folding, Figure 3
Superposition of the eight conformations of Met-enkephalin obtained as the lowest-energy structures by Monte-Carlo simulated annealing in gas phase (a), simple-repulsive solvent (b), and water (c) together with superposition of five conformations deduced from the NMR experiment (d)

hydrogen-bond interactions of side chains with each other, with main chain, and with the solvent.

In order to analyze how much α-helix formation is obtained by simulations, we first define α-helix state of a residue. We consider that a residue is in the α-helix state when the dihedral angles (ϕ, ψ) fall in the range $(-60 \pm 45°, -50 \pm 45°)$ (Definition I) [23,46]. The length ℓ of a helical segment is then defined by the number of successive residues that are in the α-helix state. The number n of helical residues in a conformation is defined by the sum of ℓ over all helical segments in the conformation. Note that $\ell = 3$ corresponds to roughly one turn of α-helix. We therefore consider a conformation as helical if it has a segment with helix length $\ell \geq 3$.

The average values of the dihedral angles ϕ and ψ for the helical segments based on Definition I (with helix length $\ell \geq 3$) are $-70°$ and $-37°$, respectively, and the standard deviation is $\sim 10°$ for ECEPP/2 energy function [44,46]. Hence, for detailed analyses of the data we adopt a more stringent criterion for α-helix state (Definition II): The range is $(\phi, \psi) = (-70 \pm 20°, -37 \pm 20°)$ [44].

We likewise consider that a residue is in the β-strand state when the dihedral angles (ϕ, ψ) fall in the range $(-130 \pm 50°, 135 \pm 45°)$ [44]. The β-strand length m is then defined to be the number of successive residues that are in the β-strand state. We consider a conformation as β-stranded if it has a segment with β-strand length $m \geq 3$.

In Table 1 we summarize the α-helix formation in the 20 Monte-Carlo simulated annealing runs [44]. The results are for Definition II of the α-helix state.

We see that $(\text{Met})_{10}$, $(\text{Ala})_{10}$, and $(\text{Leu})_{10}$ gave many helical conformations: 15, 9, and 9 (out of 20), respectively. In particular, $(\text{Met})_{10}$ and $(\text{Ala})_{10}$ produced long helices, some conformations being almost entirely helical ($\ell \geq 8$). On the other hand, $(\text{Val})_{10}$, $(\text{Ile})_{10}$, and $(\text{Gly})_{10}$ gave few helical conformations: 2, 2, and 1 (out of 20), respectively. We obtained not only a smaller number of helices but also shorter helices for these homo-oligomers than the above three homo-oligomers. Finally, the results for $(\text{Phe})_{10}$ indicate that Phe has intermediate helix-forming tendency between these two groups. We thus have the following rank order of helix-forming tendency for the seven amino acids [44]:

$$\text{Met} > \text{Ala} > \text{Leu} > \text{Phe} > \text{Val} > \text{Ile} > \text{Gly}. \quad (10)$$

Monte-Carlo Simulated Annealing in Protein Folding, Table 1
α-Helix formation in homo-oligomers from 20 Monte-Carlo simulated annealing runs

Peptide ℓ	$(Met)_{10}$	$(Ala)_{10}$	$(Leu)_{10}$	$(Phe)_{10}$	$(Val)_{10}$	$(Ile)_{10}$	$(Gly)_{10}$
3	1	0	4	1	0	2	1
4	2	0	2	2	2	0	0
5	0	1	1	1	0	0	0
6	2	3	2	1	0	0	0
7	2	1	0	0	0	0	0
8	7	4	0	0	0	0	0
9	1	0	0	0	0	0	0
10	0	0	0	0	0	0	0
Total	15/20	9/20	9/20	5/20	2/20	2/20	1/20

This can be compared with the experimentally determined helix propensities [6,8]. Our rank order (10) is in good agreement with the experimental data.

We then analyzed the relation between helix-forming tendency and energy. We found that the differences $\Delta E = E_{NH} - E_H$ between minimum energies for nonhelical (NH) and helical (H) conformations is large for homo-oligomers with high helix-forming tendency (9.7, 10.2, 21.5 kcal/mol for $(Met)_{10}$, $(Ala)_{10}$, $(Leu)_{10}$, respectively) and small for those with low helix-forming tendency (0.5, 1.6, −3.2 kcal/mol for $(Val)_{10}$, $(Ile)_{10}$, $(Gly)_{10}$, respectively). Moreover, we found that the large ΔE for the former homo-oligomers are caused by the Lennard–Jones term ΔE_{LJ} (13.3, 8.0, 17.5 kcal/mol for $(Met)_{10}$, $(Ala)_{10}$, $(Leu)_{10}$, respectively). Hence, we conjecture that the differences in helix-forming tendencies are determined by the following factors [44]. A helical conformation is energetically favored in general because of the Lennard–Jones term E_{LJ}. For amino acids with low helix-forming tendency except for Gly, however, the steric hindrance of side chains raises E_{LJ} of helical conformations so that the difference ΔE_{LJ} between nonhelical and helical conformations are reduced significantly. The small ΔE_{LJ} for these amino acids can be easily overcome by the entropic effects and their helix-forming tendencies are small. Note that such amino acids (Val and Ile here) have two large side-chain branches at C^{β}, while the helix forming amino acids such as Met and Leu have only one branch at C^{β} and Ala has a small side chain.

We now study the β-strand forming tendencies of these seven homo-oligomers. In Table 2 we summarize the β-strand formation in 20 Monte-Carlo simulated annealing runs [44].

The implications of the results are not as obvious as in the α-helix case. This is presumably because a short, isolated β-strand is not very stable by itself, since hydrogen bonds between β-strands are needed to stabilize them. However, we can still give a rough estimate for the rank order of strand-forming tendency for the seven amino acids [44]:

$$\text{Val} > \text{Ile} > \text{Phe} > \text{Leu} > \text{Ala} > \text{Met} > \text{Gly}. \qquad (11)$$

Here, we considered Val as more strand-forming than Ile, since the longer the strand segment is, the harder it is to form by simulation. Our rank order (11) is again in good agreement with the experimental data [8].

By comparing (11) with (10), we find that the helix-forming group is the strand-breaking group and vice versa, except for Gly. Gly is both helix and strand breaking. This reflects the fact that Gly, having no side chain, has a much larger (backbone) conformational space than other amino acids.

The helix-coil transitions of homo-oligomer systems were further analyzed by multicanonical algorithms [3] in [47,48]. The obtained results gave quantitative support to those by Monte-Carlo simulated annealing described above [44].

We have so far studied peptides with nonpolar amino acids each of which is electrically neutral as a whole. We now discuss the helix-forming tendencies of peptides with polar amino acids where side chains are charged by protonation or deprotonation. One example is the C-peptide, residues 1–13 of ribonuclease A.

Monte-Carlo Simulated Annealing in Protein Folding, Table 2
***β*-Strand formation in homo-oligomers from 20 Monte-Carlo simulated annealing runs**

Peptide *m*	$(Met)_{10}$	$(Ala)_{10}$	$(Leu)_{10}$	$(Phe)_{10}$	$(Val)_{10}$	$(Ile)_{10}$	$(Gly)_{10}$
3	0	0	2	5	1	7	0
4	0	0	0	1	0	4	0
5	0	0	0	0	2	1	0
6	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0
8	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0
Total	0/20	0/20	2/20	6/20	5/20	12/20	0/20

It is known from the *X*-ray diffraction data of the whole enzyme that the segment from Ala-4 to Gln-11 exhibits a nearly 3-turn α-helix [58,64]. It was also found by CD [56] and NMR [53] experiments that the isolated C-peptide also has significant α-helix formation in aqueous solution at temperatures near 0°C.

Furthermore, the CD experiment of the isolated C-peptide showed that the side-chain charges of residues Glu-2^- and His-12^+ enhance the stability of the α-helix, while the rest of the charges of other side chains do not [56]. The NMR experiment [53] of the isolated C-peptide further observed the formation of the characteristic salt bridge between Glu-2^- and Arg-10^+ that exists in the native structure determined by the *X*-ray experiments of the whole protein [58,64].

In order to test whether our simulations can reproduce these experimental results, we made 20 Monte-Carlo simulated annealing runs of 10,000 MC sweeps with several C-peptide analogues [23,46]. The amino-acid sequences of four of the analogues are listed in Table 3.

Monte-Carlo Simulated Annealing in Protein Folding, Table 3
Amino-acid sequences of the peptide analogues of C-peptide studied by Monte-Carlo simulated annealing

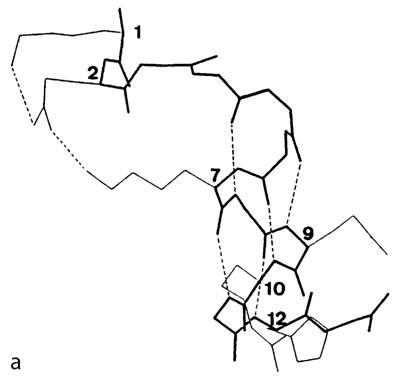
Peptide	I	II	III	IV
Sequence				
1	Lys^+			
2	Glu^-			Glu
3	Thr			
4	Ala			
5	Ala			
6	Ala			
7	Lys^+			
8	Phe			
9	Glu^-	Glu	Leu	
10	Arg^+			
11	Gln			
12	His^+			His
13	Met			

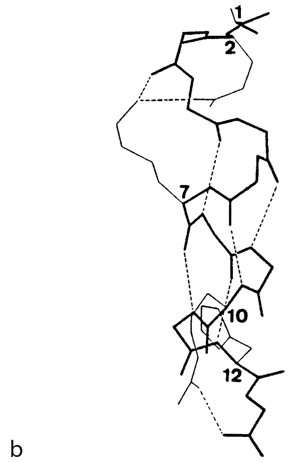
The simulations were performed in gas phase ($\epsilon = 2$). The temperature was decreased exponentially from 1000 K to 250 K for each run. As usual, all the simulations were started from random conformations.

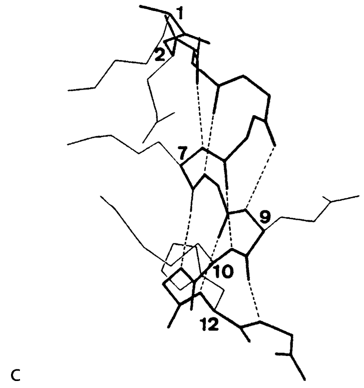
In Table 4 we summarize the helix formation of all the runs [46]. Here, the number of conformations with segments of helix length $\ell \geq 3$ are given with Definition I of the α-helix state. From this table one sees that α-helix was hardly formed for Peptide IV where Glu-2 and His-12 are neutral, while many helical conformations were obtained for the other peptides. This is in accord with the experimental results that the charges of Glu-2^- and His-12^+ are necessary for the α-helix stability [56].

Peptides II and III had conformations with the longest α-helix ($\ell = 7$). These conformations turned out to have the lowest energy in 20 simulation runs for each peptide. They both exhibit an α-helix from Ala-5 to Gln-11, while the structure from the X-ray data has an α-helix from Ala-4 to Gln-11. These three conformations are compared in Fig. 4.

As mentioned above, the agreement of the backbone structures is conspicuous, but the side-chain







Monte-Carlo Simulated Annealing in Protein Folding, Figure 4
The lowest-energy conformations of Peptide II (a) and Peptide III (b) of C-peptide analogues obtained from 20 Monte-Carlo simulated annealing runs in gas phase, and the corresponding *X*-ray structure (c)

Monte-Carlo Simulated Annealing in Protein Folding, Table 4
α-Helix formation in C-peptide analogues from 20 Monte-Carlo simulated annealing runs

Peptide ℓ	I	II	III	IV
3	4	2	3	1
4	3	2	3	0
5	1	1	0	0
6	0	1	0	0
7	0	1	1	0
Total	8/20	7/20	7/20	1/20

structures are not quite similar. In particular, while the *X*-ray [58,64] and NMR [53] experiments imply the formation of the salt bridge between the side chains of Glu-2^- and Arg-10^+, the lowest-energy conformations of Peptides II and III obtained from the simulations do not have this salt bridge.

The disagreement is presumably caused by the lack of solvent in our simulations. We have therefore made multicanonical Monte-Carlo simulations of Peptide II with the inclusion of solvent effects by the distance-dependent dielectric function (see (2)) [18,19]. It was found that the lowest-energy conformation obtained has an α-helix from Ala-4 to Gln-11 and does have the characteristic salt bridge between Glu-2^- and Arg-10^+ [18,19].

Similar dependence of α-helix stability on side-chain charges was observed in Monte-Carlo simulated annealing runs of a 17-residue synthetic peptide [43]. The pH difference in the experimental conditions was represented by the corresponding difference in charge assignment of the side chains, and the agreement with the experimental results (stable α-helix formation at low pH and low helix content at high pH) was observed in the simulations by Monte-Carlo simulated annealing with the distance-dependent dielectric function [43].

Considering our simulation results on homo-oligomers of nonpolar amino acids, C-peptide, and the synthetic peptide, we conjecture that the helix-forming tendencies of oligopeptide systems are controlled by the following factors [43]. An α-helix structure is generally favored energetically (especially, the Lennard–Jones term). When side chains are uncharged, the steric hindrance of side chains is the key factor for the difference in helix-forming tendency. When some of the side chains are charged, however, these charges play an important role in the helix stability in addition to the above factor: Some charges enhance helix stability, while others reduce it.

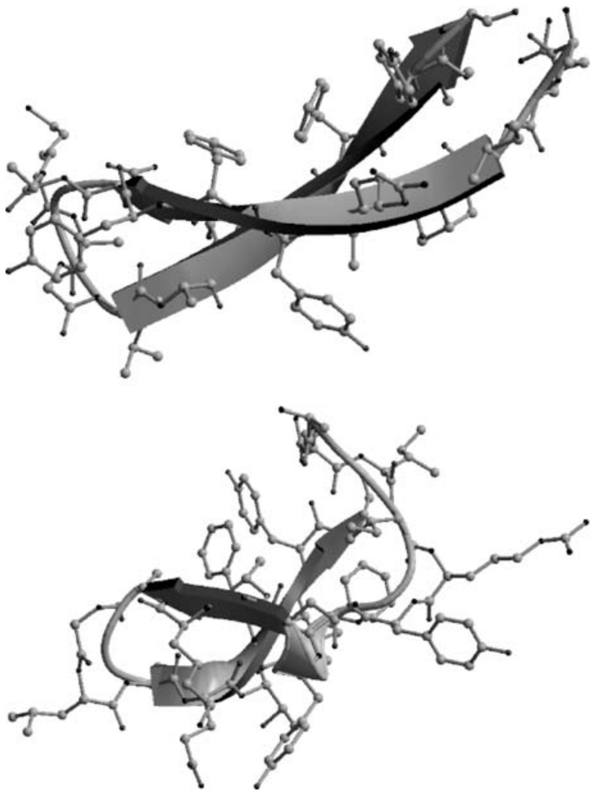
We have up to now discussed α-helix formations in our simulations of oligopeptide systems. We have also studied β-sheet formations by Monte-Carlo simulated annealing [38,39,51]. The peptide that we studied is the fragment corresponding to residues 16–36 of bovine pancreatic trypsin inhibitor (BPTI) and has the amino-acid sequence: Ala^{16}-Arg^{+}-Ile-Ile-Arg^{+}-Tyr-Phe -Tyr -Asn -Ala -Lys^{+} -Ala -Gly -Leu -Cys -Gln -Thr-Phe-Val-Tyr-Gly^{36}. An antiparallel β-sheet structure in residues 18–35 is observed in X-ray crystallographic data of the whole protein [10].

We first performed 20 Monte-Carlo simulated annealing runs of 10,000 MC sweeps in gas phase ($\epsilon = 2$) with the same protocol as in the previous simulations [38]. Namely, the temperature was decreased exponentially from 1000 K to 250 K for each run, and all the simulations were started from random conformations. The difference of the present simulation and the previous ones comes only from that of the amino-acid sequences.

The most notable feature of the obtained results is that α-helices, which were the dominant motif in previous simulations of C-peptide and other peptides, are absent in the present simulation. Most of the conformations obtained consist of stretched strands and a 'turn' which connects them. The lowest-energy structure indeed exhibits an antiparallel β-sheet [38].

We next made 10 Monte-Carlo simulated annealing runs of 100,000 MC sweeps for BPTI(16–36) with two dielectric functions: $\epsilon = 2$ and the sigmoidal, distance-dependent dielectric function of (2) [39]. The results with $\epsilon = 2$ reproduced our previous results: Most of the obtained conformations have β-strand structures and no extended α-helix is observed. Those with the sigmoidal dielectric function, on the other hand, indicated formation of α-helices. One of the low-energy conformations, for instance, exhibited about a four-turn α-helix from Ala-16 to Gly-28 [39]. This presents an example in which a peptide with the same amino-acid sequence can form both α-helix and β-sheet structures, depending on its electrostatic environment.

NMR experiments suggest that this peptide actually forms a β-sheet structure [40]. The representation of


Monte-Carlo Simulated Annealing in Protein Folding, Figure 5
The structure of BPTI(16–36) deduced from X-ray experiments (a) and the lowest-energy conformation of BPTI(16–36) obtained from 20 Monte-Carlo simulated annealing runs in aqueous solution represented by solvent-accessible surface area (b)

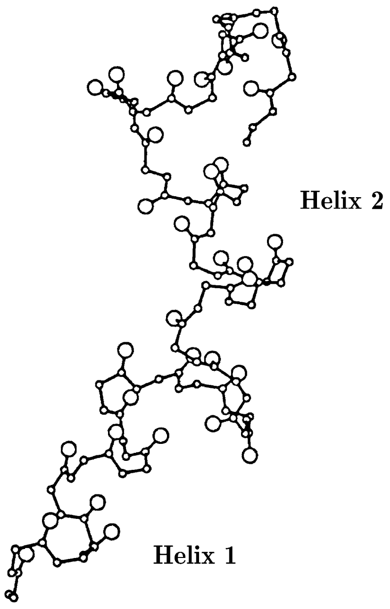
solvent by the sigmoidal dielectric function (which gave α-helices instead) is therefore not sufficient. Hence, the same peptide fragment, BPTI(16–36), was further studied in aqueous solution that is represented by solvent-accessible surface area of (3) by Monte-Carlo simulated annealing [51]. Twenty simulation runs of 100,000 MC sweeps were made. It was indeed found that the lowest-energy structure obtained has a β-sheet structure (actually, type II′ β-turn) at the very location suggested by the NMR experiments [40]. This structure and that deduced from the X-ray experiments [10] are compared in Fig. 5. The figures were created with Molscript [29] and Raster3D [2,35].

Although both conformations are β-sheet structures, there are important differences between the two: The positions and types of the turns are different. Since the *X*-ray structure is taken from the experiments on the whole BPTI molecule, it does not have to agree with that of the isolated BPTI(16–36) fragment. It was found [51] that the simulated results in Fig. 5b have remarkable agreement with those in the NMR experiments of the isolated fragment [40].

We have so far dealt with peptides with small number of amino acids (up to 21) with simple secondary structural elements: a single α-helix or β-sheet. The native proteins usually have more than one secondary structural elements. We now discuss our attempts on the first-principles tertiary structure predictions of larger and more complicated systems.

The first example is the fragment corresponding to residues 1–34 of human parathyroid hormone (PTH). An NMR experiment of PTH(1–34) suggested the existence of two α-helices around residues from Ser-3 to His-9 and from Ser-17 to Leu-28 [28]. Another NMR experiment of a slightly longer fragment, PTH(1–37), in aqueous solution also suggested the existence of the two helices [32]. One of the determined structures, for instance, has α-helices in residues from Gln-6 to His-9 and from Ser-17 to Lys-27 [32].

For PTH(1–34) we performed 20 Monte-Carlo simulated annealing runs of 10,000 MC sweeps in gas phase ($\epsilon = 2$) with the same protocol as in the previous simulations [50]. Many conformations among the 20 final conformations obtained exhibited α-helix structures (especially in the N-terminus area). In Fig. 6 we show the lowest-energy conformation of PTH(1–34) [50].



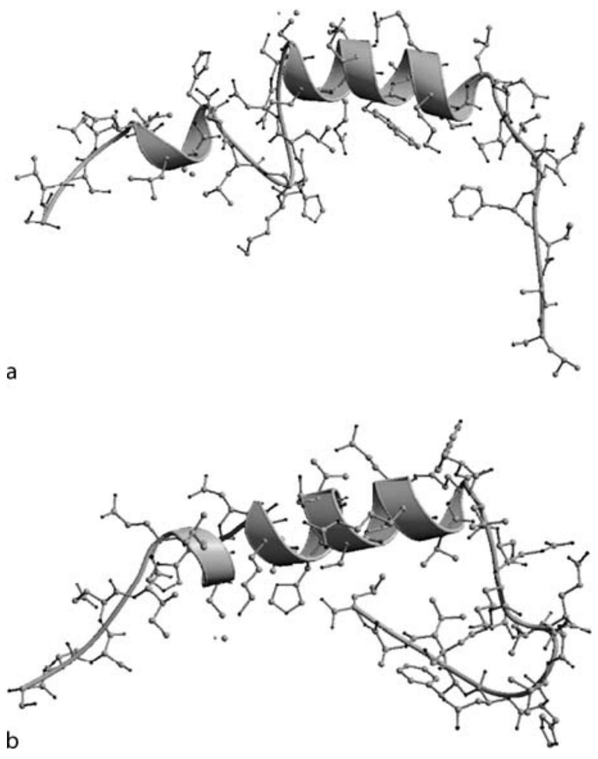
Monte-Carlo Simulated Annealing in Protein Folding, Figure 6
Lowest-energy conformation of PTH(1–34) obtained from 20 Monte-Carlo simulated annealing runs in gas phase

This conformation indeed has two α-helices around residues from Val-2 to Asn-10 (Helix 1) and from Met-18 to Glu-22 (Helix 2), which are precisely the same locations as suggested by experiment [28], although Helix 2 is somewhat shorter (5 residues long) than the corresponding one (12 residues long) in the experimental data.

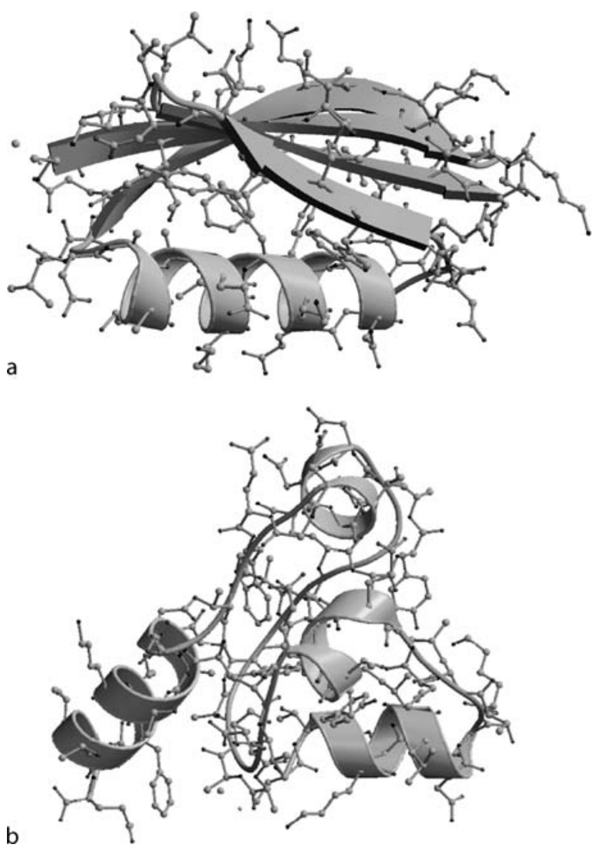
A slightly larger peptide fragment, PTH(1–37), was also studied by Monte-Carlo simulated annealing [34] to compare with the results of the recent NMR experiment in aqueous solution [32]. Ten simulation runs of 100,000 MC sweeps were made in gas phase ($\epsilon = 2$) and in aqueous solution that is represented by the terms proportional to the solvent-accessible surface area (see (3)). Although the results are preliminary, the simulations in gas phase did not produce two helices this time in contrast to the previous work [50], where a short second helix was observed, as discussed in the previous paragraph. The lowest-energy conformation has an α-helix from Val-2 to Asn-10. The simulations in aqueous solution, on the other hand, did observe the two α-helices. The lowest-energy conformation obtained has α-helices from Gln-6 to His-9 and from Gly-12 to Glu-22. Note that the second helix is now more extended than the first one in agreement with experiments. This structure together with one of the NMR structure [32] is shown in Fig. 7. The figures were again created with Molscript [29] and Raster3D [2,35].

Generalized-ensemble simulations of PTH(1–37) are now in progress in order to obtain more quantitative information such as average helicity as a function of residue number, etc.

The second example of more complicated system is the immunoglobulin-binding domain of streptococcal protein G. This protein is composed of 56 amino acids and the structure determined by an NMR experiment [14] and an *X*-ray diffraction experiment [1] has an α-helix and a β-sheet. The α-helix extends from residue Ala-23 to residue Asp-36. The β-sheet is made of four β-strands: from Met-1 to Gly-9, from Leu-12 to Ala-20, from Glu-42 to Asp-46, and from Lys-50 to Glu-56.



Monte-Carlo Simulated Annealing in Protein Folding, Figure 7
A structure of PTH(1–37) deduced from NMR experiments (a) and the lowest-energy conformation of PTH(1–37) obtained from 10 Monte-Carlo simulated annealing runs in aqueous solution represented by solvent-accessible surface area (b)



Monte-Carlo Simulated Annealing in Protein Folding, Figure 8
A structure of protein G deduced from an X-ray experiment (a) and the lowest-energy conformation of protein G obtained from Monte-Carlo simulated annealing runs with the distance-dependent dielectric function (b)

This structure is shown in Fig. 8a). The figures in Fig. 8 were again created with Molscript [29] and Raster3D [2,35].

We have performed eight Monte-Carlo simulated annealing runs of 50,000 to 400,000 MC sweeps with the sigmoidal, distance-dependent dielectric function of (2). The lowest-energy conformation so far obtained has four α-helices and no β-sheet in disagreement with the X-ray structure. This structure is shown in Fig. 8b).

The disagreement of the lowest-energy structure (Fig. 8b) so far obtained with the X-ray structure (Fig. 8a) is presumably caused by the poor representation of the solvent effects. As can been seen in Fig. 8a), the X-ray structure has both interior where a well-defined hydrophobic core is formed and exterior where it is exposed to the solvent. The distance-dependent dielectric function, which mimics the solvent effects only in electrostatic interactions, is therefore not sufficient to represent the effects of the solvent here.

Conclusions

In this article we have reviewed theoretical aspects of the protein folding problem. Our strategy in tackling this problem consists of two elements: 1) inclusion of accurate solvent effects, and 2) development of powerful simulation algorithms that can avoid getting trapped in states of energy local minima.

We have shown the effectiveness of Monte-Carlo simulated annealing by showing that direct folding of α-helix and β-sheet structures from randomly-generated initial conformations are possible.

As for the solvent effects, we considered several methods: a distance-dependent dielectric function, a term proportional to solvent-accessible surface area, and the reference interaction site model (RISM). These methods vary in nature from crude but computationally inexpensive (distance-dependent dielectric function) to accurate but computationally demanding (RISM theory). In the present article, we have shown that the inclusion of some solvent effects is very important for a successful prediction of the tertiary structures of small peptides and proteins.

See also

- ▶ Adaptive Simulated Annealing and its Application to Protein Folding
- ▶ Bayesian Global Optimization
- ▶ Genetic Algorithms
- ▶ Genetic Algorithms for Protein Structure Prediction
- ▶ Global Optimization Based on Statistical Models
- ▶ Global Optimization in Lennard–Jones and Morse Clusters
- ▶ Global Optimization in Protein Folding
- ▶ Molecular Structure Determination: Convex Global Underestimation
- ▶ Monte-Carlo Simulations for Stochastic Optimization
- ▶ Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach
- ▶ Packet Annealing
- ▶ Phase Problem in X-ray Crystallography: Shake and Bake Approach
- ▶ Protein Folding: Generalized-ensemble Algorithms
- ▶ Random Search Methods
- ▶ Simulated Annealing
- ▶ Simulated Annealing Methods in Protein Folding
- ▶ Stochastic Global Optimization: Stopping Rules
- ▶ Stochastic Global Optimization: Two-phase Methods

References

1. Achari A, Hale SP, Howard AJ, Clore GM, Gronenborn AM, Hardman KD, Whitlow M (1992) 1.67- Å X-ray structure of the B2 immunoglobulin-binding domain of streptococcal protein G and comparison to the NMR structure of the B1 domain. Biochemistry 31:10449–10457
2. Bacon D, Anderson WF (1988) A fast algorithm for rendering space-filling molecular pictures. J Mol Graphics 6:219–220
3. Berg BA, Neuhaus T (1991) Multicanonical algorithms for first order phase transitions. Phys Lett B267:249–253
4. Brooks III CL (1998) Simulations of protein folding and unfolding. Curr Opin Struct Biol 8:222–226
5. Brünger AT (1988) Crystallographic refinement by simulated annealing: Application to a 2.8 Å resolution structure of aspartate aminotransferase. J Mol Biol 203:803–816
6. Chakrabartty A, Kortemme T, Baldwin RL (1994) Helix propensities of the amino acids measured in alanine-based peptides without helix-stabilizing side-chain interactions. Protein Sci 3:843–852
7. Chandler D, Andersen HC (1972) Optimized cluster expansions for classical fluids. Theory of molecular liquids. J Chem Phys 57:1930–1937
8. Chou PY, Fasman GD (1974) Prediction of protein conformation. Biochemistry 13:222–245
9. Daggett V, Kollman PA, Kuntz ID (1991) Molecular dynamics simulations of small peptides: dependence on dielectric model and pH. Biopolymers 31:285–304
10. Deisenhofer J, Steigemann W (1975) Crystallographic refinement of the structure of bovine pancreatic trypsin inhibitor at 1.5 Å resolution. Acta Crystallogr B31:238–250
11. Dill K (1990) The meaning of hydrophobicity. Science 250:297–297
12. Epstain CJ, Goldberger RF, Anfinsen CB (1963) The genetic control of tertiary protein structure: studies with model systems. Cold Spring Harbor Symp Quant Biol 28: 439–449
13. Graham WH, Carter ES, II, Hicks RP (1992) Conformational analysis of Met-enkephalin in both aqueous solution and in the presence of sodium dodecyl sulfate micelles using multidimensional NMR and molecular modeling. Biopolymers 32:1755–1764
14. Gronenborn AM, Filpula DR, Essig NZ, Achari A, Whitlow M, Wingfield PT, Clore GM (1991) A novel, highly stable fold of the immunoglobulin binding domain of streptococcal protein G. Science 253:657–661
15. Hansmann UHE, Okamoto Y (1993) Prediction of peptide conformation by multicanonical algorithm: new approach to the multiple-minima problem. J Comput Chem 14:1333–1338
16. Hansmann UHE, Okamoto Y (1994) Comparative study of multicanonical and simulated annealing algorithms in the protein folding problem. Phys A 212:415–437
17. Hansmann UHE, Okamoto Y (1994) Sampling ground-state configurations of a peptide by multicanonical annealing. J Phys Soc Japan 63:3945–3949
18. Hansmann UHE, Okamoto Y (1998) Tertiary structure prediction of C-peptide of ribonuclease A by multicanonical algorithm. J Phys Chem B 102:653–656
19. Hansmann UHE, Okamoto Y (1999) Effects of side-chain charges on α-helix stability in C-peptide of ribonuclease

A studied by multicanonical algorithm. J Phys Chem B 103:1595–1604
20. Hingerty BE, Ritchie RH, Ferrell T, Turner JE (1985) Dielectric effects in biopolymers: the theory of ionic saturation revisited. Biopolymers 24:427–439
21. Hirata F, Rossky PJ (1981) An extended RISM equation for molecular polar fluids. Chem Phys Lett 83:329–334
22. Kawai H, Kikuchi T, Okamoto Y (1989) A prediction of tertiary structures of peptide by the Monte Carlo simulated annealing method. Protein Eng 3:85–94
23. Kawai H, Okamoto Y, Fukugita M, Nakazawa T, Kikuchi T (1991) Prediction of α-helix folding of isolated C-peptide of ribonuclease A by Monte Carlo simulated annealing. Chem Lett:213–216
24. Kinoshita M, Okamoto Y, Hirata F (1997) Calculation of hydration free energy for a solute with many atomic sites using the RISM theory: robust and efficient algorithm. J Comput Chem 18:1320–1326
25. Kinoshita M, Okamoto Y, Hirata F (1997) Solvation structure and stability of peptides in aqueous solutions analyzed by the reference interaction site model theory. J Chem Phys 107:1586–1599
26. Kinoshita M, Okamoto Y, Hirata F (1998) First-principle determination of peptide conformations in solvents: combination of Monte Carlo simulated annealing and RISM theory. J Amer Chem Soc 120:1855–1863
27. Kirkpatrick S, Gelatt, CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
28. Klaus W, Dieckmann T, Wray V, Schomburg D, Wingender E, Mayer H (1991) Investigation of the solution structure of the human parathyroid hormone fragment (1–34) by ^{1}H NMR spectroscopy, distance geometry, and molecular dynamics calculations. Biochemistry 30:6936–6942
29. Kraulis PJ (1991) MOLSCRIPT: A program to produce both detailed and schematic plots of protein structures. J Appl Crystallogr 24:946–950
30. Levinthal C (1968) Are there pathways for protein folding? J Chem Phys 65:44–45
31. Li Z, Scheraga HA (1987) Monte Carlo-minimzation approach to the multiple-minima problem in protein folding. Proc Natl Acad Sci USA 84:6611–6615
32. Marx UT, Austermann S, Bayer P, Adermann K, Ejchart A, Sticht H, Walter S, Schmid F-X, Jaenicke R, Forssmann W-G, Rösch P (1995) Structure of human parathyroid hormone 1–37 in solution. J Biol Chem 270:15194–15202
33. Masuya M, Okamoto Y, in preparation
34. Masuya M, Okamoto Y, in preparation
35. Merritt EA, Murphy MEP (1994) Raster3D version 2.0. A program for photorealistic molecular graphics. Acta Crystallogr D50:869–873
36. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21:1087–1092
37. Momany FA, McGuire RF, Burgess AW, Scheraga HA (1975) Energy parameters in polypeptides. VII. Geometric parameters, partial atomic charges, nonbonded interactions, hydrogen bond interactions, and intrinsic torsional potentials for the naturally occurring amino acids. J Phys Chem 79:2361–2381
38. Nakazawa T, Kawai H, Okamoto Y, Fukugita M (1992) β-sheet folding of bovine pancreatic trypsin inhibitor fragment (16–36) as predicted by Monte Carlo simulated annealing. Protein Eng 5:495–503
39. Nakazawa T, Okamoto Y (1999) Electrostatic effects on the α-helix and β-strand folding of BPTI(16–36) as predicted by Monte Carlo simulated annealing. J Peptide Res 54:230–236
40. Nakazawa T, Okamoto Y, Kobayashi Y, Kyogoku Y, Aimoto S, in preparation
41. Némethy G, Pottle MS, Scheraga HA (1983) Energy parameters in polypeptides. 9. Updating of geometrical parameters, nonbonded interactions, and hydrogen bond interactions for the naturally occurring amino acids. J Phys Chem 87:1883–1887
42. Nilges M, Clore GM, Gronenborn AM (1988) Determination of three-dimensional structures of proteins from interproton distance data by hybrid distance geometry-dynamical simulated annealing calculations. FEBS Lett 229:317–324
43. Okamoto Y (1994) Dependence on the dielectric model and pH in a synthetic helical peptide studied by Monte Carlo simulated annealing. Biopolymers 34:529–539
44. Okamoto Y (1994) Helix-forming tendencies of nonpolar amino acids predicted by Monte Carlo simulated annealing. PROTEINS: Struct Funct Genet 19:14–23
45. Okamoto Y (1998) Protein folding problem as studied by new simulation algorithms. Recent Res Developm Pure Appl Chem 2:1–23
46. Okamoto Y, Fukugita M, Nakazawa T, Kawai H (1991) α-helix folding by Monte Carlo simulated annealing in isolated C-peptide of ribonuclease A. Protein Eng 4:639–647
47. Okamoto Y, Hansmann UHE (1995) Thermodynamics of helix-coil transitions studied by multicanonical algorithms. J Phys Chem 99:11276–11287
48. Okamoto Y, Hansmann UHE, Nakazawa T (1995) α-Helix propensities of amino acids studied by multicanonical algorithm. Chem Lett 391–392
49. Okamoto Y, Kikuchi T, Kawai H (1992) Prediction of low-energy structures of Met-enkephalin by Monte Carlo simulated annealing. Chem Lett 1275–1278
50. Okamoto Y, Kikuchi T, Nakazawa T, Kawai H (1993) α-Helix structure of parathyroid hormone fragment (1–34) predicted by Monte Carlo simulated annealing. Internat J Peptide Protein Res 42:300–303
51. Okamoto Y, Masuya M, Nabeshima M, Nakazawa T (1999) β-Sheet formation in BPTI(16–36) by Monte Carlo simulated annealing. Chem Phys Lett 299:17–24
52. Ooi T, Oobatake M, Némethy G, Scheraga HA (1987) Accessible surface areas as a measure of the thermodynamic parameters of hydration of peptides. Proc Natl Acad Sci USA 84:3086–3090

53. Osterhout JJ, Baldwin RL, York EJ, Stewart JM, Dyson HJ, Wright PE (1989) ^{1}H NMR studies of the solution conformations of an analogue of the C-peptide of ribonuclease A. Biochemistry 28:7059–7064
54. Ramstein J, Lavery R (1988) Energetic coupling between DNA bending and base pair opening. Proc Natl Acad Sci USA 85:7231–7235
55. Sayle RA, Milner-White EJ (1995) RasMol: biomolecular graphics for all. TIBS 20:374–376
56. Shoemaker KR, Kim PS, Brems DN, Marqusee S, York EJ, Chaiken IM, Stewart JM, Baldwin RL (1985) Nature of the charged-group effect on the stability of the C-peptide helix. Proc Natl Acad Sci USA 82:2349–2353
57. Sippl MJ, Némethy G, Scheraga HA (1984) Intermolecular potentials from crystal data. 6. Determination of empirical potentials for O-H $\cdots$ O = C hydrogen bonds from packing configurations. J Phys Chem 88:6231–6233
58. Tilton RF Jr, Dewan JC, Petsko GA (1992) Effects of temperature on protein structure and dynamics: X-ray crystallographic studies of the protein ribonuclease-A at nine different temperatures from 98 to 320 K. Biochemistry 31:2469–2481
59. Wesson L, Eisenberg D (1992) Atomic solvation parameters applied to molecular dynamics of proteins in solution. Protein Sci 1:227–235
60. Wetlaufer DB (1973) Nucleation, rapid folding, and globular intrachain regions in proteins. Proc Natl Acad Sci USA 70:697–701
61. Wilson C, Doniach S (1989) A computer model to dynamically simulate protein folding: studies with crambin. PROTEINS: Struct Funct Genet 6:193–209
62. Wilson SR, Cui W (1994) Conformation searching using simulated annealing. In: The Protein Folding Problem and Tertiary Structure Prediction. In: Lecture Notes. Birkhäuser, Basel, pp 43–70
63. Wilson SR, Cui W, Moskowitz JW, Schmidt KE (1988) Conformational analysis of flexible molecules: location of the global minimum energy conformation by the simulated annealing method. Tetrahedron Lett 29:4373–4376
64. Wychoff HW, Tsernoglou D, Hanson AW, Knox JR, Lee B, Richards FM (1970) The three-dimensional structure of ribonuclease-S. J Biol Chem 245:305–328

Monte-Carlo Simulations for Stochastic Optimization

DAVID P. MORTON, ELMIRA POPOVA
Oper. Res. and Industrial Engineering,
University Texas at Austin, Austin, USA

MSC2000: 90C15, 65C05, 65K05, 90C31, 62F12

Article Outline

Keywords

Stochastic programming; Simulation-based optimization; Monte-Carlo method

Many important real-world problems contain stochastic elements and require optimization. *Stochastic programming* and *simulation-based optimization* are two approaches used to address this issue. We do not explicitly discuss other related areas including stochastic control, stochastic dynamic programming, and Markov decision processes. We consider a stochastic optimization problem of the form

$$\text{(SP)} \quad z^* = \min_{x \in X} \mathsf{E} f(x, \xi),$$

where x is a vector of decision variables with deterministic feasible region $X \subset \mathbf{R}^d$, ξ is a random vector, and f is a real-valued function with finite expectation, $\mathsf{E} f(x, \xi)$, for all $x \in X$. We use x^* to denote an optimal solution to (SP). Note that the decision x must be made prior to observing the realization of ξ.

A wide variety of types of problems can be expressed as (SP) depending on the definitions of f and X. Two of the most commonly-used approaches are rooted in mathematical programming and in discrete-event simulation modeling.

In a two-stage stochastic linear program with recourse [6,14], X is a polyhedral set and f is defined as the optimal value of a linear program, given x and ξ, i. e.,

$$f(x, \xi) = cx + \begin{cases} \min\limits_{y \geq 0} & qy \\ \text{s.t.} & Wy = Tx + h. \end{cases} \tag{1}$$

Here, ξ is the vector of random elements from h, q, T, and W. A prototypical problem of this nature is a capacity allocation model under uncertain demand and/or capacity availabilities. x is a strategic decision allocating resources while y represents an operational recourse decision that is made *after* observing the demand and availabilities. Example applications of this type include capacity expansion planning in an electric power system [16] and in a telecommunications network [61]. The two-stage model generalizes to a more dynamic, multistage model (see, e. g., [10]) in which decisions are made, and random events unfold, over time. For multistage applications in asset-liability management see [13] and in hydro-electric scheduling see [39].

In the context of a simulation model, $f(x, \xi)$ could represent a performance measure under a design specified by x. For example, $f(x, \xi)$ might represent the number of hours in a workday that a critical machine is blocked in a queueing network model of a manufacturing system in which buffer sizes are determined by x. In another application, E.L. Plambeck et al. [53] allocate constrained processing rates to unreliable machines with buffers in a fluid serial queueing network in order to maximize steady-state throughput. In nonterminating simulations, the expectation in $\mathsf{E}f(x, \xi)$ is typically with respect to a steady-state distribution.

Note that $\mathsf{E}f(x, \xi)$ can capture objectives not usually thought of as a 'mean'. For example, if c represents random rates of return and x investment amounts, we might want to maximize the probability of exceeding a return threshold, T. We can write $\mathsf{P}(cx \geq T) = \mathsf{E}I(cx \geq T)$ where $I(\cdot)$ is the *indicator function* that takes value one if its argument is true and zero otherwise. For more on probability maximization models (and generalizations of (SP) in which X contains probabilistic constraints) see [54]. See [45] for a discussion of risk modeling in stochastic optimization.

A more general model than (SP) allows the distribution of ξ to depend on x. Some simple types of dependencies can effectively be captured in (SP) via modeling tricks, such as the x scaling random elements of T in (1). General dependencies, however, are difficult to handle. For work on decision-dependent distributions when there are a finite number of possibilities see [26,40].

Regardless of whether it is defined as the expected value of a mathematical program or as a long-run average performance measure of a discrete-event simulation model, it is usually impossible to calculate $\mathsf{E}f(x, \xi)$ exactly- even for a fixed value of x. When the dimension of the random vector ξ is relatively low, one approach is to obtain deterministic approximations of $\mathsf{E}f(x, \xi)$ using numerical quadrature or related ideas. In stochastic programming, this corresponds to generating and refining bounds on $\mathsf{E}f(x, \xi)$ within a sequential approximation algorithm [20,24,43]. For problems in which ξ is of moderate-to-high dimension and is continuous or has a large number of realizations, *Monte-Carlo simulation* is widely regarded as the method of choice for estimating $\mathsf{E}f(x, \xi)$. As a result, it is not surprising that Monte-Carlo techniques play a fundamental role in solving (SP).

In recent years (1999), considerable progress has been made in solving realistically-sized problems with a significant number of stochastic parameters and decision variables. The telecommunications model considered in [61] has 86 random point-to-point demand pairs and 89 links on which capacity may be installed. In [53] queueing networks with up to 50 nodes are studied. Each node represents a machine with random failures and has a decision variable denoting its assigned cycle time. [53] also solves a stochastic PERT (program evaluation and review technique) problem with 70 nodes and 110 stochastic arcs. The arcs model the times required to complete activities and a decision variable associated with each arc influences (parameterizes) the distribution of the random activity duration. These problems contain objectives with high-dimensional expectations and all were solved using Monte-Carlo methods.

In this article we discuss:

i) several types of Monte-Carlo-based solution procedures that can be used for solving (SP);
ii) methods for testing the quality of a candidate solution $\widehat{x} \in X$;
iii) variance reduction techniques used in stochastic optimization; and
iv) theoretical justification for using sampling.

Solution Procedures

Monte-Carlo methods for approximately solving stochastic optimization problems can typically be classified on the basis of whether the sampling is external to, or internal to, the optimization algorithm. Solution procedures of both types are driven by estimates of objective function values and/or gradients. Before turning to solution procedures we briefly discuss *gradient estimation*.

In stochastic programming, gradient (or subgradient) estimates of $\mathsf{E}f(x,\xi)$ are typically available via duality. In simulation-based optimization, the primary methods for obtaining gradient estimates are finite differences, the *likelihood ratio* (LR) method (also called the *score function method*) [29,57], and *infinitesimal perturbation analysis* (IPA) [27,35]. *Finite-difference approximations* require minimal structure, needing only estimates of $\mathsf{E}f(x,\xi)$; however, they result in solution procedures that can converge slowly. The LR method is more widely applicable than IPA, but when both apply the IPA approach tends to produce estimators with lower variance. See, for example, [28] for a discussion of these issues.

In the simplest form of 'external sampling' (also called '*sample-path optimization*' [55] and the '*stochastic counterpart*' *method* [57]) we generate independent and identically distributed (i.i.d.) replicates $\xi^1, \dots, \xi^n$ from the distribution of ξ and form the approximating problem

$$(\mathrm{SP}_n) \quad z_n^* = \min_{x\in X} \frac{1}{n}\sum_{i=1}^{n} f(x,\xi^i).$$

Even when it is possible to construct (SP_n) using i.i.d. variates, it may be preferable to use another sampling scheme in order to reduce the variance of the resulting estimators. Moreover, in nonterminating simulation models, generating i.i.d. replicates from a stationary distribution is often impossible (for exceptions see recent work on *exact sampling*, e. g., [3,22]), but under appropriate conditions we may run the simulation for a length n and replace the objective function in (SP_n) with a consistent estimate of the desired long-run average performance measure.

After constructing an instance of (SP_n) we employ a (deterministic) optimization algorithm to obtain a solution x_n^*. In the case of stochastic linear programming, (SP_n) is a large scale linear program. The cutting plane algorithm of R.M. Van Slyke and R.J-B. Wets [64], its variant with a quadratic proximal term [58], and its multistage version [7,9] are powerful tools for solving such problems. A cutting plane algorithm with a proximal term and IPA-based gradients is used in an external sampling method for solving the queueing network problem in [53]. See [8] for a recent survey of computational methods for stochastic programming instances of (SP_n).

Intuitively, we might expect solutions of (SP_n) to more accurately approximate solutions of (SP) as n increases. We discuss results supporting this in Sect. "Theoretical Justification for Sampling". In addition, after having solved (SP_n) to obtain x_n^* it would be desirable to know whether n was 'large enough'. More generally, we would like to be able to test the quality of a candidate solution (such as x_n^*). This is discussed in the next section.

We now turn to solution procedures based on internal sampling. These algorithms adapt deterministic optimization algorithms by replacing exact function and gradient evaluations with Monte-Carlo estimates. The sampling is internal because new observations of ξ are generated on an as-needed basis at each iteration of the algorithm. We briefly discuss stochastic adaptations of steepest descent and cutting plane methods.

A deterministic *steepest descent algorithm* for (SP) forms iterates $\{x^\ell\}$ using the recursion

$$x^{\ell+1} \leftarrow \Pi_X\left[x^\ell - \rho^\ell \nabla E f(x^\ell,\xi)\right].$$

Π_X performs a projection onto X and $\{\ \rho^\ell\ \}$ are steplengths. It is usually impossible to calculate $\nabla\mathsf{E}f(x,\xi)$ exactly and it must be estimated. *Stochastic approximation* (SA) and *stochastic quasigradient* (SQG) algorithms are stochastic variants of a steepest descent search. The *Keifer-Wolfowitz* SA method uses unbiased estimates of $\mathsf{E}f(x,\xi)$ to form finite-difference approximations of the gradient. The *Robbins–Monro* SA procedure requires unbiased estimates of $\nabla\mathsf{E}f(x,\xi)$. SQG methods do not require that $\mathsf{E}f(x,\xi)$ be differentiable and work under more general assumptions concerning the estimates of (sub)gradients of $\mathsf{E}f(x,\xi)$. In particular, the estimates need not be unbiased but the bias must effectively shrink to zero as the algorithm proceeds. For convergence properties of SA methods see [49] and for SQG procedures see [23].

Cutting plane methods are applicable when $\mathsf{E}f(x,\xi)$ is convex. The iterates $\{x^\ell\ \}$ are found by solving a sequence of optimization problems of the form

$$\min_{x\in X}\ \max_{\ell=1,\dots,L}\ \mathsf{E}f(x^\ell,\xi) + \nabla\mathsf{E}f(x^\ell,\xi)(x-x^\ell),$$

where L grows as the algorithm proceeds. At each iteration a first order Taylor approximation of $\mathsf{E}f(x,\xi)$, i. e., a cutting plane, is computed at the current iterate x^{ℓ} and is used to refine the piecewise-linear outer approximation of $\mathsf{E}f(x,\xi)$. The key idea is that this approximation need only be accurate in the neighborhood of an optimal solution. For stochastic linear programs, G.B. Dantzig, P.W. Glynn [15], and G. Infanger [37,38] and J.L. Higle and S. Sen [32,34] have developed Monte-Carlo-based cutting plane methods by using statistical estimates for the cut intercepts and gradients. Dantzig, Glynn, and Infanger use separate streams of observations of ξ to estimate each cut. The *stochastic decomposition* algorithm of Higle and Sen uses common random number streams to calculate each cut and employs an updating procedure to ensure that the statistical cuts are asymptotically valid (i. e., lie below $\mathsf{E}f(x,\xi)$). Relative to SA and SQG methods, cutting plane procedures avoid potentially difficult projections and, in practice, have a reputation for converging more quickly, particularly when X is high dimensional.

Grid search and optimization of *metamodels* are two common approaches to optimizing system performance in discrete-event simulation models. In grid search, X is replaced by a 'grid' of points $X_m = \{ x^1, \ldots, x^m \}$ and sample-mean estimates

$$\overline{f}_n(x) = 1/n \sum_{i=1}^{n} f(x,\xi^i)$$

are formed at each $x \in X_m$. (SP) is then approximately solved by $z_n^* = \min_{x\in X_m} \overline{f}_n(x)$ with x_n^* being the associated minimizer. Grid search is attractive because it requires minimal structure, but in implementing this procedure, we must exercise care in selecting m and n. With independent sampling at each grid point, K.B. Ensor and Glynn [21] consider the rate at which n must grow relative to m in order to achieve consistency and they also discuss the method's limiting behavior when the rate of growth is at (and slower than) the critical rate.

A metamodel can be used to approximate a more complex simulation model which, in turn, is an approximation of the real system. In such a metamodel, estimates of $\mathsf{E}f(x,\xi)$ are formed at each point in a set specified by an experimental design, and the parameters of the postulated *response surface* are fit to these observed values. The resulting function is then optimized with respect to x. For more on metamodels see, e. g., [11,47]. The review in [25] includes optimization using response surfaces, and metamodeling has also been applied in stochastic programming [5].

The grid-search and metamodel approaches are classified as external sampling procedures if the procedure is executed once. However, it may be desirable to refine the grid (or the region covered by the experimental design) in the neighborhood of promising values of x and repeat the methodology. When it is adaptively repeated in this fashion the procedure is classified as an internal sampling method.

We have not explicitly discussed approaches for when X is discrete. These range from methods for selecting the best design in simulation to those for solving stochastic integer programming models. Finally, sampling-based procedures for multistage stochastic programs have been proposed in [17].

Establishing Solution Quality

Establishing solution quality is a key concept when using an approximation scheme to solve an optimization problem. When applying Monte-Carlo techniques to (SP), the best we can expect are probabilistic quality statements. In the context of external sampling, there has been significant work on studying the behavior of solutions to (SP$_n$) for large sample sizes (see the last section). There are analogous convergence results for algorithms based on internal sampling. Such results take a number of forms but perhaps the most fundamental is to show that limit points of the sequence of solutions are, say, almost surely optimal to (SP). Next, it is desirable to have a statement regarding the rate of convergence and an associated asymptotic distribution. These consistency and limiting distribution results are aimed at justifying sampling-based methods and may be viewed as establishing solution quality. However, the approach discussed in this section centers on the question: Given a candidate solution $\widehat{x} \in X$, what can be said regarding its quality? Because candidate solutions may be obtained by internal or external sampling schemes or via another, heuristic, method, procedures that can directly test the quality of $\widehat{x}$, regardless of its origin, are very attractive.

One natural way of defining solution quality is by the optimality gap, $\mathsf{E}f(\widehat{x},\xi) - z^*$. An optimal solution has an optimality gap of zero, but in our setting we hope to make probabilistic statements such as

$$\mathsf{P}\{\mathsf{E}f(\widehat{x},\xi) - z^* \leq \epsilon\} \geq \alpha, \tag{2}$$

where ϵ is a random confidence interval width and α is a confidence level, e. g., $\alpha = 0.95$. Unfortunately, exact confidence intervals such as (2) can be difficult to obtain even in relatively simple statistical settings so we attempt to construct approximate confidence intervals

$$\mathsf{P}\{\mathsf{E}f(\widehat{x},\xi) - z^* \leq \epsilon\} \approx \alpha. \tag{3}$$

To form a *confidence interval* (3) for $\mathsf{E}f(\widehat{x},\xi) - z^*$ we estimate the mean of a gap random variable $G_n = U_n - L_n$ that is expressed as the difference between upper and lower bound estimators and satisfies $\mathsf{E}G_n \geq \mathsf{E}f(\widehat{x},\xi) - z^*$.

In many problems it is relatively straightforward to estimate the performance of a suboptimal decision $\widehat{x}$ via simulation. For example, the standard sample mean estimator, $U_n = 1/n \sum_{i=1}^n f(\widehat{x},\xi^i)$, provides an unbiased estimate of the expected cost of using decision $\widehat{x}$, i. e., $\mathsf{E}f(\widehat{x},\xi)$.

To construct a confidence interval for the optimality gap we also want an estimate of z^*. However, unbiased estimates of z^* are difficult to obtain so an estimator L_n that satisfies $\mathsf{E}L_n \leq z^*$ is used. In [51] it is shown that if the objective in (SP$_n$) is an unbiased estimate of $\mathsf{E}f(x,\xi)$ then $\mathsf{E}z_n^* \leq z^*$, i. e., z_n^* is one possible lower bound estimator L_n. Higle and Sen [33] perform a Lagrangian relaxation of a reformulation of (SP$_n$) which uses explicit 'nonanticipativity' constraints. The resulting lower bound is weaker in expectation than z_n^* but has the computational advantage that the optimization problem separates by scenario.

Once observations of G_n can be formed, we can appeal to the batch means method and use the central limit theorem [51], or a nonparametric approach [31,33], to construct approximate confidence intervals (3). Another approach to examining solution quality is to test the null hypothesis that the (generalized) Karush-Kuhn-Tucker (KKT) optimality conditions are satisfied; see [63]. Higle and Sen [31] also consider the KKT conditions but use them to derive bounds on the optimality gap.

Variance Reduction Techniques

When applying the 'crude' Monte-Carlo method to estimate $\mathsf{E}f(x,\xi)$ for fixed x, we use the standard sample mean estimator based on i.i.d. terms,

$$\frac{1}{n}\sum_{i=1}^{n} f(x,\xi^i).$$

The error associated with this estimate is proportional to

$$\left[\frac{\operatorname{var} f(x,\xi)}{n}\right]^{1/2}. \tag{4}$$

This error can be decreased by increasing the sample size. However, obtaining an additional digit of accuracy requires increasing the sample size by a factor of 100. If f is defined as the optimal value of a mathematical program or as the performance measure of a simulation model, increasing the number of evaluations of f in this fashion can be prohibitively expensive. *Variance reduction techniques* (VRTs) effectively decrease the numerator in (4) instead of increasing the denominator. Many problems for which crude Monte-Carlo would yield useless results are instead made computationally tractable via VRTs. As described in Sect. "Solution Procedures", sampling is also used to estimate $\nabla\ Ef(x,\xi)$, but for simplicity we primarily restrict our attention to VRTs for estimating $\mathsf{E}f(x,\xi)$.

Some VRTs, including *control variates* (CVs) and *importance sampling* (IS), exploit special structures of $f(x,\xi)$. Suppose that we have $\Gamma_x(\xi)$, with known mean μ_Γ, which is believed to approximate (be positively correlated with) $f(x,\xi)$. In CVs we attempt to 'subtract out' variation by generating observations of $[f(x,\xi) - \Gamma_x(\xi)] + \mu_\Gamma$, which has the same expectation as $f(x,\xi)$. (It is common to incorporate a multiplicative factor with the control variate $\Gamma_x(\xi)$ and also possible to use multiple controls.) In IS we attempt to reduce variance by generating observations of $\mu_\Gamma\ [f(x,\xi)/\ \Gamma_x(\xi)]$. In CVs observations of ξ are generated from its original distribution. However, in IS the expected value of the ratio is not the ratio of expectations and, as a result, there is a change of measure induced by Γ_x that is required to yield an unbiased estimate. Under the new IS distribution, we are more likely to sample ξ where $\Gamma_x(\xi)$ is large, i. e., scenarios that our approximation function predicts have high cost. In an IS scheme for stochastic linear programs, [15,37] use an approximation function that is separable in the components of ξ while [48] utilizes a piecewise-linear approximation. See [12] for the solution of a stochastic optimization problem to price American-style financial options using the simpler European option as a control variate. These papers report significant variance reduction in computational results.

Other VRTs exploit correlation structures in the solution methodology.*Common random numbers* (CRNs) are often used in simulation when comparing the performance of two systems. The use of CRNs has been suggested in a stochastic approximation method with finite differences where the same stream is used for the forward and backward point estimates [50]. The upper and lower bounds used to determine solution quality (see the previous section) may be viewed as two 'systems' and the use of CRNs in estimating their difference has been advocated in [34,51]. In order to reduce the error in the resulting response surface, various methods have been proposed for generating the streams of observations of ξ at each point in the experimental design. The *Schruben–Margolin* scheme [59] uses a mixture of CRNs and antithetic variates and an extension [65] also incorporates CVs.

Another group of VRTs attempts to more regularly spread the sampled observations over the support of ξ. Such techniques include stratified sampling and Latin hypercube sampling as well as quasi-Monte-Carlo techniques in which the sequence of observations is deterministic. Empirical results in [30] for two-stage stochastic linear programming compare the variance reduction obtained by stratified sampling, antithetic variates, IS, and CVs and suggest that a CV procedure performs relatively well, particularly on high-variance problems.

Theoretical Justification for Sampling

In Sect. "Solution Procedures" we formed an approximating problem for external sampling procedures by using the sample mean estimator of $\mathsf{E}f(x,\xi)$. Here we redefine (SP$_n$) as

$$(\mathrm{SP}_n) \quad z_n^* = \min_{x \in X} \mathsf{E}_n f(x,\xi),$$

with x_n^* again denoting an optimal solution. In (SP) the expected value operator E is with respect to the 'true' probability measure P while in (SP$_n$), E_n is with respect to a measure P_n that is a statistical estimate of P. If Monte-Carlo methods are used to generate i.i.d. replicates from P then P_n is the associated (random) *empirical measure*.

Since z_n^* is an estimator of z^* and x_n^* an estimator of an optimal solution to (SP), it is natural to study the behavior of these estimators for large sample sizes. For example, under what conditions do we obtain consistency and what can be said concerning rates of convergence? Positive answers to such questions provide theoretical justification for employing external Monte-Carlo sampling techniques to solve (SP).

In general, (SP$_n$) and (SP) may have multiple optimal solutions and so we cannot expect $\{x_n^*\}$ to converge. Instead, establishing consistency of x_n^* amounts to showing that the accumulation points of the sequence are almost surely optimal to (SP). If, for example, the samples are i.i.d. then by the strong law of large numbers we have $\mathsf{E}_n f(x,\xi) \to \mathsf{E}f(x,\xi)$, a.s., for all x. Unfortunately, this does not ensure that $\{x_n^*\}$ has accumulation points that are optimal to (SP) and that $z_n^* \to z^*$, a.s. [4].

The notion of *epiconvergence* plays a fundamental role in establishing consistency results for x_n^* and z_n^*; see [4]. A sequence of functions $\{\phi_n\}$ is said to epi-converge to ϕ (written $\phi_n \xrightarrow{\text{epi}} \phi$) if the *epigraphs* of ϕ_n, $\{(x,\beta)\colon \beta \geq \phi_n(x)\}$, converge to that of ϕ. Epiconvergence is weaker than classical uniform convergence. P. Kall [41] provides an excellent review of various types of convergence, their relations, and their implications for approximations of optimization models. Epiconvergence is a valuable property because of the following result:

Theorem 1 *Suppose $\phi_n \xrightarrow{epi} \phi$. If $\widehat{x}$ is an accumulation point of $\{x_n^*$, where $x_n^* \in$ argmin $\phi_n(x)$, then* $\widehat{x} \in \operatorname{argmin} \phi(x)$.

Constrained optimization is captured in this result because ϕ_n and ϕ are defined to be extended-real-valued functions that take value $+\infty$ at infeasible points. While it is possible that the sequence of optimizers $\{x_n^*\}$ has no accumulation points, this potential difficulty is avoided if the feasible region X is compact (i. e., closed and bounded).

Because of the implications of epiconvergence, there is considerable interest in determining sufficient conditions on f, P_n, and P under which $\mathsf{E}_n f(x,\xi) \overset{\text{epi}}{\to} \mathsf{E} f(x,\xi)$, a.s. Note that because $\{\mathsf{P}_n\}$ are random measures, the epiconvergence of the approximating functions is with probability one (also called *epiconsistency*). Under this hypothesis the accumulation points of $\{x_n^*\}$ are almost surely optimal to (SP); see [19].

Sufficient conditions for achieving $\mathsf{E}_n f(x,\xi) \overset{\text{epi}}{\to} \mathsf{E} f(x,\xi)$, a.s. are examined in [19,42,55], and [56]. Roughly speaking, we will obtain epiconsistency if f is sufficiently smooth, P_n converges weakly to P with probability one, and the tails of the distributions are well-behaved relative to f. See [2,60] for results when f is discontinuous.

For two-stage stochastic programming in which the recourse matrix W in (1) is deterministic and P_n is the empirical measure, [46] contains consistency results under modest assumptions. We note that is possible to develop consistency results using other (stronger) types of convergence of $\mathsf{E}_n f(x,\xi)$ to $\mathsf{E} f(x,\xi)$; see, for example, [52].

There is a large literature on consistency, stability, and rates of convergence for solutions of (SP$_n$). Much of this work may be viewed as generalizing earlier results on constrained *maximum likelihood estimation* in [1] and [36]. Under restrictive assumptions, asymptotic normality for $\sqrt{n}(z_n^* - z^*)$ and $\sqrt{n}(x_n^* - x^*)$ may be obtained, e. g., [19]. However, when inequality constraints in X play a nontrivial role we cannot, in general, expect to obtain limiting distributions that are normal [18,44,62]. See [44] for a limiting distribution for $\sqrt{n}(x_n^* - x^*)$ that is the solution of a (random) quadratic program.

See also

► Monte-Carlo Simulated Annealing in Protein Folding

References

1. Aitchison J, Silvey SD (1958) Maximum-likelihood estimation of parameters subject to restraints. Ann Math Statist 29:813–828
2. Artstein Z, Wets RJ-B (1994) Stability results for stochastic programs and sensors, allowing for discontinuous objective functions. SIAM J Optim 4:537–550
3. Asmussen S, Glynn PW, Thorisson H (1992) Stationary detection in the initial transient problem. ACM Trans Modeling and Computer Simulation 2:130–157
4. Attouch H, Wets RJ-B (1981) Approximation and convergence in nonlinear optimization. In: Mangasarian O, Meyer R, Robinson S (eds) Nonlinear Programming, vol 4. Acad Press, New York, pp 367–394
5. Bailey TG, Jensen PA, Morton DP (1999) Response surface analysis of two-stage stochastic linear programming with recourse. Naval Res Logist 46:753–778
6. Beale EML (1955) On minimizing a convex function subject to linear inequalities. J Royal Statist Soc 17B:173–184
7. Birge JR (1985) Decomposition and partitioning methods for multistage stochastic linear programs. Oper Res 33:989–1007
8. Birge JR (1997) Stochastic programming computation and applications. INFORMS J Comput 9:111–133
9. Birge JR, Donohue CJ, Holmes DF, Svintsitski OG (1996) A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs. Math Program 75:327–352
10. Birge JR, Louveaux F (1997) Introduction to stochastic programming. Springer, Berlin
11. Box GEP, Draper NR (1987) Empirical model-building and response surfaces. Wiley, New York
12. Broadie M, Glasserman P (1997) Pricing American-style options using simulation. J Econom Dynam Control 21:1323–1352
13. Cariño DR, Kent T, Meyers DH, Stacy C, Sylvanus M, Turner AL, Watanabe K, Ziemba WT (1994) The Russell-Yasuda Kasia model: An asset/liability model for a Japanese insurance company using multistage stochastic programming. Interfaces 24:29–49
14. Dantzig GB (1955) Linear programming under uncertainty. Managem Sci 1:197–206
15. Dantzig GB, Glynn PW (1990) Parallel processors for planning under uncertainty. Ann Oper Res 22:1–21
16. Dantzig GB, Glynn PW, Avriel M, Stone JC, Entriken R, Nakayama M (1989) Decomposition techniques for multi-area generation and transmission planning under uncertainty. Report Electric Power Res Inst EPRI 2940-1
17. Dempster MAH, Thompson RT (1999) EVPI-based importance sampling solution procedures for multistage stochastic linear programmes on parallel MIMD architectures. Ann Oper Res 90:161–184
18. Dupačová J (1991) On non-normal asymptotic behavior of optimal solutions for stochastic programming problems and on related problems of mathematical statistics. Kybernetika 27:38–51
19. Dupačová J, Wets RJ-B (1988) Asymptotic behavior of statistical estimators and of optimal solutions of stochastic optimization problems. Ann Statist 16:1517–1549
20. Edirisinghe C, Ziemba WT (1996) Implementing bounds-based approximations in convex-concave two-stage stochastic programming. Math Program 75:295–325
21. Ensor KB, Glynn PW (1997) Stochastic optimization via grid search. In: Yin GG, Zhang Q (eds) Mathematics of Stochas-

tic Manufacturing Systems, vol 33. Lect Appl Math Amer Math Soc, Providence, pp 89–100
22. Ensor KB, Glynn PW (2000) Simulating the maximum of a random walk. J Statist Planning Inference 85:127–135
23. Ermoliev Y (1988) Stochastic quasigradient methods. In: Ermoliev Y, Wets RJ-B (eds) Numerical Techniques for Stochastic Optimization. Springer, Berlin, pp 141–185
24. Frauendorfer K (1992) Stochastic two-stage programming. of Lecture Notes Economics and Math Systems, vol 392. Springer, Berlin
25. Fu MC (1994) Optimization via simulation: A review. Ann Oper Res 53:199–248
26. Futschik A, Pflug GCh (1997) Optimal allocation of simulation experiments in discrete stochastic optimization and approximative algorithms. Europ J Oper Res 101:245–260
27. Glasserman P (1991) Gradient estimation via perturbation analysis. Kluwer, Dordrecht
28. Glynn PW (1989) Optimization of stochastic systems via simulation. In: Proc 1989 Winter Simulation Conf, pp 90–105
29. Glynn PW (1990) Likelihood ratio gradient estimation for stochastic systems. Comm ACM 33(10):75–84
30. Higle JL (1998) Variance reduction and objective function evaluation in stochastic linear programs. INFORMS J Comput 10:236–247
31. Higle JL, Sen S (1991) Statistical verification of optimality conditions for stochastic programs with recourse. Ann Oper Res 30:215–240
32. Higle JL, Sen S (1991) Stochastic decomposition: An algorithm for two-stage linear programs with recourse. Math Oper Res 16:650–669
33. Higle JL, Sen S (1996) Duality and statistical tests of optimality for two stage stochastic programs. Math Program 75:257–275
34. Higle JL, Sen S (1996) Stochastic decomposition: A statistical method for large scale stochastic linear programming. Kluwer, Dordrecht
35. Ho YC, Cao XR (1991) Perturbation analysis of discrete event dynamic systems. Kluwer, Dordrecht
36. Huber PJ (1967) The behavior of maximum likelihood estimates under nonstandard conditions. In: Proc Fifth Berkeley Symp Math Stat Probab, pp 221–233
37. Infanger G (1992) Monte Carlo (importance) sampling within a Benders decomposition algorithm for stochastic linear programs. Ann Oper Res 39:69–95
38. Infanger G (1993) Planning under uncertainty: Solving large-scale stochastic linear programs. Sci Press Ser. Boyd & Fraser, Danvers
39. Jacobs J, Freeman G, Grygier J, Morton D, Schultz G, Staschus K, Stedinger J (1995) SOCRATES: A system for scheduling hydroelectric generation under uncertainty. Ann Oper Res 59:99–133
40. Jonsbråten TW, Wets RJ-B, Woodruff DL (1998) A class of stochastic programs with decision dependent random elements. Ann Oper Res 82:83–106
41. Kall P (1986) Approximation to optimization problems: An elementary review. Math Oper Res 11:9–18
42. Kall P (1987) On approximations and stability in stochastic programming. In: Guddat J, Jongen HTh, Kummer B, Nožička F (eds) Parametric Optimization and Related Topics. Akad Verlag, Berlin, pp 387–407
43. Kall P, Ruszczyński A, Frauendorfer K (1988) Approximation techniques in stochastic programming. In: Ermoliev Y, Wets RJ-B (eds) Numerical Techniques for Stochastic Optimization. Springer, Berlin, 33–64
44. King AJ, Rockafellar RT (1993) Asymptotic theory for solutions in statistical estimation and stochastic programming. Math Oper Res 18:148–162
45. King AJ, Takriti S, Ahmed S (1997) Issues in risk modeling for multi-stage systems. IBM Res Report RC 20993
46. King AJ, Wets RJ-B (1991) Epi-consistency of convex stochastic programs. Stochastics 34:83–91
47. Kleijnen JPC, Groenendaal WVan (1992) Simulation: A statistical perspective. Wiley, New York
48. Krishna AS (1993) Enhanced algorithms for stochastic programming. SOL Report Dept Oper Res Stanford Univ 93–8
49. Kushner HJ, Yin GG (1997) Stochastic approximation algorithms and applications. Springer, Berlin
50. L'Ecuyer P, Giroux N, Glynn PW (1994) Stochastic optimization by simulation: numerical experiments with the M/M/1 queue in steady-state. Managem Sci 40:1245–1261
51. Mak WK, Morton DP, Wood RK (1999) Monte Carlo bounding techniques for determining solution quality in stochastic programs. Oper Res Lett 24:47–56
52. Pflug GCh, Ruszczyński A, Schultz R (1998) On the Glivenko–Cantelli problem in stochastic programming: Linear recourse and extensions. Math Oper Res 23:204–220
53. Plambeck EL, Fu B-R, Robinson SM, Suri R (1996) Sample-path optimization of convex stochastic performance functions. Math Program 75:137–176
54. Prékopa A (1995) Stochastic programming. Kluwer, Dordrecht
55. Robinson SM (1996) Analysis of sample-path optimization. Math Oper Res 21:513–528
56. Robinson SM, Wets RJ-B (1987) Stability in two-stage stochastic programming. SIAM J Control Optim 25:1409–1416
57. Rubinstein RY, Shapiro A (1993) Discrete event systems: Sensitivity and stochastic optimization by the score function method. Wiley, New York
58. Ruszczyński A (1986) A regularized decomposition method for minimizing a sum of polyhedral functions. Math Program 35:309–333
59. Schruben LW, Margolin BH (1978) Pseudorandom number assignment in statistically designed simulation and distribution sampling experiments. J Amer Statist Assoc 73:504–525

60. Schultz R (1995) On structure and stability in stochastic programs with random technology matrix and complete integer recourse. Math Program 70:73–89
61. Sen S, Doverspike RD, Cosares S (1994) Network planning with random demand. Telecommunication Systems 3:11–30
62. Shapiro A (1989) Asymptotic properties of statistical estimators in stochastic programming. Ann Statist 17:841–858
63. Shapiro A, Homem-de-Mello T (1998) A simulation-based approach to two-stage stochastic programming with recourse. Math Program 81:301–325
64. Slyke RM Van, Wets RJ-B (1969) L-Shaped linear programs with applications to optimal control and stochastic programming. SIAM J Appl Math 17:638–663
65. Tew JD (1995) Simulation metamodel estimation using a combined correlation-based variance reduction technique for first and higher-order metamodels. Europ J Oper Res 87:349–367

Motzkin Transposition Theorem

MTT

ARKADI NEMIROVSKI[1], KEES ROOS[2]
[1] Fac. Industrial Engineering and Management Technion, Israel Institute Technol., Haifa, Israel
[2] Department ITS/TWI/SSOR, Delft University Technol., Delft, The Netherlands

MSC2000: 15A39, 90C05

Article Outline

Keywords

Inequality systems; Duality; Certificate; Transposition theorem

Motzkin's transposition theorem (MTT) [1] is a so-called *theorem of the alternative* (cf. ▶ Linear Optimization: Theorems of the Alternative). It deals with the question whether or not a given system of *linear inequalities* has a solution. In the most general case such a system has the form

$$\text{(S)}\quad Ax \geq a, \quad Bx > b,$$

where A and B are matrices of size $m \times n$ and $p \times n$, respectively, and where $Ax \geq a$ contains the 'larger than or equal' inequalities and $Bx > b$ the 'larger than' inequalities. Note that inequalities of the opposite type ('smaller than or equal' or 'smaller than') can be turned into the appropriate form by multiplying them by -1.

The Motzkin transposition theorem states that the system (S) has no solution if and only if at least one of the systems (T1) and (T2) has a solution, where the latter systems are given by

$$\text{(T1)}\quad \begin{cases} y^\top A + v^\top B = 0, & y^\top a + v^\top b > 0, \\ y \geq 0, & v \geq 0, \end{cases}$$

and

$$\text{(T2)}\quad \begin{cases} y^\top A + v^\top B = 0, & y^\top a + v^\top b \geq 0, \\ y \geq 0, & v \geq 0, \quad v \neq 0, \end{cases}$$

respectively.

In other words, when one has a solution of (T1) or of (T2) this solution is a *certificate* for the fact that the given system (S) is *infeasible*, i. e., has no solution.

It makes sense to formulate two most useful principles following from the theorem.

Theorem 1 (*Principle A*) *The system (S) is infeasible if and only if one can combine the inequalities in (S) in a linear fashion (i. e., multiply each inequality with a nonnegative number and add the results) to get the contradictory inequality 0 > 0 (or 0 ≥ 1).*

To see that this is exactly what the MTT says, let y and v denote nonnegative vectors of appropriate sizes. Then the inequality

$$\left(y^\top A + v^\top B\right) x \geq y^\top a + v^\top b \tag{1}$$

is a consequence of the inequalities in (S), and if the vector v is not the zero vector, then also the stronger inequality

$$\left(y^\top A + v^\top B\right) x > y^\top a + v^\top b \tag{2}$$

is a consequence of (S). The inequalities (1) and (2) have certainly solutions if $y^\top A + v^\top B \neq 0$. But if $y^\top A + v^\top B = 0$ then (1) yields a contradiction if $y^\top a + v^\top b > 0$ and (2) if $y^\top a + v^\top b \geq 0$. The first case occurs if (T1) has a solution and the second case if (T2) has a solution.

The second principle is:

Theorem 2 (*Principle B*) *If (S) is feasible, then a linear inequality is a consequence of the inequalities in (S) if and only if it can be obtained by combining, in a linear fashion, the inequalities in (S) and the trivial inequality $0 \geq -1$.*

This principle can be understood in a similar way: If (S) is feasible, then $c^\top x \geq z$ is an implied inequality if and only if

$$Ax \geq a, \; Bx > b \quad \Rightarrow \quad c^\top x \geq z,$$

which is equivalent to the system

$$Ax \geq a, \quad Bx > b, \quad -c^\top x > -z$$

being infeasible. By Principle A this happens if and only if there exist nonnegative vectors y and v and a nonnegative scalar λ such that

$$\left(y^\top A + v^\top B - \lambda c\right) x \geq y^\top a + v^\top b - \lambda z$$

is a contradictory inequality. Hence $y^\top A + v^\top B - \lambda c = 0$ and $y^\top a + v^\top b - \lambda z > 0$. Since (S) is feasible, we must have $\lambda > 0$. Without loss of generality we may assume $\lambda = 1$. Then $c = y^\top A + v^\top B$ and $z \geq y^\top a + v^\top b$. This proves the claim.

The above principles are highly nontrivial and very deep. Consider, e. g., the following system of 4 inequalities with two variables u, v:

$$-1 \leq u \leq 1,$$
$$-1 \leq v \leq 1.$$

From these inequalities it follows that

$$u^2 + v^2 \leq 2,$$

which in turn implies, by the *Cauchy inequality*, the inequality $u + v \leq 2$:

$$u + v = 1 \cdot u + 1 \cdot v \leq \sqrt{1^2 + 1^2}\sqrt{u^2 + v^2} \leq 2.$$

The concluding inequality is linear, and is a consequence of the original system, but the above derivation is 'highly nonlinear'. It is absolutely unclear a priori why the same inequality can also be obtained from the given system in a linear manner as well, as stated by Principle B. Of course, it can – it suffices to add the inequalities $u \leq 1$ and $v \leq 1$.

The MTT is one of the deepest result in the part of mathematics dealing with linear inequalities and, in fact, is logically equivalent to other deep results in this discipline. For example, it is equivalent to the duality theorem for linear optimization (cf. ► Linear Programming). To demonstrate this, consider the *linear optimization problem*

$$\text{(P)} \quad \min\left\{c^\top x : \; Ax \geq b\right\}.$$

Let z^* denote the optimal value of (P), where we take $z^* = -\infty$ if (P) is unbounded and $z^* = \infty$ if (P) is infeasible. Now, a real z is a lower bound on the optimal value of (P) if and only if $c^\top x \geq z$ is a consequence of $Ax \geq b$, or, which is the same, if and only if the system of linear inequalities

$$(\text{S}_z) \quad Ax \geq b, \quad -c^\top x > -z$$

has no solutions. By the MTT this is the case if and only if at least one of the systems

$$(\text{T}1_z) \quad \begin{cases} y^\top A - y_0 c = 0, & y^\top b - y_0 z > 0, \\ y \geq 0, & y_0 \geq 0 \end{cases}$$

and

$$(\text{T}2_z) \quad \begin{cases} y^\top A - y_0 c = 0, & y^\top b - y_0 z \geq 0, \\ y \geq 0, & y_0 > 0 \end{cases}$$

has a solution. Note that the only difference between these two systems is that $(\text{T}1_z)$ requires $y_0 \geq 0$ whereas $(\text{T}2_z)$ requires $y_0 > 0$. Also, since the system $(\text{T}2_z)$ is homogeneous, without loss of generality we may take $y_0 = 1$. Thus it follows that z is a lower bound on the optimal value of (P) if and only if one of the following two systems

$$(\text{T}1'_z) \quad y^\top A = 0, \quad y^\top b > 0, \quad y \geq 0$$

and

$$(\text{T}2'_z) \quad y^\top A = c, \quad y^\top b \geq z, \quad y \geq 0$$

has a solution. Observe that z does not appear in $(\text{T}1'_z)$. Therefore, if this system has a solution then each real z is a lower bound on the optimal value of (P), but this occurs if and only the problem (P) is infeasible. Assuming that (P) is feasible, it follows that z is a lower bound on the optimal value of (P) if and only if the system $(\text{T}2'_z)$

has a solution. Given a solution y of $(\mathrm{T2}'_z)$ any z satisfying $y^\top b \geq z$ is a lower bound and the largest lower bound provided in this way is $y^\top b$. Hence, the largest possible lower bound on the optimal value of (P) is the optimal value of the problem

$$(\mathrm{D}) \quad \max\left\{b^\top y\colon\ y^\top A = c,\ y \geq 0\right\}.$$

If the problem (P) is unbounded, i. e., if there does not exists a lower bound on the optimal value of (P), then the problem (D) must be infeasible. Otherwise the optimal value of (D) must coincide with the optimal value of (P).

The problem (D) is called the *dual problem* of the *primal problem* (P). The above findings can be summarized as follows:

> if one of the two problems (P) and (D) is unbounded then the other is infeasible; if both problems are feasible then they have both an optimal solution and the optimal values are the same.

This is the *duality theorem for linear optimization*. Note that one other case may occur, namely that both problems are infeasible. It became clear above that (P) is infeasible if and only if $(\mathrm{T1}'_z)$ has a solution, so

> the primal problem (P) is infeasible if and only if there exists a *dual ray* y, i. e., a vector y such that

$$y^\top A = 0, \quad y^\top b > 0, \quad y \geq 0. \tag{3}$$

In fact, the latter statement is equivalent to the statement that (3) and $Ax \geq b$ are *alternative systems*, which is the special case of the MTT occurring when B is vacuous and which is known as *Farkas' lemma*. (See ► Linear Optimization: Theorems of the Alternative and ► Farkas Lemma.) In just the same way it can be derived from a variant of Farkas' lemma that:

> the dual problem (D) is infeasible if and only if there exists a *primal ray* x, i. e., a vector x such that

$$Ax \geq 0, \quad c^\top x < 0. \tag{4}$$

It has been shown above that the MTT implies the duality theorem for linear optimization. The converse is also true: Assuming the duality theorem for linear optimization, the MTT easily can be proved, showing that the two results are logically equivalent. This goes in two steps. Assuming the duality theorem for linear optimization, first one derives Farkas' lemma and then it is shown that the MTT follows. To derive Farkas' lemma, consider the problem

$$\min\left\{0^\top x\colon\ Ax \geq b\right\}.$$

Clearly, the system $Ax \geq b$ has a solution if and only if the optimal value of this problem is zero. By the duality theorem this holds if and only if the optimal value of the dual problem

$$\max\left\{b^\top y\colon\ y^\top A = 0,\ y \geq 0\right\}$$

is also zero. This holds if and only

$$y^\top A = 0,\ y \geq 0 \quad \Rightarrow \quad b^\top y \leq 0,$$

which is true if and only if the system

$$y^\top A = 0, \quad y \geq 0, \quad b^\top y > 0$$

has no solution, proving Farkas' lemma.

To prove the MTT, one derives from Farkas' lemma that the 'weaker' system

$$(\mathrm{S1}) \quad Ax \geq a, \quad Bx \geq b$$

is infeasible if and only if the system (T1) has a solution. If (S1) is feasible then one easily verifies that (S) has no solution if and only if the optimal value of the problem

$$(\mathrm{P1}) \quad \min\{v\colon\ Ax \geq a,\ Bx + ve \geq b\}$$

is a nonnegative real. Here e denotes the all-one vector. Since (P1) is feasible and below bounded, by the duality theorem this happens if and only if the optimal value of the dual problem

$$(\mathrm{D1}) \quad \max\left\{a^\top y + b^\top v\colon\ \begin{array}{c} y^\top A + v^\top B = 0, \\ e^\top v = 1, \\ y \geq 0, \quad v \geq 0 \end{array}\right\}$$

is a nonnegative real and, finally, this occurs if and only if (T2) has a solution. Thus it has been shown that the MTT is logically equivalent to the duality theorem for linear optimization.

So far the issue of how to prove the MTT has not been touched. One possible approach is to prove the duality theorem for linear optimization and then derive the MTT in the above described way. This approach is now quite popular in text books. For a recent example see, e. g., [2]. The easiest way for a direct proof is to prove first the Farkas' lemma and then derive the MTT from this lemma. The latter step uses the easy to verify statement that (S) has no solution if and only if the system

$$\begin{aligned} &Ax - ta \geq 0,\\ &Bx - tb - se \geq 0,\\ &t - s \geq 0,\\ &-s < 0 \end{aligned}$$

has no solution. Application of a suitable variant of Farkas' lemma to this system yields the MTT. Farkas' lemma and its proof have a rich history; for a nice and detailed survey one might consult [3].

See also

- ► Farkas Lemma
- ► Linear Optimization: Theorems of the Alternative
- ► Linear Programming
- ► Minimum Concave Transportation Problems
- ► Multi-index Transportation Problems
- ► Stochastic Transportation and Location Problems
- ► Tucker Homogeneous Systems of Linear Relations

References

1. Motzkin TS (1936) Beiträge zur Theorie der Linearen Ungleichungen. PhD Thesis Azriel, Jerusalem
2. Padberg M (1995) Linear optimization and extensions. of Algorithms and Combinatorics, vol 12. Springer, Berlin
3. Schrijver A (1986) Theory of linear and integer programming. Wiley, New York

Multi-Class Data Classification via Mixed-Integer Optimization

Metin Türkay, Fadime Üney Yüksektepe
Department of Industrial Engineering,
Koç University, Istanbul, Turkey

Article Outline

Introduction

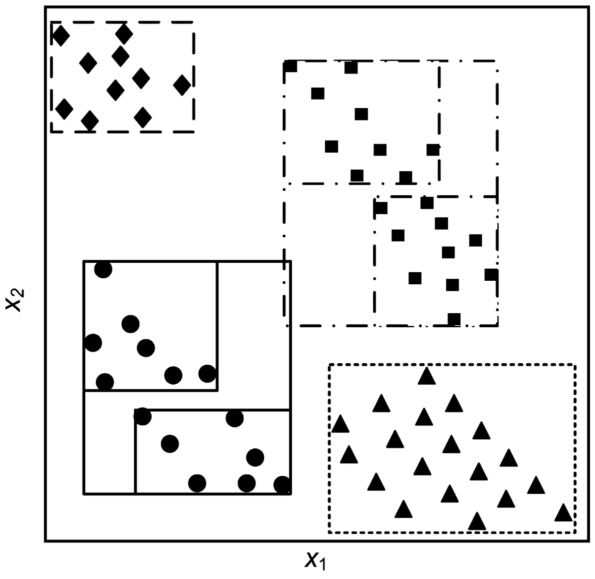
Data classification is a supervised learning strategy that analyzes the organization and categorization of data in distinct classes [14]. Generally, a training set, in which all objects are already associated with known class labels, is used by classification methods. The data classification algorithm works on this set by using input attributes and builds a model to classify new objects. In other words, the algorithm predicts output attribute values. Output attribute of the developed model is categorical [4]. There are many applications of data classification in finance [6,14], health care [14], sports [14], engineering [10,14] and science [10]. Data classification is an important problem that has applications in a diverse set of areas ranging from finance to bioinformatics.

A broad range of methods exists for data classification problem including Decision Tree Induction [14], Bayesian Classifier [14], Neural Networks (NN) [10], Support Vector Machines (SVM) [10] and Mathematical Programming (MP) [1]. An overall view of classification methods is published by Weiss and Kulikowski [21]. A neural network is a data structure that attempts to simulate the behavior of neurons in a biological brain [14]. A major shortcoming of the neural network approach is a lack of explanation of the constructed model. The possibility of obtaining a nonconvergent solution due to the wrong choice of initial weights and the possibility of resulting in a non-optimal solution due to the local minima problem are important handicaps of neural network-based methods. SVM approach operates by finding a hyper surface that will split the classes so that the distance between the hyper surface and the nearest of the points in the groups has the largest value [19]. The main goal is to generate a separating hyper surface which maximizes the margin and produces good generalization ability [10]. In recent years, SVM has been considered one of the most efficient methods for two-class classification problems [5]. SVM method has two important drawbacks in multi-class classification problems; a combination of SVM has to be used in order to solve the multi-class classification problems and some approximation algorithms are used in order to reduce the computational time for SVM while learning the large scale of data.

There have been numerous attempts to solve classification problems using mathematical programming. A survey of classification methods using mathematical programming is published by Joachimsthaler and Stam [11]. The mathematical programming approach to data classification was first introduced in early 1980's. Since then, numerous mathematical programming models have appeared in the literature. As an extension of complement to these, Erenguc and Koehler provide a comprehensive review [7]. Many distinct mathematical programming methods with different objective functions are developed in the literature. These include; minimizing the maximum exterior deviation, minimizing the weighted sum of exterior deviations, minimizing a measure of exterior deviations while maximizing a measure of interior deviations, minimizing the number of misclassifications, and minimizing a generalized distance measure. Most of these methods modeled data classification as linear programming (LP) problems to optimize a distance function. Contrary to LP problems, mixed-integer linear programming (MILP) problems that minimize the misclassifications on the design data set are also widely studied [7]. Mathematical programming methods have certain advantages over the parametric ones. For instance, they are free from parametric assumptions and weights to be adjusted. Moreover, varied objectives and more complex problem formulations can easily be accommodated. On the other hand, obtaining a solution without any discriminating power, unbounded solutions and excessive computational effort requirement are some of the problems in mathematical programming based methods. Koehler [12] surveys the potential problems in mathematical programming formulations. There have been several attempts to formulate data classification problems as MILP problems [2,8,13,15]. Since MILP methods suffer from computational difficulties, the efforts are mainly focused on efficient solutions for two-group supervised classification problems. Although ways to solve a multi-class data classification problem exist by means of solving several two-group problems, such approaches also have drawbacks including computational complexity resulting in long computational times [16].

MILP Formulation

The objective in data classification is to assign data points that are described by several attributes into a predefined number of classes. The use of hyper-boxes for defining boundaries of the sets that include all or some of the points in that set as shown in Fig. 1 can be very accurate on multi-class problems. If it is necessary, more than one hyper-box could be used in order to represent a class as shown in Fig. 1. When the classes that are indicated by square and circle data points are both represented by a single hyper-box respectively, the boundaries of these hyper-boxes will overlap. Thus, two boxes are constructed in order to eliminate this overlapping. A very important consideration in using hyper-boxes is the number of boxes used to define a class. If the total number of hyper-boxes is equal to the number of classes, then the data classification is very efficient. On the other hand; if there are as many hyper-boxes of a class as the number of data points in a class, then the data classification is inefficient.



Multi-Class Data Classification via Mixed-Integer Optimization, Figure 1
Schematic representation of multi-class data classification using hyper-boxes

The data classification problem based on this new idea is built in two parts: training and testing. Determination of the characteristics of the data points that belong to a certain class and differentiation them from the data points that belong to other classes are the targets done during the training part. Thus, boundaries of the classes are formed by the construction of hyper-boxes in the training step. After the distinguishing characteristics of the classes are determined, then the effectiveness of the classification must be tested. Predictive accuracy of the developed model is performed on a test data set during the test part.

Training Problem Formulation

Training part studies are performed on a training data set composed of a number of instances *i*. The data points are represented by the parameter a_{im} that denotes the value of attribute m for the instance *i*. The class *k* that the data point *i* belongs to are given by the set D_{ik}. Each existing hyper-box *l* encloses a number of data points belonging to the class *k*. Moreover, bounds *n* (*lower, upper)* of each hyper-box is determined by solving the training problem.

Given these parameters and the sets, the following variables are sufficient to model the data classification problem with hyper-boxes. The binary variable yb_l is indicates whether the box *l* is used or not. The position (inside or outside) of the data point *i* with regard to box *l* is represented by ypb_{il}. The assigned class *k* of box *l* and data point *i* is symbolized by ybc_{lk} and ypc_{ik}, respectively. If the data point *i* is within the bound *n* with respect to attribute *m* of box *l*, then the binary variable $ypbn_{ilmn}$ takes the value of 1, otherwise 0. Similarly, $ypbm_{ilm}$ indicates whether the data point *i* is within the bounds of attribute *m* of box *l* or not. Finally, yp_{ik} indicate the misclassification of data points. In order to define the boundaries of hyper-boxes, two continuous variables are required: X_{lmn} is the one that models bounds *n* for box *l* on attribute *m*. Correspondingly, bounds *n* for box *l* of class *k* on attribute *m* are defined with the continuous variable $XD_{l,k,m,n}$.

The following MILP problem models the training part of data classification method using hyper-boxes:

$$\min z = \sum_i \sum_k yp_{ik} + c \sum_l yb_l \tag{1}$$

subject to

$$XD_{lkmn} \le a_{im}\, ypb_{il} \quad \forall i,k,l,m,n \,|\, n = lo \tag{2}$$

$$XD_{lkmn} \ge a_{im}\, ypb_{il} \quad \forall i,k,l,m,n \,|\, n = up \tag{3}$$

$$XD_{lkmn} \le Q\, ybc_{lk} \quad \forall k,l,m,n \tag{4}$$

$$\sum_k XD_{lkmn} = X_{lmn} \quad \forall l,m,n \tag{5}$$

$$ypbn_{ilmn} \ge (1/Q)(X_{lmn} - a_{im}) \quad \forall i,l,m,n \,|\, n = up \tag{6}$$

$$ypbn_{ilmn} \ge (1/Q)(a_{im} - X_{lmn}) \quad \forall i,l,m,n \,|\, n = lo \tag{7}$$

$$\sum_l ypb_{il} = 1 \quad \forall i \tag{8}$$

$$\sum_k ypc_{ik} = 1 \quad \forall i \tag{9}$$

$$\sum_l ypb_{il} = \sum_k ypc_{ik} \quad \forall i \tag{10}$$

$$\sum_k ybc_{lk} \le yb_l \quad \forall l \tag{11}$$

$$ybc_{lk} - \sum_i ypb_{il} \le 0 \quad \forall l,k \tag{12}$$

$$ybc_{lk} - \sum_i ypc_{ik} \le 0 \quad \forall l,k \tag{13}$$

$$\sum_n ypbn_{ilmn} - ypbm_{ilm} \le \mathrm{N} - 1 \quad \forall i,l,m \tag{14}$$

$$\sum_m ypbm_{ilm} - ypb_{il} \le \mathrm{M} - 1 \quad \forall i,l \tag{15}$$

$$ypc_{ik} - yp_{ik} \le 0 \quad \forall i, k \notin D_{ik} \tag{16}$$

$$\begin{aligned} X_{lmn}, XD_{lkmn} \ge 0\,, yb_l, ybc_{lk}, ypb_{il}, ypc_{ik}, \\ ypbn_{ilmn}, ypbm_{ilm}, yp_{ik} \in \{0,1\} \end{aligned} \tag{17}$$

The objective function of the MILP problem (Eq. (1)) is to minimize the misclassifications in the data set with the minimum number of hyper-boxes. In order to eliminate unnecessary use of hyper-boxes, the unnecessary existence of a box is penalized with a small scalar *c* in the objective function. The lower and upper bounds of the boxes are given in Eqs. (2) and (3), respectively. The lower and upper bounds for the hyper-boxes are determined by the data points that are enclosed within the hyper-box. Eq. (4) enforces the bounds of hyper-boxes exist if and only if this hyper-box is assigned to a class. Eq. (5) is used to relate the two continuous variables that represent the bounds of the hyper-boxes. The position of a data point with respect to the bounds on attribute m for a hyper-box is given in Eqs. (6) and (7). The binary variable $ypbn_{ilmn}$ helps to identify whether the data point i is within the hyper-box l. Two constraints, one for the lower bound and one for the upper bound, are needed for this purpose (Eqs. (6) and (7)). Since these constraints establish a relation between continuous and binary variables, an arbitrarily large parameter, Q, is included in these constraints. The Eqs. (8) and (9) state that every data point must be assigned to a single hyper-box, l, and a single class, k, respectively. The equivalence between Eqs. (8) and (9) is given in Eq. (10); indicating that if there is a data point in the class k, then there must be a hyper-box l to represent the class k and vice versa. The existence of a hyper-box implies the assignment of that hyper-box to a class as shown in Eq. (11). If a class is represented by a hyper-box, there must be at least one data point within that hyper-box as in Eq. (12). In the same manner, if a hyper-box represents a class, there must be at least a data point within that class as given in Eq. (13). The Eq. (14) represents the condition of a data point being within the bounds of a box in attribute m. If a data point is within the bounds of all attributes of a box, then it must be in the box as shown in Eq. (15). When a data point is assigned to a class that it is not a member of, a penalty applies as indicated in Eq. (16). Finally, last constraint gives non-negativity and integrality of decision variables. By using this MILP formulation, a training set can be studied and the bounds of the classes are determined for a data classification problem.

Testing Problem Formulation

The testing problem for multi-class data classification using hyper-boxes is straight forward. If a new data point whose membership to a class is not known arrives, it is necessary to assign this data point to one of the classes. There are three possibilities for a new data point when determining its class:

i. the new data point is within the boundaries of a single hyper-box,
ii. the new data point is within the boundaries of more than one hyper-box,
iii. the new data point is not enclosed in any of the hyper-boxes determined in the training problem.

When the first possibility is realized for the new data point, the classification is made by directly assigning this data to the class that was represented by the hyper-box enclosing the data point. Since eliminating the shared areas between the constructed hyper-boxes introduces new constraints into the training problem that makes it computationally very difficult to be solved, there exists a possibility for a new data point to be within the boundaries of more than one hyper-box. In that case, the data point is assigned to the classes of the hyper-boxes that enclose this specific data point. The proportion of the number of correct classes to the number of total assigned classes to that data point determines the effect of that data point to the accuracy of the model. In the case when the third possibility applies, the assignment of the new data point to a class requires some analysis. If the data point is within the lower and upper bounds of all but not one of the attributes (i. e., m') defining the box, then the shortest distance between the new point and the hyper-box is calculated using the minimum distance between hyper-planes defining the hyper-box and the new data point. The minimum distance between the new data point j and the hyper-box is calculated using Eq. (18) considering the fact that the minimum distance is given by the normal of the hyper-plane.

$$\min_{l,m,n} \left\{ \left| \left(a_{jm} - X_{lmn} \right) \right| \right\} \tag{18}$$

When the data point is between the bounds of smaller than or equal to M-2 attributes, then the smallest distance between the point and the hyper-box is obtained by calculating the minimum distance between edges of the hyper-box and the new point. An edge is a finite segment consists of the points of a line that are between two extreme points X_{lmn} and $X_{lm'n}$. The data point j is represented by the vector $\vec{A}_j$ which is composed of a_{jm} values and $\vec{P0}_{lmn}$ and $\vec{P1}_{lmn}$ are the vector forms of two extreme points. The minimum distance between the new data point j and one of the segments of the hyper-box determined by two extreme points is calculated using Eq. (25) where $(\cdot)$ indicates the dot product of the matrices in Eq. (22) and (23).

$$\vec{W}_{jlmn} = \vec{A}_j - \vec{P0}_{lmn} \tag{19}$$

$$\vec{V}_{jlmn} = \vec{P1}_{lmn} - \vec{P0}_{lmn} \tag{20}$$

$$C1_{jlmn} = \frac{(W_{jlmn} \cdot V_{jlmn})}{\|W_{jlmn}\| \, \|V_{jlmn}\|} \tag{21}$$

$$C2_{jlmn} = \frac{(V_{jlmn} \cdot V_{jlmn})}{\|V_{jlmn}\| \, \|V_{jlmn}\|} \tag{22}$$

$$b_{jlmn} = \frac{C1_{jlmn}}{C2_{jlmn}} \tag{23}$$

$$Pb_{jlmn} = P0_{jlmn} + b_{jlmn} V_{jlmn} \tag{24}$$

$$\min_{l,n} \left\{ \sqrt{\sum_m (a_{jm} - p_{b_{jlmn}})^2} \right\} \tag{25}$$

When data point is not within the lower and upper bounds of any attributes defining the box, then the shortest distance between the new point and the hyper-box is calculated using the minimum distance between extreme points of the hyper-box and the new data. The minimum distance between the new data point j and one of the extreme points of the hyper-box is calculated using Eq. (26).

$$\min_{l,n} \left\{ \sqrt{\sum_m (a_{jm} - X_{lmn})^2} \right\} \tag{26}$$

The following algorithm assign a new data point j with attribute values a_{jm} to class k:

Step 0: Initialize $inAtt(l,m) = 0$.

Step 1: For each l and m, if

$$X_{lmn} \le a_{jm} \le X_{lmn'} \quad \forall n = lo, n' = up \tag{27}$$

Set $inAtt(l,m) = inAtt(l,m) + 1$.

Step 2: If $inAtt(l,m) = M$, then go to Step 3. Otherwise, continue. If $inAtt(l,m) \le M - 1$, then go to Step 4.

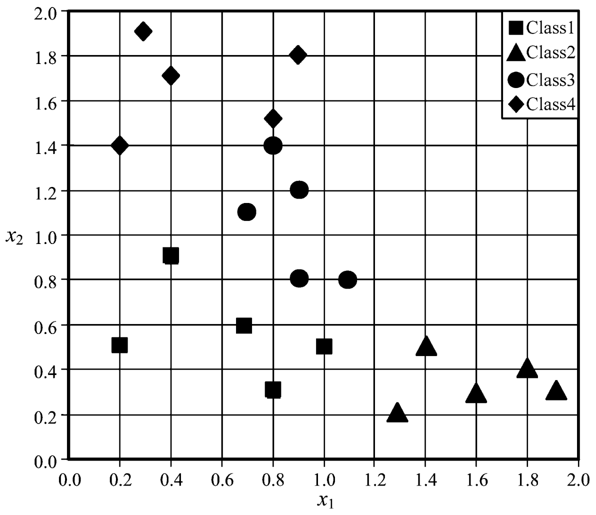
Step 3: Assign the new data point to class k where ybc_{lk} is equal to 1 for the hyper-box in Step 2. Stop.

Step 4: Calculate the minimum given by Eq. (18) and set the minimum as *min1(l)*. Calculate the minimum given by Eq. (25) and set the minimum as *min2(l)*. Calculate the minimum given by Eq. (26) and set the minimum as *min3(l)*. Select the minimum between *min1(l)*, *min2(l)* and *min3(l)* to determine the hyper-box l that is closest to the new data point j. Assign the new data point to class k where ybc_{lk} is equal to 1 for the hyper-box l. Stop.

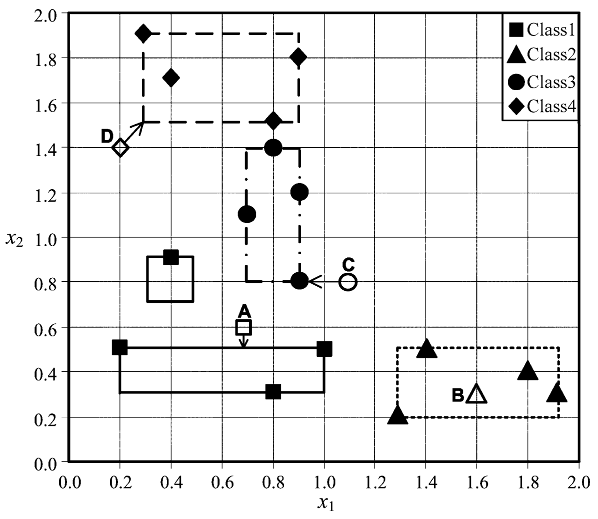
Application

We applied the mathematical programming method on a set of 16 data points in 4 different classes given in Fig. 2. The data points can be represented by two attributes, 1 and 2.

There are a total of 20 data points; 16 of these points were used in training and 4 of them used in testing. The training problem classified the data into 4 four classes using 5 hyper-boxes as shown in Fig. 3. It is interesting to note that Class1 requires two hyper-boxes while the other classes are represented with a single hyper-box only. The reason for having two hyper-boxes for Class1 is due to the fact that a single hyper-box for this



Multi-Class Data Classification via Mixed-Integer Optimization, Figure 2
Data points in the illustrative example and their graphical representation



Multi-Class Data Classification via Mixed-Integer Optimization, Figure 3
Hyper-boxes that classify the data points in the illustrative example

class would include one of the data points that belong to Class3. In order to eliminate inconsistencies in training data set, the method included one more box for Class1.

After the training is successfully completed, the test data is processed to assign them to hyper-boxes that classify the data perfectly. The assignment of the test data point B to Class2 is straightforward since it is included in the hyper-box that classifies Class2 (i. e., *inAtt(l,m)* = *M* for this data point). The test data in Class1 is assigned to one of the hyper-boxes that classify Class1. Similarly, the test data in Class3 is also assigned to the hyper-box that classifies Class3. Since the test data in these classes are included within the bounds of one of the two attributes, the minimum distance is calculated as the normal to the closest hyper-plane to these data points. In the case of data point that belongs to Class4, it is assigned to its correct class since the closest extreme point of a hyper-box classifies Class4. This extreme point of the hyper-box 5 classifying Class4 is given by $(X_{5,1,lo}, X_{5,2,lo})$. The test problem also classified the data points with 100% accuracy as shown in Fig. 3.

This illustrative example is also tested by different data classification models existing in the literature in order to compare the results and to measure the performance of the proposed model. Table 1 shows the examined models and their outcomes for this small illustrative example.

Multi-Class Data Classification via Mixed-Integer Optimization, Table 1
Comparison of different classification models for the illustrative example

Classification Model	Prediction Accuracy	Misclassified Sample(s)
Neural Networks[a]	75%	A
Support Vector Machines[b]	75%	D
Bayesian Classifier[c]	75%	C
K-nearest Neighbor Classifier[c]	75%	A
Statistical Regression Classifiers[c]	75%	C
Decision Tree Classifier[c]	50%	A, C
MILP approach	100%	–

[a] iDA implementation in MS Excel [9] [b] SVM implementation in Matlab [3] [c] WEKA [20]

Neural Networks, Support Vector Machines, Bayesian, K-nearest Neighbor and Statistical Regression classifiers have only one misclassified instance which leads to 75% accuracy value as shown in Table 1. Neural Networks and K-nearest Neighbor classifier predicts the class of test sample A as Class3. Support Vector Machine method misclassifies test sample D and assigns it to Class1 while Bayesian and Statistical Regression classifier classifies test sample C as belonging to Class2. On the other hand, Decision Tree classifier gives the lowest accuracy value (50%) with two misclassifications. Sample A and sample C is classified as Class3 and Class2, respectively. Consequently, MILP approach in this thesis classifies all of the test samples accurately and achieves 100% accuracy. As a result, the MILP approach performs better than other data classification methods that are listed in Table 1 for the illustrative example. The accuracy of the MILP approach is tested on IRIS dataset and protein folding type dataset. The results indicate that the MILP approach has better accuracy than other methods on these datasets [17,18].

Conclusion

Multi-class data classification problem can be very effectively modeled as an MILP problem. One of the most important characteristics of the MILP approach is allowing the use of hyper-boxes for defining the boundaries of the classes that enclose all or some of the points in that set. In other words, if necessary, more than one hyper-box is constructed for a specific class through the training part studies. Moreover, well-construction of the boundaries of each class provides the lack of misclassifications in the training set and indirectly improves the accuracy of the model. The model does not require the underlying distribution of the training data set and learns from the training set in a reasonable time. With only one parameter (c: the penalty parameter to minimize the total number of hyper-boxes), the suggested model is simple and very effective. Furthermore, the proposed model can be used for both binary and multi-class data classification problems without any modifications. Hence, the performance of the model does not depend on the class related changes.

References

1. Adem J, Gochet W (2006) Mathematical programming based heuristics for improving LP-generated classifiers for the multi-class supervised classification problem. Eur J Oper Res 168:181–199
2. Bajgier SM, Hill AV (1982) An experimental comparison of statistical and linear programming approaches to the discriminant problem. Decis Sci 13:604–618
3. Cawley G (2000) Matlab Support Vector Machine Toolbox. School of Information Systems, University of East Anglia
4. Chen MS, Han J, Yu PS (1996) Data Mining: An overview from a database perspective. IEEE Trans Knowl Data Eng 8:866–883
5. Cortes C, Vapnik V (1995) Support vector network. Mach Learn 20:273–297
6. Edelstein H (2003) Building Profitable Customer Relationships with Data Mining. Two Crows Corporation, Maryland
7. Erenguc SS, Koehler GJ (1990) Survey of mathematical programming models and experimental results for linear discriminant analysis. Manag Decis Econ 11:215–225
8. Gehrlein WV (1986) General mathematical programming formulations for the statistical classification problem. Oper Res Lett 5(6):299–304
9. iData Analyzer, Version 2.0, Information Acumen Corporation.
10. Jagota A (2000) Data Analysis and Classification for Bioinformatics, Bay Press, New York
11. Joachimsthaler EA, Stam A (1990) Mathematical programming approaches for the classification problem in two-group discriminant analysis. Multivar Behav Res 25:427–454
12. Koehler GJ (1990) Considerations for mathematical programming models in discriminant analysis. Manag Decis Econ 11:227–234
13. Littschwager JM, Wang C (1978) Integer programming solution of a classification problem. Manag Sci 24(14):1515–1525
14. Roiger RJ, Geatz MW (2003) Data Mining – A Tutorial Based Primer. Addison Wesley Press, Boston
15. Stam A, Joachimsthaler EA (1990) A comparison of a robust mixed-integer approach to existing methods for establishing classification rules for the discriminant problem. Eur J Oper Res 46(1):113–122
16. Tax D, Duin R (2002) Using two-class classifiers for multi class classification, Proc 16th Int Conference Pattern Recogn, Quebec City, Canada, vol II, IEEE Computers Society Press, Los Alamitos, pp 124–127
17. Turkay M, Uney F, Yilmaz O (2005) Prediction of Folding type of Proteins Using Mixed-Integer Linear Programming. In: Puigjaner L, Espuna A (eds) Computer-Aided Chem. Eng. vol 20A: ESCAPE-15. Elsevier, Amsterdam, pp 523–528
18. Uney F, Turkay M (2006) A Mixed-Integer Programming Approach to Multi-Class Data Classification Problem. Eur J Oper Res 173(3):910–920
19. Vapnik VN (1998) Statistical Learning Theory. Wiley, New York
20. WEKA (Waikato Environment for Knowledge Analysis) (199–2005) Version 3.4.5, University of Waikato, New Zealand
21. Weiss SM, Kulikowski CA (1991) Computer systems that learn: classification and prediction methods from statistics, neural networks, machine learning and expert systems. Morgan Kaufmann, San Mateo, CA

Multicommodity Flow Problems

CYNTHIA BARNHART[1], NIRANJAN KRISHNAN[1], PAMELA H. VANCE[2]
[1] Center for Transportation Studies, Massachusetts Institute Technol., Cambridge, USA
[2] Goizueta Business School, Emory University, Atlanta, USA

MSC2000: 90C35

Article Outline

Keywords

Multicommodity network flows; Column generation; Decomposition; Transportation

Linear multicommodity flow problems (MCF) are linear programs (LPs) that can be characterized by a set of commodities and an underlying network. A *commodity* is a good that must be transported from one or more origin nodes to one or more destination nodes in the network. In practice these commodities might be telephone calls in a telecommunications network, packages in a distribution network, or airplanes in an airline flight network. Each commodity has a unique set of characteristics and the commodities are not interchangeable. That is, you cannot satisfy demand for one commodity with another commodity. The objective of the MCF problem is to flow the commodities through the network at minimum cost without exceeding arc capacities. A comprehensive survey of linear multicommodity flow models and solution procedures are presented in [2].

Integer multicommodity flow (IMCF) problems, a constrained version of the linear multicommodity flow problem in which flow of a commodity (specified in this case by an origin-destination pair) may use only one path from origin to destination.

MCF and IMCF problems are prevalent in a number of application contexts, including transportation, communication and production.

MCF Example Applications

- *Routing* vehicles in traffic networks (*dynamic traffic assignment*). This involves the determination of minimum delay routes for vehicles from their origins to their respective destinations over the traffic network. The allowable congestion levels determine the arc capacities. Alternatively, there are no capacities but the cost on an arc is a function of the amount of flow on the arc. In the former case, the objective function is linear while in the latter it is nonlinear.
- *Distribution systems planning*. In this problem there are different products (or, commodities) produced at several plants with known production capacities. Each commodity has a certain demand in each customer zone. The demand is satisfied by shipping via regional distribution centers with finite storage capacities. A.M. Geoffrion and G.W. Graves [28] model this problem of routing the commodities from the manufacturing plants to the customer zones through the distribution centers as a MCF problem.
- *Import* and *export models*. One of the factors that may affect export is handling capacity at ports. D. Barnett, J. Binkley and B. McCarl [8] use a MCF model to analyze the effect of US port capacities on the export of wheat, corn and soybean.
- Optimization of *freight operations*. T. Crainic, J.A. Ferland and J.M. Rousseau [20] develop a MCF-based routing and scheduling optimization model that considers the planning issues for the railroad industry. More recently, H.N. Newton [48] and C. Barnhart, H. Jin and P.H. Vance [13] study the railroad blocking problem using multicommodity based formulations.
- Freight Assignment in the *Less-than-Truckload* (LTL) industry. An LTL carrier has to consolidate many shipments to make economic use of the vehicles. This requires the establishment of a large number of terminals to sort freight. Trucking companies use forecasted demands to define routes for each vehicle to carry freight to and from the terminals. Once the routes are fixed, the problem is to deliver all the shipments with minimum total service time or cost. This problem is formulated as a MCF problem in [17] and [24].
- *Express Shipment Delivery*. D. Kim [40] models the shipment delivery problem faced by express carriers like Federal Express, United States Postal Service, United Parcel Service, etc. as a MCF problem on a network in space and time.
- Routing messages in a *telecommunications* or computer network. The network consists of transmission lines. Each message request is a commodity. The problem is to route the messages from origins to the respective destinations at a minimum cost. T.L. Magnanti et al. [42] and others provide MCF-based formulations for this problem.

- Long-term *hydro-generation* optimization. The task in this case is to determine the amount of hydro-generation at a reservoir in an interval of time, that minimizes the expected cost of power generation over a period of time, divided into several intervals. N. Nabonna [47] showed that this problem can be modeled as a MCF problem with inflows given as probabilistic density functions.
- *Forest management*. For each planning period, forest managers have to make decisions concerning the land areas to be harvested, the volume of timber to be harvested from these areas, the land areas to be developed for recreation and the road network to be built and maintained in order to support both the timber haulers and recreationists. This problem has been formulated as a MCF problem in [33].
- *Street planning*. L.R. Foulds [26] introduced this problem and modeled it as a MCF problem. The objective is to identify a set of two-way streets such that making these streets one-way minimizes the total congestion cost in the network.
- *Spatial price equilibrium* (SPE) problem. This problem requires modeling consumer flows within a general network. The SPE problem determines the optimum levels of production and consumption at each market and the optimal flows satisfy the equilibrium property. R.S. Segall [59] models and solves the SPE problem as a MCF problem.

For a more comprehensive description of MCF applications, see [2,37,57].

IMCF Example Applications

- *Airline fleet assignment*. Given a time table of flight arrivals and departures, the expected demand on the flights and a set of aircraft, the objective is to arrive at a minimum cost assignment of aircraft to the flights. This problem has been extensively studied in [1,31].
- *Airline crew scheduling*. This problem deals with the minimum cost scheduling of crews. Factors such as hours of work limitations and Federal Aviation Administration regulations must be taken into account while solving the problem. For an in-depth study see [5,14].
- *Airline maintenance routing problems* require that single aircraft be routed such that maintenance requirements are satisfied and each flight is assigned to exactly one aircraft. This problem has been studied in [10,19,25].
- *Bandwidth packing problems* require that bandwidth be allocated in telecommunications networks to maximize total revenue. The demands, or calls, on the networks are the commodities and the objective is to route the calls from their origin to their destination. In the case of video teleconferencing, since call splitting is not allowed, each call must be routed on exactly one network path. This IMCF problem is described in [49].
- *Package flow problems*, such as those arising in express package delivery operations, require that shipments, each with a specific origin and destination, be routed over a transportation network. Each set of packages with a common origin-destination pair can be considered as a commodity and often, to facilitate operations and ensure customer satisfaction, must be assigned to a single network path. These problems are cast as IMCF problems in [12].

Formulations

Multicommodity flow problems can be modeled in a number of ways depending how one defines a commodity. There are three major options: a commodity may originate at a subset of nodes in the network and be destined for another subset of nodes, or it may originate at a single node and be destined for a subset of the nodes, or finally it may originate at a single node and be destined for a single node. K.L. Jones et al. [34] present models for each of these different cases. In the interest of space, we will only consider models for the last case. The other cases can also be modeled using variants of the models presented here.

We present two different formulations of the MCF problem: the *node-arc* or *conventional* formulation and the *path* or *column generation formulation*. The MCF is defined over the network G comprised of node set N and arc set A. MCF contains decision variables x, where x_{ij}^k is the fraction of the total quantity (denoted q^k) of commodity k assigned to arc ij. In the IMCF problem these variables are restricted to be binary. The cost of assigning commodity k in its entirety to arc ij equals q^k times the unit flow cost for arc ij, denoted c_{ij}^k. Arc ij has capacity d_{ij}, for all $ij \in A$. Node i has supply of commodity k, denoted b_i^k, equal to 1 if i is the origin node for k, equal to -1 if i is the destination node for k, and equal to 0 otherwise.

The node-arc MCF formulation is:

$$\text{minimize} \sum_{k\in K}\sum_{ij\in A} c_{ij}^k q^k x_{ij}^k \tag{1}$$

such that

$$\sum_{ij\in A} x_{ij}^k - \sum_{ji\in A} x_{ji}^k = b_i^k, \quad \forall i\in N, \forall k\in K, \tag{2}$$

$$\sum_{k\in K} q^k x_{ij}^k \le d_{ij}, \quad \forall ij\in A, \tag{3}$$

$$x_{ij}^k \ge 0, \quad \forall ij\in A, \quad \forall k\in K. \tag{4}$$

Note that without restricting generality of the problem, we model the arc flow variables x having values between 0 and 1. To do this, we scale the demand for each commodity to 1 and accordingly adjust the coefficients in the objective function (1) and in constraints (3). Also note the block-angular structure of this model. The conservation of flow constraints (2) form nonoverlapping blocks, one for each commodity. Only the arc capacity constraints (3) link the values of the flow variables of different commodities.

To contrast, the path-based or column generation MCF formulation has fewer constraints, and far more variables. Again, the underlying network G is comprised of node set N and arc set A, with q^k representing the quantity of commodity k. $P(k)$ represents the set of all origin-destination paths in G for k, for all $k\in K$. In the column generation model, the binary decision variables are denoted y_p^k, where y_p^k is the fraction of the total flow of commodity k assigned to path $p\in P(k)$. The cost of assigning commodity k in its entirety to path p equals q^k times the unit flow cost for path p, denoted c_p^k. c_p^k represents the sum of the c_{ij}^k costs for all arcs ij contained in path p. As before, arc ij has capacity d_{ij}, for all $ij\in A$. Finally, δ_{ij}^p is equal to 1 if arc ij is contained in path $p\in P(k)$, for all $k\in K$; and is equal to 0 otherwise.

The path or column generation IMCF formulation is then:

$$\text{minimize} \sum_{k\in K}\sum_{p\in P(k)} c_p^k q^k y_p^k \tag{5}$$

such that

$$\sum_{k\in K}\sum_{p\in P(k)} q^k y_p^k \delta_{ij}^p \le d_{ij}, \quad \forall ij\in A, \tag{6}$$

$$\sum_{p\in P(k)} y_p^k = 1, \quad \forall k\in K, \tag{7}$$

$$y_p^k \ge 0, \quad \forall p\in P(k), \quad \forall k\in K. \tag{8}$$

LP Solution Methods

Comprehensive surveys of the available multicommodity network flow solution techniques are provided in [6,37]. Descriptions of these approaches are also provided in [2,38].

Price-directive decomposition techniques use the path-based MCF model. To limit the number of variables considered in finding an optimal solution, *column generation* techniques are used. Further details of price-directive *decomposition* and column generation are provided in [18,22,41,45,61].

Resource-directive decomposition techniques attempt to solve MCF problems by allocating arc capacity by commodity and solving the resulting decoupled minimum cost flow problems for each commodity. Additional descriptions of this technique can be found in [27,30,35,37,39,41,52,60,61].

Computational comparisons of the performance of price- and resource-directive decomposition methods can be found in [3,4]. A. Ali, R.V. Helgason, J.L. Kennington, and H. Lall [4] report that specialized decomposition codes can be expected to run from three to ten times faster than a general linear programming package. Furthermore, A.A. Assad [7] reports that resource-directive algorithms converge quickly for small problems but are outperformed by the price-directive method for larger MCF problems.

G. Saviozzi [56] uses *subgradient* techniques on the *Lagrangian relaxation* of the bundle constraints and proposes a method of arriving at an advanced starting basis for the minimum cost multicommodity flow problem.

Partitioning methods specialize the simplex method by partitioning the current basis to exploit the underlying network structure. Experiences with primal partitioning techniques have been reported in [24,32, 36,43,51,53,54,55], among others. J.B. Rosen [53] develops a partitioning strategy for angular problems. J.K. Hartman and L.S. Lasdon [32] develop a generalized upper bounding algorithm for multicommodity network flow problems in which the special structure of the MCF problem is exploited. Their primal partitioning procedure, a specialization of the generalized upper bounding procedure developed by G.B. Dantzig and R.M. Van Slyke [21], involves the determination at each iteration of the inverse of a basis containing only one row for each saturated arc. Similarly, C.J. McCallum [44] developed a generalized upper bounding algorithm for a communications network planning problem. All of these procedures exploit the block-diagonal problem structure and perform all steps of the simplex method on a reduced working basis of dimension m, where m represents the size of set A.

Interior point methods and parallel computing techniques have also been applied to MCF problems. Interior point methods provide polynomial time algorithms for the MCF problems. The best time bound is due to P.M. Vaidya [62]. G.L. Schultz and R.R. Meyer [58] provide an interior point method with massive parallel computing to solve multicommodity flow problems.

Development of new heuristic procedures for MCF problems include the primal and dual-ascent heuristics described in [17] and [9], respectively. A. Gersht and A. Shulman [29] use a barrier-penalty method to find nearly optimal solutions for multicommodity problems, while R. Schneur [62] describes a scaling algorithm to determine nearly feasible MCF solutions.

Recently, price-directive decomposition or column generation approaches, such as those presented in [2,11,23,34] have been the most extensively used method for solving large versions of the linear MCF problem. The general idea of column generation is that optimal solutions to large LP's can be obtained without explicitly including all columns (i. e., variables) in the constraint matrix (called the *Master Problem* or MP). In fact, only a very small subset of all columns will be in an optimal solution and all other (nonbasic) columns can be ignored. In a minimization problem, this implies that all columns with positive reduced cost can be ignored. The multicommodity flow column generation strategy, then, is:

0) RMP Construction. Include a subset of columns in a restricted MP, called the *Restricted Master Problem*, or RMP;
1) RMP Solution. Solve the RMP LP;
2) Pricing Problem Solution. Use the dual variables obtained in solving the RMP to solve the pricing problem. The pricing problem either identifies one or more columns with negative reduced cost (i. e., columns that price out) or determines that no such column exists.
3) Optimality Test. If one or more columns price out, add the columns (or a subset of them) to the RMP and return to Step 1; otherwise stop, the MP is solved.

For any RMP in Step 1, let $-\pi_{ij}$ represent the nonnegative dual variables associated with constraints (6) and σ^k represent the unrestricted dual variables associated with constraints (7). Since c_p^k can be represented as $\sum_{ij\in A} c_{ij}^k\,\delta_{ij}^p$, the reduced cost of column p for commodity k, denoted $\bar{c}_p^k q^k$, is:

$$\bar{c}_p^k q^k = \sum_{ij\in A} q^k(c_{ij}^k + \pi_{ij})\delta_{ij}^p - \sigma^k, \qquad \forall p \in P(k), \forall k \in K. \quad (9)$$

For each RMP solution generated in Step 1, the pricing problem in Step 2 can be solved efficiently. Columns that price out can be identified by solving one shortest path problem for each commodity $k \in K$ over a network with arc costs equal to $c_{ij}^k + \pi_{ij}$, for each $ij \in A$. Let $p*$ represent a resulting shortest path $p*$ for commodity k. Then, if for all $k \in K$,

$$c_{p*}^k q^k \geq 0,$$

the MP is solved. Otherwise, the MP is not solved and, for each $k \in K$ with

$$c_{p*}^k q^k < 0,$$

path $p* \in P(k)$ is added to the RMP in Step 3.

IP Solution Methods

The ability to solve large MCF LP's enables the solution of large IMCF problems. Successful approaches for solving large IMCF problems use the path-based or column generation formulation of the problem. Column generation IP's can be solved to optimality using a procedure known as *branch and price*, detailed in [15,23,64]. Branch and price, a generalization of branch and bound with LP relaxations, allows column generation to be applied at each node of the branch and bound tree. Branching occurs when no columns price out to enter the basis and the LP solution does not satisfy the integrality conditions.

Applying a standard branch and bound procedure to the final restricted master problem with its existing columns will not guarantee an optimal (or feasible) solution. After the branching decision modifies RMP, it may be the case that there exists a column for MP that prices out favorably, but is not present in RMP. Therefore, to find an optimal solution we must maintain the ability to solve the pricing problem after branching. The importance of generating columns after the initial LP has been solved is demonstrated for airline crew scheduling applications in [63]. Although they were unable to find even feasible IP solutions using just the columns generated to solve the initial LP relaxation, they were able to find quality solutions using a branch and price approach for crew scheduling problems in which they generated additional columns whenever the LP bound at a node exceeded a preset IP target objective value.

The difficulty of performing column generation with branch and bound is that conventional integer programming branching on variables may not be effective because fixing variables can destroy the structure of the pricing problem. For the multicommodity flow application, a branching rule is needed that ensures that the pricing problem for the LP with the branching decisions included can be solved efficiently with a shortest path procedure. To illustrate, consider branching based on variable dichotomy in which one branch forces commodity k to be assigned to path p, i. e., $y_p^k = 1$, and the other branch does not allow commodity k to use path p, i. e., $y_p^k = 0$. The first branch is easy to enforce since no additional paths need to be generated once k is assigned to path p. The latter branch, however, cannot be enforced if the pricing problem is solved as a shortest path problem. There is no guarantee that the solution to the shortest path problem is not path p. In fact, it is likely that the shortest path for k is indeed path p. As a result, to enforce a branching decision, the pricing problem solution must be achieved using a *next shortest path procedure*. In general, for a subproblem, involving a set of a branching decisions, the pricing problem solution must be achieved using a kth shortest path procedure.

The key to developing a branch and price procedure is to identify a branching rule that eliminates the current fractional solution without compromising the tractability of the pricing problem. In general, J. Desrosiers et al [23] argue this can be achieved by basing branching rules on variables in the original formulation, and not on variables in the column generation formulation. This means that branching rules should be based on the arc flow variables x_{ij}^k from the node-arc formulation of the problem. Barnhart et al. [15] develop branching rules for a number of different master problem structures. They also survey specialized algorithms that have appeared in the literature for a broad range of applications.

M. Parker and J. Ryan [49] present a branch and price algorithm for the bandwidth packing problem. in which the objective is to choose which of a set of commodities to send in order to maximize revenue. They use a path-based formulation. Their branching scheme selects a fractional path and creates a number of new subproblems equal to the length of the path (measured in the number of arcs it contains) plus one. On one branch, the path is fixed into the solution and on each other branch, one of the arcs on the path is forbidden. To limit time spent searching the tree they use a dynamic optimality tolerance. They report the solution of 14 problems with as many as 93 commodities on networks with up to 29 nodes and 42 arcs. All but two of the instances are solved to within 95% of optimality.

K. Ziarati et al. [16] consider the problem of assigning railway locomotives to trains. They model the problem as an integer multicommodity flow problem with side constraints and solve using a Dantzig–Wolfe decomposition technique, where subproblems are formulated as constrained or unconstrained shortest path problems.

P. Raghavan and C.D. Thompson [50] illustrate the use of randomized algorithms to solve some integer multicommodity flow problems. They use randomized rounding procedures that give provably good solutions in the sense that they have a very high probability of being close to optimality.

Barnhart et al. [12] present a *branch and price and cut* algorithm for general IMCF problems where each commodity is represented by an origin-destination pair and flow volume. *Branch and cut*, another variant of branch and bound, allows valid inequalities, or cuts, to be added throughout the branch and bound tree. *Branch and price and cut* combines column and row generation to yield very strong LP relaxations at nodes of the branch and bound tree.

See also

- ▶ Auction Algorithms
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Generalized Networks
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Nonoriented Multicommodity Flow Problems
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Abara J (1989) Applying integer linear programming to the fleet assignment problem. Interfaces 19:20–28
2. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs
3. Ali AI, Barnett D, Farhangian K, Kennington JL, Patty B, Shetty B, McCarl B, Wong P (1984) Multicommodity network problems: Applications and computations. IIE Trans 16:127–134
4. Ali A, Helgason R, Kennington J, Lall H (1980) Computational comparison among three multicommodity network flow algorithms. Oper Res 28:995–1000
5. Anbil R, Gelman E, Patty B, Tanga R (1991) Recent advances in crew-pairing optimization at American Airlines. Interfaces 21:62–64
6. Assad AA (1978) Multicommoditynetwork flows - A survey. Networks 8:37–91
7. Assad AA (1980) Solving linear multicommodity flow problems. Proc IEEE Internat Conf Circuits and Computers 1, pp 157–161
8. Barnett D, Binkley J, McCarl B (1982) The effects of US port capacity constraints on national and world grain shipments. Techn Paper, Purdue Univ
9. Barnhart C (1993) Dual-ascent methods for large-scale multi-commodity flow problems. Naval Res Logist 40:305–324
10. Barnhart C, Boland NL, Clarke LW, Johnson EL, Nemhauser GL, Shenoi RG (1998) Flight string models for aircraft fleeting and routing. Transport Sci 32(3):208–220, Focused Issue on Airline Optimization
11. Barnhart C, Hane CA, Johnson EL, Sigismondi G (1995) A column generation and partitioning approach for multicommodity flow problems. Telecommunication Systems 3:239–258
12. Barnhart C, Hane CA, Vance PH (2000) Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. Oper Res 48(2):318–326
13. Barnhart C, Jin H, Vance PH (1997) Railroad blocking: A network design application. Working Paper Center Transport Stud, MIT
14. Barnhart C, Johnson EL, Anbil R, Hatay L (1994) A column generation technique for the long-haul crew-assignment problem. In: Ciriani TA, Leachman RC (eds) Optimization in Industry: Math. Programming and Optimization Techniques 2. Wiley, New York, pp 7–24
15. Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWF, Vance PH (1998) Branch-and-price: Column generation for solving huge integer programs. Oper Res 46(3):316–329
16. Barnhart C, Johnson EL, Nemhauser GL, Sigismondi G, Vance P (1993) Formulating a mixed integer programming problem to improve solvability. Oper Res 41:1013–1019
17. Barnhart C, Sheffi Y (1993) A network-based primal-dual heuristic for the solution of multi-commodity network flow problems. Transport Sci 27:102–117
18. Bazaraa MS, Jarvis JJ (1977) Linear programming and network flows. Wiley, New York
19. Clarke LW, Johnson GL, Nemhauser GL, Zhu Z (1997) The aircraft rotationproblem. Ann Oper Res: Math Industr Systems II 69:33–46
20. Crainic T, Ferland JA, Rousseau JM (1984) A tactical planning model for Rail freight transportation. Transport Sci 18:165–184
21. Dantzig GB, Van Slyke RM (1967) Generalized upper bounding techniques. J Comput Syst Sci 1:213–226
22. Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. Oper Res 8:108–111
23. Desrosiers J, Dumas Y, Solomon MM, Soumis F (1995) Time constrained routing and scheduling. In: Ball ME, Magnanti TL, Monma C, Nemhauser GL (eds) Handbook Oper. Res. and Management Sci. 8 Elsevier, Amsterdam, pp 35–139

24. Farvolden JM, Powell WB, Lustig IJ (1993) A primal partitioning solution for the arc-chain formulation of a multicommodity network flow problem. Oper Res 4(4):669–693
25. Feo TA, Bard JF (1989) Flight scheduling and maintenence base planning. Managem Sci 35:1415–1432
26. Foulds LR (1981) A multicommodity flow network design problem. Transport Res B 15:273–283
27. Geoffrion AM (1970) Primal resource-directive approaches for optimizing non-linear decomposable systems. Oper Res 18:375–403
28. Geoffrion AM, Graves GW (1974) Multicommodity distribution systems design by Bender's decomposition. Managem Sci 20:822–844
29. Gersht A, Shulman A (1987) A new algorithm for the solution of the minimum cost multicommodity flow problem. Proc 26th IEEE Conf Decision and Control, pp 748–758
30. Grinold RC (1972) Steepest ascent for large scale linear program. SIAM Rev 14:447–464
31. Hane CA, Barnhart C, Johnson EL, Marsten RE, Nemhauser GL, Sigismondi G (1995) The fleet assignment problem: solving a large-scale integer program. Math Program 70:211–232
32. Hartman JK, Lasdon LS (1972) A generalized upper bounding algorithm for multicommodity network flow problems. Networks 1:333–354
33. Helgason R, Kennington J, Wong P (1981) An application of network programming for national forest planning. Techn Report, Dept Oper Res Southern Methodist Univ, Dallas
34. Jones KL, Lustig IJ, Farvolden JM, Powell WB (1993) Multicommodity network flows: The impact of formulation on decomposition. Math Program 62:95–117
35. Karkazis J, Boffey TB (1981) A subgradient based optimal solution method for the multicommodity problem. In: Burkard RE, Ellinger T (eds) Methods Oper Res 40. Anton Hain Verlag, pp 339–344
36. Kennington JL (1977) Solving multicommodity transportation problems using a primal partitioning simplex technique. Naval Res Logist Quart 24:309–325
37. Kennington JL (1978) A survey of linear cost network flows. Oper Res 26:209–236
38. Kennington JL, Helgason RV (1980) Algorithms for network programming. Wiley, New York
39. Kennington JL, Shalaby M (1977) An effective subgradient procedure for minimal cost multicommodity flow problems. Managem Sci 23:994–1004
40. Kim D (1997) Large scale transportation service network design: Models, algorithms and applications. PhD Thesis, Dept Civil Engin, MIT
41. Lasdon L (1970) Optimization theory for large systems. MacMillan, New York
42. Magnanti TL, Mirchandani P, Vachani R (1995) Modeling and solving the two-facility capacitated network loading problem. Oper Res 43(1)
43. Maier SF (1974) A compact inverse scheme applied to a multicommodity network with resource constraints. In: Cottle R, Krarup J (eds) Optimization Methods for Resource Allocation. English Univ Press, London, pp 179–203
44. McCallum CJ (1977) A generalized upper bounding approach to a commununications network planning problem. Networks 7:1–23
45. Minoux M (1986) Mathematical programming: Theory and algorithms. Wiley, New York
46. Moore E (1957) The shortest path through a maze. In: Proc Internat Symposium on the Theory of Switching, Harvard Univ Press, pp 282–292
47. Nabonna N (1993) Multicommodity network flow model for long term hydro-generation optimization. IEEE Trans Power Systems 8:395–404
48. Newton HN (1996) Network design under budget constraints with application to the railroad blocking problem. PhD Thesis, Auburn Univ, Alabama
49. Parker M, Ryan J (1994) A column generation algorithm for bandwidth packing. Telecommunication Systems 2:185–195
50. Raghavan P, Thompson CD (1987) Randomized rounding: A technique for provably good algorithms and algorithmic proofs. Combinatorica 4:365–374
51. Ritter K (1967) A decomposition method for linear programming problems with coupling constraints and variables. Techn Report, Math Res Center Univ Wisconsin 739
52. Robacker JT (1956) Notes on linear programming: Part XXXVII concerning multicommodity networks. Techn Report, The Rand Corp RM-1799
53. Rosen JB (1964) Primal partition programming for block diagonal matrices. Numerische Math 6:250–260
54. Saigal R (1967) Multicommodity flows in directed networks. Techn Report Oper Res Center, Univ Calif ORC 67–38
55. Sakarovitch M, Saigal R (1967) An extension of generalized upper bounding techniques for structured linear programs. SIAM J Appl Math 15:906–914
56. Saviozzi G (1986) Advanced start for the multicommodity network flow problem. Math Program Stud 26:221–224
57. Schneur R (1991) Scaling algorithms for multi-commodity flow problems and network flow problems with side constraints. PhD Diss, Massachusetts Inst Techn
58. Schultz GL, Meyer RR (1991) An interior point method for block angular optimization. SIAM J Optim 1:583–602
59. Segall RS (1995) Mathematical modeling of spatial price equilibrium for multicommodity consumer flows of large markets using variational inequalities. Appl Math Modeling 19(2):112–122
60. Shetty B, Muthukrishnan R (1990) A parallel projection for the multicommodity network model. J Oper Res Soc 41:837–842
61. Swoveland C (1971) Decomposition algorithms for the multicommodity distribution problem. Working Paper Western Management Sci Inst Univ Calif, Los Angeles 184

62. Vaidya PM (1989) Speeding up linear programming using fast matrix multiplication. Proc 30th Annual Symp Foundations of Computer Sci, pp 332–337
63. Vance PH, Barnhart C, Johnson EL, Nemhauser GL, Mahidara D, Krishna A, Rebello R (1994) Exceptions in crew planning. ORSA/TIMS Detroit, Michigan
64. Vanderbeck F, Wolsey LA (1996) An exact algorithm for IP column generation. Oper Res Lett 19:151–159
65. Zenios SA (1991) On the fine-grain decomposition of multicommodity transportation problems. SIAM J Optim 1:643–669
66. Ziarati K, Soumis F, Desrosiers J, Gelinas S, Saintonge A (1995) Locomotive assignment with heterogeneous consists at CN North America. Working Paper GERAD and École Polytechnique de Montréal

Multicriteria Decision Support Methodologies for Auditing Decisions

Chrysovalantis Gaganis,
Constantin Zopounidis
Financial Engineering Laboratory, Department of Production Engineering and Management, Technical University of Crete, University Campus, Chania, Greece

MSC2000: 90C90, 90C11, 91B28

Article Outline

Keywords

Qualified audit reports; Multicriteria decision aid; Auditing; Classification

Introduction

References to Financial Statements Fraud (FSF) and earnings manipulation have attracted the attention of market participants, academics, and regulators all over the world especially in recent years and following the collapse of Enron. During these years there have also been several cases of financial statement fraud which have been undetected by the auditors.

Using normal audit procedures, the detection of falsified financial statements is a difficult task [18,73]). There are numerous reasons for these difficulties such as a shortage of knowledge concerning the characteristics of management fraud, the efforts of managers to deceive auditors, and difficulties in collecting, analyzing and synthesizing large quantities of data from several different sources.

Models of audit reporting have several uses (i. e. prediction, determination, bankruptcy), as described by Dopuch et al. [26]. For example, they can provide a benchmark representing the probability that an auditor would issue a modified audit report on a given company. Furthermore, these models can be imperative in an auditing system that enables the users to take preventive or corrective actions [30,57,80]).

Most of the earlier studies of FSF have used discrete choice models in which the dependent variable was dichotomous. Mutchler [66] and Levitan and Knoblett [60] used discriminant analysis, Dopuch et al. [26] and Lennox [59] used probit models, Keasey et al. [48], Bell and Tabor [9], Monroe and Teh [65], Louwers [62], DeFond et al. [25], Citron and Taffler [17], Menon and Schwartz [63], and Spathis [80] used logit models, Krishnan [55] used an ordered probit model, Spathis et al. [79,81]), and Pasiouras et al. [71] used multicriteria decision aid (UTADIS) and multivariate statistical techniques (e. g. discriminant and logit analysis), Gaganis and Pasiouras [35] used discriminant and logit models, Gaganis et al. [36] used probabilistic neural network models, Gaganis et al. [33] used nearest neighbor models, Fanning et al. [31] and Fanning and Cogger [30] used artificial neural networks, and Doumpos et al. [27] used support vector machines.

In the present study, a multicriteria approach was followed through the application of the nonparametric **M**ulti-group **H**ierarchical **DIS**crimination (MHDIS) method with the aim of developing a sorting model to detect those firms that issue FSF in Greece. The MHDIS model was compared with logit analysis in order to test its efficiency against a benchmark that has been commonly used in previous studies. MHDIS is not based on statistical assumptions, which often cause problems during the application of statistical methods (logit and probit analysis), and furthermore, it can easily incorporate qualitative data.

Although there have recently been a few attempts to develop models to detect falsified audit statements in Greece [15,51,79,81]. The present study differs in several respects. First, we have used a more recent and larger dataset than in previous studies, which contains more detailed information. The data used corresponds to the fiscal years 2001–2004 and covers 398 companies. Spathis et al. [79,81] and Kirkos et al. [51] examined the same random sample of 76 manufacturing companies covering the period 1997–1999, and Caramanis and Spathis [15] examined a sample of 182 companies. Second, we examined both listed and unlisted companies from the manufacturing, trade, and services sectors in contrast to Spathis et al. [79,81] and Kirkos et al. [51] who examined only manufacturing listed companies and Caramanis and Spathis [15] who considered only listed companies. Third, we used out-of-time and out-of-sample testing samples. When evaluating the classification ability of a model, it is important to ensure that it has not been over-fitted to the training (estimation) dataset. As Stein [83] mentions "a model without sufficient validation may only be a hypothesis". Previous research has shown that when classification models are used to reclassify the observations of the training sample, the classification accuracies are biased upward. Thus, it is necessary to classify a set of observations which were not used during the development of the model, using some kind of testing sample.

The rest of the paper is organized as follows: Section "Sample" describes the sample used in this study. Section "Method" describes the methodology. Section "Empirical Results" presents the empirical results, and the Sect. "Conclusions and Further Research" discusses the concluding remarks and suggests some possible future research directions.

Sample

The data used in the study consisted of financial statement information (i. e. balance sheet, income statement, auditors' opinions, and the notes to financial statements) of a sample of companies obtained from ICAP[1] database and Athens Stock Exchange (ASE). Our analysis was restricted to Greek limited (société anonyme) and limited liability companies, which are obliged by law to have their financial statements audited, and we focused on the period between 2001 and 2004.

We obtained 199 qualified cases which were distributed over various sectors[2]. The next step was to select unqualified firms. We used a pair-matching method by sector. Matching of firms is common practice when conducting classification studies in auditing as well as in other areas of finance, such as bankruptcy or acquisitions prediction (e. g. [11,34,50,58,71]). There are two primary reasons for following this procedure, which is known as choice-based sampling. The first is the lower cost of collecting data in comparison with an unmatched sample [6,46,90]). The second and most important is that a choice-based sample provides greater information content than a random sample [19,45,69]). Hence, our sample consisted of the same number of qualified and unqualified cases.

Most of the previous studies concerning the development of models to replicate (or predict) auditors' opinion used training and testing samples from the same period, or re-sampling techniques such as jackknife and bootstrap (e. g. [57,79,81]). However, as Espahbodi and Espahbodi [29] point out, the real test of a classification model and its practical usefulness is its ability to classify objects correctly in the future. The main reason, as stated by Barnes [5], is that given inflationary effects, technological and other reasons, such as accounting policies, it is not reasonable to expect financial ratios to be stable over time. To account for this *population drifting*, in the present study, we split our sample of 398 companies into two distinct samples. The training sample, used for the development of the models, consisted of 234 companies and covered the period 2001–2003. The validation sample consisted of the remaining 164 companies and used data from 2004.

[1]ICAP is the largest company providing Business Information and Consulting Services in Greece.

[2]The sample for this study consists of 164 manufacturing, 122 trade and 110 services companies.

Financial Variables

Similar to previous studies we used financial ratios as the indicators of FSF. One of the problems with the selection of suitable ratios is that not only are there many financial variables which could be potential candidates for inclusion in the model, but also that previous studies have, in general, failed to select variables that reflect theoretical models of FFS as well. We therefore selected variables that were regarded as important in previous studies, such as Albrecht and Romney [1], Palmrose [70], Dopouch et al. [26], Loebbecke et al. [61], Green [38], Stice [84], Davia et al. [23], Bell et al. [10], Schilit [75], Arens and Loebbecke [3], Beasley [7], Bologna et al. [12], Krishnan and Krishnan [54], Green and Choi [39], Hoffman [42], Hollman and Patton [43], Zimbelman [89], Laitinen and Laitinen [57], Spathis et al. [79,81], Spathis [80], Doumpos et al. [27], Gaganis et al. [33,36], Gaganis and Pasiouras [35], and Pasiouras et al. [71]. Table 1 present a full list of the variables considered in this study. There are 28 financial variables covering all aspects of the performance of the selected companies, such as liquidity, leverage, profitability, managerial activity, and annual changes in basic accounts [20].

An examination of previous research indicates that the prediction variables range between 6 and 20. Most of these studies selected the effective independent variables using a statistical method, in an attempt to reduce the number of independent variables and the impact of potential multicollinearity. There is, however, little relevant theory about the selection of independent variables for the nonlinear methods. From a practical point of view, developing a model that considers a large number of variables poses problems for the applicability of the model on a daily basis by the auditor. This is because any application of the model requires that the auditor collects all necessary data, which leads to increased time and cost for data collection and management [79]. In the present study, we used a combination of two statistical analysis methods to examine whether there was an association between our variables and auditors' opinions and hence to select our final set.

First, we used the Kruskal–Wallis non-parametric test to examine the differences between qualified and unqualified companies. Table 1 present the results of the Kruskal–Wallis test for the training sample. Only three variables: 365*Stock/Cost of Sales; Logarithm of Debt; and Inventories/Total Assets were not statistically significant at the 10% level. We then reduced the number of variables to a manageable size using factor analysis.

This approach can be used to uncover the latent structure of a set of variables by reducing the attribute space from a larger number of variables to a smaller number of factors. The factor loadings were then used to select a limited set of financial variables.

Finally, seven financial variables were selected, each being the variable with the highest loading in each factor. These were: Receivable/Sales; Current Assets/Current Liabilities; Current Assets/Total Assets; Cash/Total Assets; Profit before tax/Total Assets; Inventories/Total Assets; and Annual Change in Sales.

Non-financial Variables

In addition to the seven financial variables discussed above, six non-financial ones were also used. Dopuch et al. [26] presented a predictive model of audit opinion qualifications in which the variables with greatest predictive power were categorical ones.

Previous studies mainly dealt with the construction of bankruptcy models for making audit opinions relative to going concern (e. g. [44,52,53]). Prior research (e. g. [17,30,41,61,74,79,84]) suggests that financial distress is very important in the issuing of an audit qualification. Although most of the previous studies used Altman's z-score [2] or credit risk assessment of a rating agency [71] as a proxy of default, such an approach may not be appropriate in our case. The reason is that Altman's z-score was developed for a particular industry (i. e. manufacturing), under different economic conditions (i. e. in the 1960s) and for a specific country (i. e. USA). In the present study, we used a score (UTADISCR) estimated from the UTADIS bankruptcy prediction model of Zopounidis et al. [92]. We anticipated that the use of this measure, which indicates the likelihood of default of Greek firms over the 12 months following the date of its calculation, might provide more accurate results.

Spathis [80] found that audit qualification decision was positively associated with company litigation. In this study, the client litigation variable was coded as zero if a company had litigation in the year preceding

Multicriteria Decision Support Methodologies for Auditing Decisions, Table 1
Descriptive statistics

	Unqualified		Qualified		Kruskal–Wallis
	Mean	St.Dev	Mean	St.Dev	
INVREC / TA	0.58	0.26	0.53	0.22	2.83 ***
REC / SAL	0.31	0.18	0.52	0.59	16.84 *
CA / CL	1.52	0.60	1.00	0.69	66.07 *
CA / TA	0.75	0.22	0.65	0.22	14.48 *
CASH / CL	0.22	0.28	0.08	0.18	26.59 *
CASH / TA	0.12	0.16	0.05	0.11	13.85 *
ROA	0.19	0.12	−0.12	0.16	157.80 *
PRBT / FA	5.04	11.16	−0.62	1.29	149.09 *
INV / SAL	0.12	0.12	0.27	0.41	6.28 **
SAL / TA	1.52	0.83	1.09	0.84	22.70 *
TD / TA	0.59	0.21	0.90	0.34	66.26 *
PRBT / CL	0.37	0.25	−0.15	0.24	157.60 *
CL / TA	0.56	0.23	0.82	0.34	40.98 *
WC / TA	0.20	0.18	−0.17	0.34	80.45 *
EBIT MARGIN	0.15	0.12	−0.19	0.39	157.17 *
GP / SAL	0.36	0.18	0.13	0.21	72.76 *
GP / TA	0.52	0.33	0.16	0.22	83.78 *
(CA − ST) / CL	1.18	0.47	0.74	0.56	65.87 *
365 * AREC / SAL	126.70	66.49	238.46	330.86	21.81 *
365 * ST / CS	73.29	79.92	113.16	163.22	0.41
SAL / EQ	7.36	10.81	0.05	32.71	31.89 *
SAL / TD	2.84	1.63	1.24	0.87	78.28 *
365 * AP / SAL	139.63	550.32	155.07	249.57	65.98 *
TACH	0.22	0.46	0.10	0.39	11.76 *
SALCH	0.15	0.39	0.10	0.44	5.79 **
LOGTA	7.45	0.53	7.28	0.46	7.50 *
LOGDEPT	7.19	0.54	7.19	0.45	0.05
INV / TA	0.16	0.13	0.16	0.15	0
UTADISCR	0.74	0.10	0.49	0.12	126.51**

Notes: INVREC / TA: (Inventories + Receivable) / Total Assets, REC / SAL: Receivable / Sales, CA / CL: Current Assets / Current Liabilities, CA / TA: Current Assets / Total Assets, CASH / CL: Cash / Current Liabilities, CASH / TA: Cash / Total Assets, ROA: Profit before tax × 100 / Total assets, PRBT / FA: Profit (Loss) before tax × 100 / Fixed Assets, INV / SAL: Inventories/Sales, SAL\TA: Sales / Total Assets, TD / TA: Total Dept / Total Assets, PRBT / CL : Profit (Loss) before tax × 100 / Current Liabilities, CL / TA: Current Liabilities / Total Assets, WC / TA: Working Capital / Total Assets, EBIT Margin: Profits before interest and taxes × 100 / Turnover, GP / SAL: Gross Profit / Sales, GP / TA: Gross Profit / Total Assets, (CA − ST) / CL: (Current assets − Stock) / Current liabilities, 365 * AREC / SAL:365 Accounts Receivable / Sales, 365 * ST / CS: 365 * Stock / Cost of Sales, SAL / EQ: Sales / Equity, SAL / TD: Sales / Total Dept, 365 * AP / SAL: 365* Accounts Payable / Sales, TACH: (Total Assets in year *t* − Total Assets in year *t*−1) × 100 / Total Assets in year *t*−1, SALCH: (Sales in year *t* − Sales in year *t*−1) × 100 / Sales in year *t*−1, LOGTA: Logarithm of Total Assets, LOGDEPT: Logarithm of Dept, INV / SAL : Inventories / Total Assets. The Kruskal–Wallis test indicates whether there are statistically significant differences between the two groups. ** Significant at the 1% level, * Significant at the 5% level, *** Significant at the 10% level the audit opinion and as one otherwise. Skinner [77] considered companies having litigation in the following cases: (a) a lawsuit had been filed in a Greek court; (b) there had been an allegation of common stock price fraud; or (c) there had been an allegation of stock exchange violation under Greek law.

The most consistent result in all previous research has been that auditor size can explain the supply of a higher level of audit quality, defined as the joint probability of detecting and reporting material financial errors (i. e., [4,8,22,24,25,28,47,48,64,72]). The evidence concerning the relationship between audit firms and audit report is mixed. Whereas Warren [88] did find a significant association between large audit firms and qualified audit reports, Shank and Murdock [76] found otherwise.

Previous studies also examined whether the auditor is one of the Big Four (namely PricewaterhouseCoopers, Deloitte and Touche, KPMG, Ernst and Young) or not [35]. We use a dummy variable set to zero (Domestic = 0) if the auditor was one of the domestic audit firms and one if the auditor was one of the foreign audit firms in Greece (Foreign = 1). In Greece the auditing profession was liberalized in 1992 by enabling legislation [37], see Caramanis [13,14]. The competition between the local Greek and foreign audit companies has increased since 1992. Nowadays, Greek companies audit a greater percentage of companies than foreign audit companies. It is possible that smaller companies avoid paying the premium price levied by the large audit companies, since Krishnan et al. [54] found that smaller firms in the US are less likely to be audited by the 'Big Four' companies. Furthermore, it is possible that the partners of domestic audit companies are more likely to develop close personal relationships with the directors of Greek client companies. On the other hand, there is a chance that the domestic audit companies will be more familiar with the 'small acceptable standards of control' in Greece.

Various papers examine the relationship between the audit opinion before and after the chance of auditors (switching). Chow and Rice [16], Craswell [21], Gul et al. [40], and Krishnan et al. [56] found a significant positive association between qualified opinions and subsequent auditor switching. As Nieves [68] points out, two effects may obscure the influence of the audit report in motivating a change: (a) many auditor changes may be unrelated to audit opinion; and (b) the reasons for auditor changes are an internal state which is not directly observable. We tested the importance of auditor changes to detection of FFS over a 3-year period. The 3-year period included the first year of the financial statements and auditors' opinions and the 2 years before this first year. We used a dummy variable (Prior 2 year auditor) that takes a value of one (yes = 1) if the auditor had been retained and a value of zero if the firm had switched auditors (no = 0).

Finally, we used two other variables, LOSS and STOCK. LOSS is an indicator variable whose value is zero if an auditee experienced a loss in the year of audit opinion and one (profit) otherwise. Spathis [80] found a significant difference between qualified and non-qualified audit reports for this variable. STOCK is a dummy variable that takes a value of zero (yes = 0) for companies listed in the Athens Stock Exchange and one for unlisted companies (no = 1). Ireland [46] reported that whether a company is listed or unlisted may influence the auditor's independence. Listed companies may have greater supervision and training of their stock exchange authority. Furthermore, as Ireland [46] points out, large companies are more likely to have good accounting systems and internal controls, thus reducing disagreements and limitations on scope while, at the same time, auditors are more likely to waive earnings management attempts (resulting in mis-statements) in large clients, even after controlling for the materiality of such attempts [67].

Method

Multicriteria decision making (MCDM) provides the methodological basis for the combination of qualitative and quantitative data. MCDA has been applied in finance as a sophisticated tool to improve the decision-taking in the turbulent and complex financial environment that exists nowadays. Spronk et al. [82] thoroughly investigated the application of this technique in the financial field. In the present study we used the MHDIS method [91].

The problem considered in this case study falls within the classification problematic that in general involves the assignment of a finite set of alternatives $A = \{a_1, a_2, \ldots, a_n\}$ to a set of q ordered classes $C_1 \succ C_2 \succ \cdots \succ C_q$. Each alternative was evaluated along a set of m criteria $g_1, g_2, \dots, g_m$. In the present case study the alternatives involved the companies in the sample, the criteria correspond to the set of seven financial variables, and six non-financial variables and there were two classes, the unqualified financial statements (class C_1) and the qualified financial statements (class C_2).

MHDIS distinguishes the groups progressively, starting by discriminating the first group from the others, and then proceeds to the discrimination between the alternatives belonging to the other group. To accomplish this task two additive utility functions are developed in each one of the $q-1$ steps, where q is the number of groups. The first function $U_k(a)$ describes the alternatives of group C_1, and the second function $U_{\sim k}(a)$ describes the remaining alternatives that are classified in lower groups $C_{k+1}, \dots, C_q$.

$$U_k(a) = \sum_{i=1}^{m} p_{ki} u_{ki}(g_i)$$
$$\text{and } U_{\sim k}(a) = \sum_{i=1}^{m} p_{\sim ki} u_{\sim ki}(g_i),$$
$$k = 1, 2, \dots, q-1.$$

The corresponding marginal utility functions for each criterion g_i are denoted as $u_{ki}(g_i)$ and, $u_{\sim ki}\left(g_i\right)$ which are normalized between 0 and 1, while the criteria weights p_{ki} and $p_{\sim ki}$ sum up to 1. As mentioned above, the model is developed in $q-1$ steps. In the first step, the method develops a pair of additive utility functions $U_1(a)$ and $U_{\sim 1}(a)$ to discriminate between the alternatives of group C_1 and the alternatives of the other groups $C_2, \dots, C_q$. On the basis of the above function forms the rule to decide upon the classification of any alternative has the following form:

If $U_1(a) \geq U_{\sim 1}(a)$ then a belongs in C_1.

Else if $U_1(a) \leq U_{\sim 1}(a)$ then a belongs in ($C_2, C_3, \dots, C_q$).

The alternatives that are found to belong in class C_1 (correctly or incorrectly) are excluded from further analysis. In the next step, another pair of utility functions $U_2(a)$ and $U_{\sim 2}(a)$ is developed to discriminate between the alternatives of group C_2 and the alternatives of the groups $C_3, \dots, C_q$. Similarly to step 1, the alternatives that are found to belong in group C_2 are excluded from further analysis. This procedure is repeated up to the last stage $(q-1)$, when all groups have been considered. The estimation of the weights of the criteria in the utility functions as well as the marginal utility functions is accomplished through mathematical programming techniques. More specifically, at each stage of the hierarchical discrimination procedure, two linear programs and a mixed-integer one are solved to estimate the two additive utility functions optimally and to minimize the classification error. Further details of the mathematical programming formulations used in MHDIS can be found in Zopounidis and Doumpos [91].

Empirical Results

Table 1 presents descriptive statistics which indicate the magnitude of the difference in the independent variables between the qualified and unqualified reports over the period 2001–2003. A comparison between the mean value of UTADISCR for the qualified and unqualified companies in the training sample shows that the former had a lower average value, which was statistically significant at the 1% level. Hence, many companies that had manipulated their financial statements were in financial distress [30,36,78]). Statistically significant differences at the 1% level were also found for two others variables, namely Inventories/Sales and Sales Annual Change, between the two groups in the training sample. Thus, companies with lower sales are more likely to receive a qualified report than other companies in Greece.

Furthermore, the variable Profits before tax/Total Assets (ROA) had lower means for the qualified companies, which was consistent with most of the previous studies, indicating that firms which receive qualified opinions made less profit [7,61,78,81,86]).

Table 2 illustrates the contribution of each of the financial and non-financial criteria in our auditing model. As our study involved two groups (unqualified and qualified) the hierarchical discrimination process of MHDIS consisted of only one stage, during which two additive utility functions were developed. The utility function U_1 characterizes the unqualified companies whereas the utility function $U_{\sim 1}$ characterizes those that were qualified.

ROA is indicated as one of the important criteria in most cases. Particularly in the case of unquali-

Multicriteria Decision Support Methodologies for Auditing Decisions, Table 2
Average weights for the criteria in the 2 models

	MHDIS financial		MHDIS No financial	
	U1	U∼1	U1	U∼1
REC / SAL	2.28%	3.82%	0.02%	0.02%
CA / CL	7.17%	25.18%	3.99%	16.41%
CA / TA	21.49%	5.02%	15.04%	10.14%
CASH / TA	4.39%	11.23%	16.87%	1.11%
ROA	44.75%	25.71%	28.09%	11.57%
INV / SAL	12.54%	13.74%	10.55%	0.02%
SALCH	7.38%	15.30%	1.50%	3.94%
Profit or Loss in the year			0.00%	0.00%
Stock Exchange			0.00%	17.49%
Auditor			0.00%	0.00%
Prior 2 years Auditor			9.93%	0.00%
Litigation			0.00%	22.37%
UTADISCR			14.01%	16.94%

Notes: REC / SAL: Receivable / Sales, CA / CL: Current Assets / Current Liabilities, CA / TA: Current Assets / Total Assets, CASH / TA: Cash / Total Assets, ROA: Profit before Tax × 100 / Total Assets, INV / SAL: Inventories / Total Assets, SALCH: (Sales in year *t* − Sales in year *t* − 1) × 100 / Sales in year *t* − 1

Multicriteria Decision Support Methodologies for Auditing Decisions, Table 3
Classification results (accuracies in %) for the MHDIS and Logit models

	Unqualified	Qualified	Average
Panel A:Financial Variables			
Training(2001-2003)			
MHDIS	94.87	95.73	95.30
LA	95.7	95.7	95.7
Holdout (2004)			
MHDIS	80.49	87.80	84.15
LA	78.00	86.6	82.3
Panel B: Non-financial Variables			
Training (2001-2003)			
MHDIS	98.29	98.29	98.29
LA	97.43	95.73	96.58
Holdout (2004)			
MHDIS	84.15	91.46	87.81
LA	74.39	93.90	84.15

fied firms, it has a weight that is as high as 44.75% in the financial model and 28.09% in the non-financial one. Similar findings were observed in previous studies [7,61,71,78,81,86]).

The most important criteria that characterized the qualified firms in the case of the financial model are ROA and Current Assets/Current Liabilities (CA/CL) followed by Sales in year *t* − Sales in year *t* − 1 (SALCH) with weights of 25.71, 25.18 and 15.30%, respectively. Pasiouras et al. [71] also found ROA to be statistically significant at the 1% level and one of the most important criteria for the models. In addition, Ireland [46] reported that companies with high liquidity (CA/CL) might increase the likelihood of a qualified audit opinion as assets may have been overstated.

From the non-financial criteria, litigation, STOCK, UTADISCR and CA/CL were the most important criteria with 22.37, 17.49, 16.94 and 16.41%, respectively, for qualified companies. A comparison with the results of previous studies showed that the results were similar. In particular, Spathis [80] found that litigation and financial distress were among the most important variables, and Spathis et al. [79] found CA/CL to be among the most important factors.

Table 3 presents the classification results obtained from the financial and non financial models. The classification ability of the models was tested further using the out-of-time and out-of-sample companies. The results indicated that the MHDIS models developed with the selected variables were able to provide a satisfactory distinction between qualified and unqualified statements.

The overall correct classifications at the training and holdout stages were 95.3 and 84.15%, respectively. The differences between the financial and the non-financial MHDIS models were significant. Overall, the non-financial MHDIS model provided higher overall classification accuracy in both the training (92.3%) and holdout samples (87.81%). This means that the inclusion of non-financial variables in the model yielded a more accurate distinction between qualified and unqualified companies than the inclusion of financial variables alone.

For benchmarking purposes, we developed additional models with logit analysis (LA). These models were developed with the same input variables. In the case of the financial model, the classification accuracy in the training sample was 78%, and the corresponding figure for the holdout sample was 74.3%. In the case of the non-financial model, however, the classification accuracies were 82.3 and 84.15% in the training and holdout samples, respectively. It is therefore clear that MHDIS was more efficient than LA during both the training and holdout stages (for both financial and non-financial models).

MHDIS achieves more balanced results in terms of type I and type II errors in the holdout sample. Whereas Bell and Tabory [9] reported that type II errors are more costly than type I errors, Kida [49] argued that type I errors might result in: (a) a company changing its audit firm (switching), which means loss in audit firm revenue; (b) a lawsuit by a client against the accounting firm; (c) a negative effect on the auditor's reputation in the business community; (d) a deterioration in relations with the client; or (e) the so-called self-fulfilling prophecy – the qualification itself jeopardizes client survival, which in turn increases the probability of that consequence.

Conclusions and Further Research

This study investigated the extent to which MHDIS models based on financial and non-financial variables could predict auditors' decisions to issue qualified opinions in the Greek market.

The sample consisted of 199 companies operating in the Greek manufacturing, trade and service sectors with FSF between 2001 and 2004, matched by industry and total assets with 199 non-FSF ones, yielding a total of 398 companies. We used out-of-time and out-of-sample testing samples to evaluate the classification ability of the model and ensure that they were not overfitted to the training dataset. The sample was split into a training dataset of 234 companies using data from the period 2001–2003 and a validation dataset of 164 companies using data from the year 2004.

Seven financial and six non-financial variables, representing all dimensions of companies' performance, were selected for inclusion in the models that were developed through the MHDIS approach. The results indicated that ROA, CA/CL and Current Assets/Total Assets A were the important criteria for financial model. In additional, litigation, stock exchange, UTADISCR and CA/CL were the most important criteria for the non-financial model. Furthermore, the non-financial MHDIS model provides higher overall classification accuracy indicating that the inclusion of non-financial variables resulted in a more accurate distinction between qualified and unqualified companies.

By using such models, auditors can simultaneously screen a large number of firms and direct their attention to the ones that are more likely to contain misstatements, saving time or money. These models can also be used by policy-makers in an attempt to stop tax evasion (i. e. the tax evasion consisting of filing fraudulent tax declarations in Italy is estimated to be between 3 and 10% of GNP [87]). In addition, these models can be useful to investors, managers, banks and others companies to identify 'red flags'.

The current research could be extended in several directions. First, future research could be extended towards the inclusion of additional variables such as managers' experience, market characteristics (i. e. industry concentration, industry growth), audit fees and non-audit fees, subsidiaries, and stock prices. Second, companies could be classified into more specific groups. Third, the inclusion of data from a longer time period, could allow the consideration of industry and macroeconomic effects. Finally, future research could be directed towards the comparison and integration of alternative or additional classification techniques, such as neural networks, rough sets, expert systems, support vector machine, and others. The integration of the models through additional techniques, such as bagging and boosting, could also be examined.

References

1. Albrecht S, Romney M (1986) Red-flagging management: a validation. Adv Account 3:323–33
2. Altman E (1983) Corporate financial distress: A complete guide to predicting, avoiding, and dealing with bankruptcy. Wiley, New York
3. Arens A, Loebbecke J (1994) Auditing: An Integrated Approach, 6th edn. Prentice-Hall, Englewood Cliffs
4. Balvers R, McDonald B, Miller R (1988) Underpricing of New Issues and the Choice of Auditor as a Signal of Investment Banker Reputation. Account Rev 63:605–21
5. Barnes P (1990) The prediction of takeover targets in the U.K. by means of multiple discriminant analysis. J Bus Finance Account 17(1):73–84
6. Bartley JW, Boardman CM (1990) The relevance of inflation adjusted accounting data to the prediction of corporate takeovers. J Bus Finance Account 17(1):53–72

7. Beasley SM, Carcello JV, Hermanson DR (1999) fraudulent financial reporting: 1987–1997: an analysis of US public companies, research report, Committee on Sponsoring Organizations of the Treadway Commission, AICPA, New York
8. Beatty R (1989) Auditor Reputation and the Pricing of Initial Public Offerings. Account Rev 64:693–709
9. Bell T, Tabor R (1991) Empirical analysis of audit uncertainty qualifications. J Account Res 29:350–370
10. Bell T, Szykowny S, Willingham J (1993) Assessing the likelihood of fraudulent financial reporting: a cascaded logic approach, Working Paper. KPMG Peat Marwick, Montvale
11. Bhargava M, Dubelaar C, Scott T (1998) Predicting bankruptcy in the retail sector: an examination of the validity of key measures of performance. J Retailing Consumer Services 5(2):105–117
12. Bologna G, Lindquist R, Wells J (1996) The Accountant's Handbook of Fraud and Commercial Crime. Wiley, New York
13. Caramanis VC (1997) The enigma of the Greek auditing profession: some preliminary results concerning the impact of liberalization on auditor behaviour. Eur Account Rev 6(1):85–108
14. Caramanis VC (1999) International accounting firms versus indigenous auditors: Intra-professional conflict in the Greek auditing profession. Critical Persp Account 10:153–196
15. Caramanis C, Spathis C (2006) Auditee and audit firm characteristics as determinants of audit qualifications evidence from the Athens stock exchange. Managerial Auditing J 21(9):905–920
16. Chow CW, Rice SJ (1982) Notes: qualified audit opinions and auditor switching. Account Rev LVII(April):326–335
17. Citron DB, Taffler RJ (1992) The audit report under going concern uncertainties: an empirical analysis. Account Bus Res 22(88/Autumn):337–345
18. Coderre GD (1999) fraud detection, using Data Analysis Techniques to detect fraud. Global Audit Publications, Vancover
19. Cosslett SR (1981) Efficient estimation of discrete choice models. In: Manski CF, McFadden D (eds) Structural analysis of discrete data with econometric applications. MIT Press, Cambridge
20. Courtis JK (1978) Modelling a financial ratios categoric framework. J Bus Finance Account 5(4):371–386
21. Craswell AT (1988) The association between qualified opinions and auditor switches. Account Bus Res 19:23–31
22. Craswell AT, Francis R, Taylor SL (1995) Auditor Brand Name Reputations and Industry Specialization. J Account Econ 20:297–322
23. Davia H, Coggins P, Wideman J, Kastantin J (1992) Management Accountant's Guide to Fraud Discovery and Control. Wiley, New York
24. De Angelo LE (1981) Auditor size and Audit Quality. J Account Econ 3:183–199
25. DeFond ML, Wong TJ, Li S (2000) The Impact of Improved Auditor Independence on Audit Market Concentration in China. J Account Econ 28:269–305
26. Dopouch N, Holthausen R, Leftwich R (1987) Predicting audit qualifications with financial and market variables. Account Rev 62(3):431–454
27. Doumpos M, Gaganis C, Pasiouras F (2005) Explaining qualifications in audit reports using a support vector machine methodology. Intelligent Syst Account, Finance Manage 13:197–215
28. Dye R (1993) Auditing Standards, Legal Liability and Auditor Wealth. J Polit Econ 101:887–914
29. Espahbodi H, Espahbodi P (2003) Binary choice models for corporate takeover. J Banking Finance 27:549–574
30. Fanning K, Cogger KO (1998) Neural network detection of management fraud using published financial data. Int J Intelligent Syst Account, Finance Manage 7(1):21–41
31. Fanning K, Cogger K, Srivastava R (1995) Detection of management fraud: a neural network approach. Int J Intelligent Syst Account, Finance Manage 4(2):113–26
32. Fanning K, Cogger K (1998) Neural network detection of management fraud using published financial data. Int J Intelligent Syst Account, Finance Manage 7(1):21–24
33. Gaganis C, Pasiouras F, Spathis C, Zopounidis C (2005) Identifying qualified audit reports in UK firms: a nearest neighbours approach. In: 28th European Accounting Association Annual Congress, Göteborg, Sweden, 18–20 May 2005
34. Gaganis C, Pasiouras F, Tzanetoulakos A (2005) A Comparison and Integration of Classification Techniques for the Prediction of Small UK Firms Failure. J Financ Decis Making 1:55–69
35. Gaganis C, Pasiouras F (2006) Auditing models for the detection of qualified audit opinions in the UK public services sector, Int J Account, Auditing Perform Eval 3(4):471–493
36. Gaganis C, Pasiouras F, Doumpos M (2007) Probabilistic neural networks for the identification of qualified audit opinions. Expert Syst Appl 32:114–124
37. Government Gazette (1993) Law 2166/1993, A/137/24–8–1993. Ethniko Typographeio, Athens
38. Green B (1991) Identifying management irregularities through preliminary analytical procedures, unpublished doctoral dissertation. Kent State University, Kent
39. Green BP, Choi JH (1997) Assessing the risk of management fraud through neuralnetwork technology. J Pract Theory 16(1):14–28
40. Gul FA, Lee DS, Lynn M (1992) A note on audit qualification and switches: some further evidence from a small sample study. J Int Account, Auditing Taxation 1(1):111–120
41. Haskins M, Williams D (1990) A contingent model of intra-Big Eight auditor changes. J Pract Theory 3:55–74
42. Hoffman VB (1997) Discussion of the effects of SAS No. 82 on auditors attention to fraud risk factors and audit planning decisions. J Account Res 35(5):99–104

43. Hollman VP, Patton JM (1997) Accountability, the dilution effect and conservatism in auditors fraud judgments. J Account Res 35(2):227–237
44. Hopwood W, McKeown J, Mutchler J (1994) A Reexamination of Auditor versus Model Accuracy within Xontext of the Going-Concern Opinion Decision. Contemp Account Res 2(10):409–431
45. Imbens W (1992) An efficient method of moments estimator for discrete choice models with choice-base sampling. Econ 60:1187–1214
46. Ireland J (2003) An empirical investigation of determinants of audit reports in the UK. J Bus Finance Account 30(7&8):975–1015
47. Ireland JC, Lennox CS (2002) The Large Audit Firm Fee Premium: A Case of Selectivity Bias? J Account, Auditing Finance 17(1/Winter):73–91
48. Keasey K, Watson R, Wynarzcyk P (1988) The Small Company Audit Qualification: A reliminary Investigation. Account Bus Res 18:323–33
49. Kida T (1980) An investigation into auditor's continuity and related qualification judgments. J Account Res Autumn:506–523
50. Kira D, Morin D (1993) Prediction of takeover targets of Canadian Firms. In: Janssen J, Skiadas CH (eds) Applied Stochastic Models and Data Analysis, pp 507–518. World Scientific Publ Co, Singapore
51. Kirkos E, Spathis C (2006) Data mining techniques for the detection of fraudulent financial statements. Expert Systems with Application, Forthcoming
52. Koh HC, Killough LN (1990) The Use of Multiple Discriminant Analysis in the assessment of the Going Concern status of an audit client. J Bus Finance Account 17(2):179–192
53. Koh HC (1991) Model predictions and auditor assessments of going concern status. Account Bus Res 21(84):331–338
54. Krishnan J, Krishnan J (1996) The role of economic tradeoffs in the audit opinion decision: an empirical analysis. J Account, Auditing Finance 11(4):565–586
55. Krishnan J (1994) Auditor Switching and Conservatism. Account Rev 69:200–16
56. Krishnan J, Krishnan J, Stephens RG (1996) The simultaneous relation between auditor switching and audit opinion: an empirical analysis'. Account Bus Res 26(Summer):224–236
57. Laitinen EK, Laitinen T (1998) Qualified audit reports in Finland: evidence from large companies. Eur Account Rev 7(4):639–653
58. Laitinen T, Kankaanpaa M (1999) Comparative analysis of failure prediction methods: the Finnish case. Eur Account Rev 8:67–92
59. Lennox CS (1999) The Accuracy and Incremental Information Content of Audit Reports in Predicting Bankruptcy. J Bus, Finance Account 26(May–June):757–70
60. Levitan AS, Knoblett JA (1985) Indicators of Expectations to the Going-Concern Assumption. J Pract Theory 5(1):26–39
61. Loebbecke J, Eining M, Willingham J (1989) Auditor's experience with material irregularities: frequency, nature and detectability. J Pract Theory 9:1–28
62. Louwers TJ (1998) The Relation Between Going-Concern Opinions and the Auditor's Loss Function. J Account Res 36(1):143–56
63. Menon K, Schwartz KB (1987) An empirical investigation of audit qualification decisions in the presence of going-concern uncertainties. Contemp Account Res 3(2):302–15
64. Menon K, Williams D (1991) Auditor Credibility and Initial Public Offerings. Account Rev 66:313–32
65. Monroe G, The S (1993) Predicting Uncertainty Audit Qualifications in Australia Using Publicly Available Information. Account Finance 33(2):79–106
66. Mutchler JF (1985) A Multivariate Analysis of the Auditor's Going-Concern Opinion Decision. J Account Res 23(2): 668–82
67. Nelson M, Elliot JA, Tarpley RL (2000) Where do companies attempt earnings management, and when do auditors prevent it? Working Paper, Cornell University, New York
68. Nieves GA, Eiliano RB (2003) Do Spanish firms changes auditor to avoid a qualified audit report? Int J Auditing 7:37–53
69. Palepu KG (1986) Predicting Takeover Targets: A Methodological and Empirical Analysis. J Account Econ 8:3–35
70. Palmrose Z (1987) Litigation and independent auditors: the role of business failures and management fraud. J Pract Theory 6(2):90–102
71. Pasiouras F, Gaganis C, Zopounidis C (2007) Multicriteria decision support methodologies for auditing decisions: the case of qualified audit reports in the UK. Eur J Oper Res 180(3):1317–1330
72. Pong CM, Whittington G (1994) The Determinants of Audit Fees: Some Empirical Models. J Bus Finance Account 21(8):1071–95
73. Porter B, Cameron A (1987) Company fraud-what price the auditor? Account J December:44–47
74. Reynolds J, Francis J (2001) Does size matter? The influence of large clients on office-level auditor reporting decisions. J Account Econ 30:375–400
75. Schilit H (1993) Financial Shenanigans: How to Detect Accounting Gimmicks and Fraud in Financial Reports. McGraw-Hill, New York
76. Shank J, Murdock R (1978) Comparability in the Application of Reporting Standards: Some Further Evidence. Account Rev (October):824–35
77. Skinner JD (1997) Earnings disclosures and stockholder law suits. J Account Econ 23(3):249–282
78. Spathis C (2002) Detecting false financial statements using published data: some evidence from Greece. Managerial Auditing J 17(4):174–191
79. Spathis C, Doumpos M, Zopounidis C (2003) Using client performance measures to identify pre-engagement fac-

tors associated with qualified audit reports in Greece. Int J Account 38:267–284
80. Spathis C (2003) Audit qualification, firm litigation, and financial information: an empirical analysis in Greece. Int J Auditing 7(1):71–85
81. Spathis C, Doumpos M, Zopounidis C (2002) Detecting falsified financial statements: a comparative study using multicriteria analysis and multivariate statistical techniques. Eur Account Rev 11(3):509–535
82. Spronk J, Steuer RE, Zopounidis C (2005) Multicriteria Decision Aid/Analysis in Finance. In: Figueira, Greco, Ehrgott (eds) Multiple Criteria Decision Analysis: State of the Art Surveys. Springer, Boston, Dordrecht, London, pp 799–858
83. Stein RM (2002) Benchmarking default prediction models: Pitfalls and Remedies in Model Validation Moody's KMV Technical Report #030124, June 13
84. Stice J, Albrecht S, Brown L (1991) Lessons to be learned-ZZZZBEST Regina and Lincoln saving. CPA J April:52–54
85. Stice J (1991) Using financial and market information to identify pre-engagement market factors associated with lawsuits against auditors. Account Rev 66(3):516–33
86. Summers SL, Sweeney JT (1998) Fraudulently misstated financial statements and insider trading: an empirical analysis. Account Rev 73(1):131–146
87. Tanzi V, Shome P (1993) A primer on tax evasion. In: International Monetary Fund, Staff Papers 40(4)807–828
88. Warren C (1980) Uniformity of auditing standards: a replication. J Account Res (Spring):312–24
89. Zimbelman MF (1997) The effects of SAS No. 82 on auditors attention to fraud risk-factors and audit planning decisions. J Account Res 35(5):75–9
90. Zmijewski ME (1984) Methodological Issues Related to the Estimation of Financial Distress Prediction Models. J Account Res 22(Supplement):59–86
91. Zopounidis C, Doumpos M (2000) Building additive utilities for multi-group hierarchical discrimination: the M.H.D.I.S method. Optim Methods Softw 14:219–240
92. Zopounidis C, Doumpos M, Gaganis C (2006) Prediction of firms' Bankruptcy: a Multicriteria Methodology. Rev Econ Sci (9):283–296

Multicriteria Methods for Mergers and Acquisitions

Fotios Pasiouras[1], Constantin Zopounidis[2]
[1] School of Management, University of Bath, Bath, UK
[2] Financial Engineering Laboratory, Department of Production Engineering and Management, Technical University of Crete, Chania, Greece

MSC2000: 91B28, 90C05, 90C90

Article Outline

Keywords and Phrases

Acquisitions; Additive utility functions; Mergers; Multicriteria decision aid; Takeovers

Introduction/Background

Over the last 35 years there have been several studies which have attempted to develop classification models to predict takeover targets in various countries and regions of the world, such as the USA [7,8,10,11,12,15,23,30,34,37], the UK [2,3,4,5,13,28,36], Canada [9,20,29], Greece [31,38,39], and more recently the EU [26,27] and Asia [25]. This is not surprising, since the prediction of acquisitions can be of major interest to stockholders, investors, creditors, and generally anyone who has established a relationship with the acquired firm [35].

Most of these studies have used multivariate statistical and econometric techniques such as discriminant analysis (DA) and logit analysis and only more recently the parametric nature and the statistical assumptions and restrictions of those approaches have led researchers to the application of alternative techniques such as artificial neural networks (ANN) [10], rough sets (RS) [31], recursive partitioning algorithm [15], support vector machines [26], and nearest neighbors [26].

A few recent studies have also used multicriteria decision aid (MCDA, which is the designation usually used in Europe, or multiple criteria decision making, MCDM, which is the one usually used in the USA) techniques [13,25,26,27,36,39] which over the last few years have gained significant recognition among researchers and have been employed in several studies in banking, finance, accounting, and management. For example, Steuer and Na [33] identified 256 applications that combine MCDM and finance. One of the characteristics of these techniques is that they are well suited for analyzing complex decision problems that involve multiple and usually conflicting criteria and/or goals. They can therefore prove particularly useful in the prediction of acquisitions, since there is often not one single reason but a number of reasons that lead management to the decision to merge with or acquire another firm. A further advantage of MCDA techniques is that they do not make any assumptions, as do the traditional techniques (see Barniv and McDonald [6] for a summary of the problems related to the use of discriminant, logit, and probit), about the normality of the variables or the group dispersion matrices (e. g., DA) and they are not sensitive to multicollinearity (e. g., logit analysis). In this paper we first present a brief review of the studies that have applied MCDA in the prediction of acquisition targets (Sect. "Methods/Applications"). Then, in Sect. "Formulation", we outline one of the MCDA techniques, namely, *utilité*s additives discriminantes (UTADIS), which is used for the development of our classifications models. "Cases" describes a case study and presents the results. Finally, Sect. "Conclusions" concludes our paper.

Methods/Applications

Zopounidis and Doumpos [39] were the first to propose the use of MCDA in the prediction of acquisition targets. They developed a classification model with the multigroup hierarchical discrimination (MHDIS) method using a sample of 30 acquired and 30 nonacquired Greek firms and ten financial ratios covering various aspects of a firm's financial condition. Data from 1 year prior to the acquisition (year −1) were used for the development of the model, while years 2 (year −2) and 3 (year −3) before the acquisition were used to test its discrimination ability. The model classifies correctly 58.33 and 61.67% of the firms for years 2 and 3 prior to the acquisition, respectively. The authors argue that this poor classification could be attributed to the difficulty of predicting acquisition targets in general, and not necessarily to the inability of the proposed approach as a discrimination method. To test further the proposed technique, its classification accuracy was compared with that of DA and UTADIS. The correct classification accuracy obtained using the proposed method is better for all years than that obtained using DA. As opposed to the UTADIS method, the classification accuracy under the proposed approach is significantly higher for year −1, the same for for year −2, and slightly higher for year −3. On the basis of these results the authors conclude that the iterative binary segmentation procedure is able to provide results that are at least favorably comparable with those provided by UTADIS and outperforms DA.

Tartari et al. [35] also used UTADIS in their study along with linear DA (LDA), probabilistic neural networks (PNN), and RS in an attempt to examine whether the integration of different methods using a stacked generalization approach could result in higher classification accuracies. Their sample consisted of 48 UK firms, selected from 19 industries/sectors, acquired during 2001, and 48 nonacquired firms matched by principal business activity, asset size, sales volume, and number of employees. Twenty-three financial ratios measuring profitability, liquidity and solvency, and managerial performance were initially calculated for each firm for up to 3 years prior to the acquisition (1998–2000); however, they finally used a set of nine ratios, selected on the basis of factor analysis. Their exercise consisted of two stages. First, UTADIS, LDA, PNN, and RS were used to develop individual models. The most recent year (i. e., 2000) was used as a training sample, while data from the other two years (i. e., 1998 and 1999) were used to test the generalizing performance of the proposed integration approach. An eightfold cross-validation approach was employed to develop the base models using the four methods. The classifications of the firms obtained were then used as a training sample for the development of a stacked generalization model. Finally, the development of the stacked model was performed using the UTADIS method that combines (at a metalevel) the group assignments of all the four methods considered in the analysis. The use of other methods to develop the combined model was also examined; nevertheless the results are inferior to those obtained with UTADIS. The stacked model performs better (in terms of the overall correct accuracy rate) than any of the four methods upon which it is based, throughout all the years of the analysis. Furthermore, the results indicate that the stacked model provides significant reductions in the overall error rate compared with LDA, RS, and UTADIS, although they were less significant compared with PNN.

In another UK study, Doumpos et al. [13] compared the classification ability of UTADIS against one of the models developed using DA, logistic regression (LR), and ANN. The sample included 76 UK firms acquired during 2000–2002, matched by industry and size with 76 nonacquired firms. Twenty-nine financial ratios were initial candidates for model development, representing profitability, efficiency, activity, financial leverage, liquidity, and growth; however, the authors finally selected six variables on the basis of a *t* test and correlation analysis. The UTADIS model was first developed using data drawn from the most recent year prior to the acquisition (i. e., year −1). The model developed was then applied to data from 2 and 3 years prior to the acquisition (years −2 and −3). The average accuracies were 74.34 and 78.95%, respectively. These accuracies are higher than the ones obtained by both DA and LR, and are found to comparable to or better than those of ANN when tested using data from years −2 and −3.

Pasiouras et al. [26] used both UTADIS and MHDIS, among several other classification techniques, to develop models specifically designed for the EU banking industry. They developed several models on the basis of equal and unequal training samples from the period 1998–2000, using both raw and country-adjusted variables. The models were tested in equal and unequal datasets from a future period (2001–2002). They also developed models that combine the predictions of the individual models developed in the first stage, using two integration techniques, namely, stacked generalization and majority voting. Their results were mixed and depended on the form of the variables used, the datasets, and the evaluation measure considered. Hence, they concluded that there is no clear winner technique that dominates all the others under all circumstances. However, UTADIS appears several times as one of the best techniques. Furthermore, the stacked model developed through UTADIS also performs relatively well.

Pasiouras et al. [27] also focused on the EU banking sector, but differentiated their study in two ways from that of Pasiouras et al. [26]. First, they considered an additional MCDA technique, namely, PAIRCLAS, that was applied for the first time in the prediction of acquisitions. Second, they followed a tenfold cross-validation resampling procedure for the development and evaluation of the models. Their sample consisted of 168 banks acquired between 1998 and 2002 matched with 168 nonacquired banks. MHDIS achieved the highest overall accuracy in the validation dataset, with 68% of the acquired and 63.3% of the nonacquired banks classified correctly (implying an overall classification rate of 65.7%). PAIRCLAS also achieves marginally better classification accuracies than UTADIS, and its ability to classify correctly the nonacquired banks (75%) is even higher than that of MHDIS (72.2%).

In another study, Pasiouras et al. [25] concentrated on the Asian banking sector. They used a sample of 52 targets and 47 acquirers that were involved in acquisitions in nine Asian banking markets during 1998–2004 and matched them by country and time with an equal number of banks not involved in acquisitions. The models were developed and validated through a tenfold cross-validation approach using UTADIS and MHDIS. In each case three versions of the model were developed. The first one distinguished between acquired and noninvolved banks. The second one distinguished between acquirers and noninvolved banks. The last one, was a three-outcome model that simultaneously distinguished between targets, acquirers, and noninvolved banks. For comparison purposes they also developed models through DA. The results indicate that the MCDA models are more efficient that the ones developed through DA. Furthermore, in all cases the models are more efficient in distinguishing between acquirers and noninvolved banks than between targets and noninvolved banks. Finally, the models with a binary outcome achieve higher accuracies than the ones which simultaneously distinguish between acquirers, targets, and noninvolved banks.

Formulation

The problem considered in the present study is a classification one that in general involves the assignment of a set of m alternatives $A = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m\}$, evaluated along a set of n criteria $g_1, g_2, \ldots, g_n$, to a set of q classes $C_1, C_2, \ldots, C_q$. In the case of acquisitions, the alternatives are the firms in the sample, the criteria can correspond to financial and nonfinancial variables, and there are usually two classes, the nonacquired firms (class C_1) and the acquired firms (class C_2). Hence, in what follows we consider the simple two-class case, while details on the multiclass case can be found in Doumpos and Zopounidis [14] and Zopounidis and Doumpos [39].

The UTADIS approach, used in the present study, implies the development of an additive utility function that is used to score the firms and decide upon their classification. The utility function has the following general form:

$$U(\mathbf{a}) = \sum_{i=1}^{n} w_i u_i'(g_i) \in [0,\ 1]\ , \tag{1}$$

where w_i is the weight of criterion g_i (the criteria weights sum up to 1) and $u'_i(g_i)$ is the corresponding marginal utility function normalized between 0 and 1. The marginal utility functions provide a mechanism for decomposing the aggregate result (global utility) in terms of individual assessment to the criterion level. To avoid the estimation of both the criteria weights and the marginal utility functions, it is possible to use the transformation $u_i(g_i) = w_i u_i'(g_i)$. Since $u'_i(g_i)$ is normalized between 0 and 1, it becomes obvious that $u_i(g_i)$ ranges in the interval $[0, w_i]$. In this way, the additive utility function is simplified to the following form:

$$U(\mathbf{a}) = \sum_{i=1}^{n} u_i(g_i) \in [0,\ 1]\ . \tag{2}$$

The utility function developed provides an aggregate score $U(\mathbf{a})$ of each firm along all criteria. In the case of acquisitions prediction, this score provides the basis for determining whether the firm could be classified in either the group of nonacquired ones or in the group of acquired ones. The classification rule in this case is the following (C_1 and C_2 denote the group of nonacquired and acquired firms, respectively, while u_1 is a cutoff utility point defined on the global utility scale, i. e., between 0 and 1):

$$\left.\begin{array}{ll} U(\mathbf{a}) \geq u_1 & \Rightarrow \mathbf{a} \in C_1 \\ U(\mathbf{a}) < u_1 & \Rightarrow \mathbf{a} \in C_2 \end{array}\right\}\ . \tag{3}$$

The estimation of the additive value function and the cutoff threshold is performed using linear programming techniques so that the sum of all violations of the classification rule (3) for all the firms in the training sample is minimized. A detailed description and derivation of this mathematical programming formulation can be found in Doumpos and Zopounidis [14].

Cases

In this section, our method is illustrated by a case study from the work of Pasiouras et al. [24]. The dataset considered in the study consists of 76 firms acquired between 2000 and 2002, and 76 nonacquired firms, which operate in manufacturing, construction, and mining–quarrying–extraction industries in the UK. The sample was constructed as follows. The acquired firms were first identified in the Hemscott M&As database and the financial data were collected from the Financial Analysis Made Easy database of Bureau van Dijk. After screening for data availability in FAME, 59 manufacturing, six construction, five production and six mining–quarrying–extraction firms had complete financial data for the 3 years prior to the acquisition and were included in the sample.

Although the year of acquisition is not common for all firms in the sample, they were all thought to be acquired in the "zero" year, considered as the year of reference. The years of activity prior to "zero" are coded as "year −1" (1 year prior), "year −2" (2 years prior), and "year −3" (3 years prior).

After the sample described above had been obtained, nonacquired firms were chosen to match the acquired firms. The firms were matched by industry and size (total assets) and financial data for the nonacquired companies were taken from the same calendar years as for the corresponding acquired companies.

Barnes [5] mentions that the problem for the analyst who attempts to forecast targets is simply a matter of identifying the best predictive (i. e., explanatory) variables. Unfortunately, financial theories do not offer much in selecting specific variables among the numerous ones regarded as potential candidates in model development. Given the large number of possible ratios, it is important to reduce the list of ratios that enter the final model selection process. Hence, a question that emerges when attempting to select accounting ratios for empirical research is which ones, among the hundreds, should be used? However, there is no easy way to determine how many ratios a particular model should contain. Too few and the model will not capture all the relevant information. Too many and the model will overfit the training sample, but underperform in a holdout sample, and will most likely have onerous data input requirements [21]. As Hamer [17] points out, the variable set should be constructed on the basis of (1) minimizing the cost of data collection and (2) maximizing the model applicability. Huberty [18] suggests three variable screening techniques that could be used: logical screening (e. g., financial theory and human judgment), statistical screening (e. g., test of differences of two group means such as the *t* test), and dimension reduction (e. g., factor analysis). In the present study we follow the latermost approach as in Stevens [34], Barnes [2], Kira and Morin [20], Zanakis and Zopounidis [38], and Tartari et al. [35]. Hence, a total of 25 variables are initially considered on the basis of data availability and previous studies, covering several aspects of firms' performances such as profitability, efficiency, activity, financial leverage, liquidity, and growth. Factor analysis is then used to reduce the number of variables to a smaller number of factors that are linear combinations of the initial variables. The analysis results in the extraction of seven factors, with eigenvalues higher than 1. The variable with the highest loading is selected from each one of the seven components for inclusion in the classification models. Consequently, we use the following seven variables:

1. X1: Current assets/current liabilities,
2. X2: Total liabilities/shareholders' equity,
3. X3: Annual change of total assets,
4. X4: Annual change of current liabilities,
5. X5: Profits before taxes/total assets,
6. X6: Sales/stock,
7. X7: Sales/debtors.

X1 is an indicator of liquidity that has been used in many previous studies [2,12,20,38]. The views about liquidity are somewhat mixed. It is possible that firms with excess liquidity are more likely to be acquired because of their good short-term financial position and the availability of cash or near-cash assets [36]. In this case, there is also an opportunity for the acquirers to finance the acquisition with the target's own resources [32]. On the other hand, it can be argued that a firm in need of funds to finance its working capital requirements is likely to be an acquisition target because the acquirer, after the acquisition, expects to bring additional funds into the firm to improve its liquidity [29].

X2 is a measure of financial leverage that has been used as a proxy for financial leverage in Rege [29], Palepu [23], and Kim and Arbel [19] among others. According to the *financial leverage hypothesis* the likelihood of being acquired decreases with the increase in company debt. There are two reasons why firms with lower preexisting levels of debt are considered attractive acquisition targets. The first is that the low debt ratio of the target decreases the probability of future default of the joint firm, while at the same time it increases the debt capacity of the new firm. The second is that in some cases a firm has extremely low debt ratios, the value of the firm may not be maximized, and low leverage can be seen as a sign of inefficient management.

X3 and X4 are measures of annual changes in two basic elements of the firms (i. e., assets, liabilities). Firms whose growth rates, as measured by X3, are relatively high can experience problems because their management and/or structure will not able to deal with and sustain exceptional growth. It is therefore possible that a firm which is constrained in this way will become an acquisition target of a firm with surplus resources or management available to help [14]. Furthermore, a firm with high levels of growth might be acquired by firms that what to take advantage of this increase in assets, and boost their own growth. Turning to X4, exceptional increases may indicate that the firm has problems in meeting its short-term liabilities and can therefore be acquired to avoid solvency.

Variables X5, X6, and X7 are related to the *inefficient management hypothesis*. This hypothesis argues that if the managers of a firm fail to maximize its market value, then the firm is likely to be an acquisition target and inefficient managers will be replaced. Thus, these takeovers are motivated by a belief that the acquiring firm's management can manage better the target's resources. This view is supported by two specific arguments. First, the firm might be poorly run by its current management, partly because the objectives of the management are at variance with those of the shareholders. In this case, the takeover threat can serve as a control mechanism limiting the degree of variance between management's pursuits for growth from shareholders' desire for wealth maximization. A merger may not be the only way to improve management, but if disappointed shareholders cannot accomplish a change in management that will increase the value of their investment within the firm, either because it is too costly or too slow, then a merger may be a simpler and more practical way of achieving their desired goals. Second,

Multicriteria Methods for Mergers and Acquisitions, Table 1 Weights of the variables (percent) in the *utilités* additives discriminantes (UTADIS) model (averages over the ten replications)

	Year −1	Year −2	Year −3
X1	11.15	2.45	10.40
X2	23.28	34.04	31.99
X3	5.15	7.43	8.80
X4	15.79	5.71	8.59
X5	14.67	25.08	12.22
X6	14.40	11.31	14.77
X7	15.56	13.97	13.23

the acquirer may simply have better management experience than the target. There are always firms with unexploited opportunities to cut costs and increase sales and earnings, and that makes them natural candidates for acquisition by other firms with better management [1]. Therefore, if the management of the acquirer is more efficient than the management of the target firm, a gain could result through a merger if the management of the target is replaced.

Table 1 presents the contribution of the seven criteria in the UTADIS model. To ensure the proper development and validation of the models, we follow a tenfold cross-validation. Hence, the total sample of 152 firms is randomly split into ten mutually exclusive subsamples (folds) of approximately equal size. Then ten models are developed, using each fold in turn for validation and the remaining folds for training. Therefore, in each of the ten replications, the training sample consists of 137 firms and the validation of 15 firms. The figures presented are the averages over the ten replications.

X2 (total liabilities/shareholders' equity) appears to be the most important criterion in all 3 years with an average weight that ranges between 23.28 (year −1) and 34.04% (year −3). The profitability and efficiency indicators (X5, X6, X7) also appear to be important in classifying firms within the two groups, with average weights between 11.31 and 25.08%. X1 (i. e., current assets ratio) carries a weight above 10% in years −1 and −3 but it is considerably reduced to 2.45% in year −2. Finally, X3, which corresponds to the annual growth of the firm in terms of total assets, is the least important criterion in all years.

Multicriteria Methods for Mergers and Acquisitions, Table 2 Classification accuracies in percent (averages over ten replications)

	Acquired firms	Nonacquired firms	Overall accuracy
Classification accuracies of the UTADIS model in the development stage			
Year −1	80.1	71.8	75.9
Year −2	81.9	71.1	76.5
Year −3	81.1	70.3	75.7
Classification accuracies in the validation stage			
Year −1			
UTADIS	76.2	63.9	70.1
DA	77.9	54.0	65.9
Year −2			
UTADIS	75.3	65.5	70.4
DA	77.4	47.2	62.3
Year −3			
UTADIS	77.3	63.1	70.2
DA	65.5	50.4	58.0

By comparing the score $U(\mathbf{a})$ of each firm with the cutoff threshold that was calculated through the estimation of the UTADIS model and rule (3), we can decide whether a firm can be classified as acquired or not acquired. Table 2 presents the classification results obtained by UTADIS during the development and validation process. In Table 2 we also present the classification results obtained by DA, used for benchmarking purposes.

The overall classification accuracy of the UTADIS model during the development stage is around 75%. Furthermore, the model appears to be quite robust, with classifications that do not deviate significantly from one year to another. Unsurprisingly, consistent with previous studies, the classification accuracy decreases in the validation stag; however, the decrease is relatively small and the overall classification accuracy is now around 70%. It should be mentioned that while our model misclassifies around 30% of the firms in the validation dataset, this is not uncommon for studies on the prediction of acquisitions targets.

Other studies that used resampling techniques obtained similar results. Bartley and Boardman [7] reported a classification accuracy of 64%, while in a later study [8] they obtained classification accuracies between 69.9 and 79.9%. Similarly, the classification accuracy in the study of Kira and Morin [20] was 66.17%. The study of Pasiouras et al. [27] that focused on the EU banking industry also reported classification accuracies between 61.6 and 65.7%. As Barnes [14] notes, perfect prediction models are difficult to develop even for bankruptcy prediction, where failing firms have definitely inferior or abnormal performance compared with healthy firms. The problem with the identification of acquisition targets is not only that there are potentially many reasons for acquisitions, but also that at the same time managers do not always act in a manner which maximizes shareholders' returns owing to hybris or agency motives.

While the comparison of the results obtained in the current study with the ones of previous studies gives a first indication for the performance of the model, a direct comparison is not appropriate because of differences in the datasets [16,21], the industry under investigation, the methods used to validate the models, and so on. Hence, the comparison of the UTADIS model with the one developed with DA using exactly the same dataset, variables, and development and validation procedures might provide a more accurate indication of the efficiency (in terms of classification accuracy) of the MCDA model. Looking at the results in Table 2, we see that UTADIS clearly achieves higher classification accuracies than DA. Furthermore, while the classification accuracies of DA decrease as we move back in time, the accuracies of UTADIS remain quite robust, even when we use data from 3 years prior to the acquisition. Finally, with the exception of acquired firms in year −1, UTADIS outperforms DA in classifying correct firms of both groups (i. e., acquired, nonacquired).

Conclusions

In this paper we first discussed why MCDA could be useful in the prediction of acquisition targets and provided a review of relevant studies. Then, we presented the UTADIS technique and its application on a dataset of acquired and nonacquired UK firms.

The application indicates that UTADIS not only outperforms a model developed by DA, but it also achieves quite robust results, as we use data that move away from the period of the event.

Future applications of MCDA in the area of acquisitions prediction could focus on the incorporation of nonfinancial and qualitative data (e. g. managers' experience, managers' educational background) in the analysis. Although this has been mentioned in the literature in the past [22,38], there is still a lack of studies that use such variables in the analysis, usually owing to data availability. MCDA techniques, like UTADIS, can easily incorporate qualitative data, and it would be therefore interesting to perform such an exercise. Furthermore, it would also be worthwhile to investigate the classification of firms in more than two groups (e. g., acquired, acquirers, noninvolved) as in the study of Pasiouras et al. [25]. While the results of the later study were not promising, the study focused on the banking industry, which is a special case. Hence, results from nonfinancial sectors might lead to different conclusions. MCDA techniques, like MHDIS, which was developed with the multigroup discrimination in mind, might be useful in such applications.

References

1. Arnold G (1998) Corporate Financial Management, 1st edn. Financial Times Prentice Hall, London
2. Barnes P (1990) The prediction of takeover targets in the U.K. by means of multiple discriminant analysis. J Bus Finance Account 17:73–84
3. Barnes P (1998) Can takeover targets be identified by statistical techniques?: Some UK evidence. Stat 47:573–591
4. Barnes P (1999) Predicting UK Takeover Targets: Some Methodological Issues and an Empirical Study. Rev Quant Finance Account 12:283–301
5. Barnes P (2000) The identification of U.K. takeover targets using published historical cost accounting data. Some empirical evidence comparing logit with linear discriminant analysis and raw financial ratios with industry-relative ratios. Int Rev Financial Analys 9:147–162
6. Barniv R, McDonald JB (1999) Review of Categorical Models for Classification Issues in Accounting and Finance. Rev Quant Finance Account 13:39–62
7. Bartley JW, Boardman CM (1986) Replacement-Cost-Adjusted Valuation Ratio as a Discriminator Among Takeover Target and Nontarget Firms. J Econom Bus 38:41–55
8. Bartley JW, Boardman CM (1990) The relevance of inflation adjusted accounting data to the prediction of corporate takeovers. J Bus Finance Account 17:53–72
9. Belkaoui A (1978) Financial ratios as predictors of Canadian takeovers. J Bus Finance Account 5:93–107

10. Cheh JJ, Weinber RS, Yook KC (1999) An Application Of An Artificial Neural Network Investment System To Predict Takeover Targets. J Appl Bus Res 15:33–45
11. Cudd M, Duggal R (2000) Industry Distributional Characteristics of Financial Ratios: An Acquisition Theory Application. Financial Rev 41:105–120
12. Dietrich JM, Sorensen E (1984) An Application of Logit Analysis to Prediction of Merger Targets. J Bus Res 12:393–402
13. Doumpos M, Kosmidou K, Pasiouras F (2004) Prediction of Acquisition Targets in the UK: A Multicriteria Approach. Oper Res Int J 4:191–211
14. Doumpos M, Zopounidis C (2002) Multicriteria Decision Aid Classification Methods. Kluwer, Dordrecht
15. Espahbodi H, Espahbodi P (2003) Binary choice models for corporate takeover. J Bank Finance 27:549–574
16. Gupton GM, Stein RM (2002) LossCalc^{TM}: Moody's Model for Predicting Loss Given Default (LGD). Moody's Investors Service Global Credit Research, Special Comment, February
17. Hamer MM (1983) Failure prediction: Sensitivity of classification accuracy to alternative statistical methods and variable sets. J Account Public Policy 2:289–307
18. Huberty CJ (1984) Applied Discriminant Analysis. Wiley, New York
19. Kim WG, Arbel A (1998) Predicting merger targets of hospitality firms (a Logit model). Int J Hosp Manag 17:303–318
20. Kira D, Morin D (1993) Prediction of takeover targets of Canadian Firms. In: Janssen J, Skiadas CH (eds) Applied Stochastic Models and Data Analysis. World Scientific Publ. Co., Singapore, pp 507–518
21. Kocagil AE, Reyngold A, Stein RM, Ibarra E (2002) Moody's RiskCalc^{TM} Model For Privately-Held U.S. Banks, Moody's Investors Service. Global Credit Research, July
22. Meador AL, Church PH, Rayburn LG (1996) Development of prediction models for horizontal and vertical mergers. J Financial Strateg Decis 9:11–23
23. Palepu KG (1986) Predicting Takeover Targets: A Methodological and Empirical Analysis. J Account Econ 8:3–35
24. Pasiouras F, Doumpos M, Zopounidis C (2006) Predicting acquisitions: Methodological Framework and Applications. Klidarithmos Publications (In Greek), Athens
25. Pasiouras F, Gaganis C, Zopounidis C (2007) Classification models for the detection of acquisition targets in the Asian banking sector, Paper presented at the 22nd European Conference on Operational Research, Prague, July 8–11
26. Pasiouras F, Tanna S, Zopounidis C (2005) Application of Quantitative Techniques for the Prediction of Bank Acquisition Targets. World Scientific Publishing Co, Singapore
27. Pasiouras F, Tanna S, Zopounidis C (2007) The identification of acquisition targets in the EU banking industry: an application of multicriteria approaches. Int Rev Financial Analys 16:262–281
28. Powell RG (2001) Takeover Prediction and Portfolio Performance: A Note. J Bus Finance Account 28:993–1011
29. Rege UP (1984) Accounting ratios to locate take-over targets. J Bus Finance Account 11:301–311
30. Simkowitz M, Monroe RJ (1971) A discriminant analysis function for conglomerate mergers. South J Bus 38:1–16
31. Slowinski R, Zopounidis C, Dimitras AI (1997) Prediction of company acquisition in Greece by means of the rough set approach. Eur J Oper Res 100:1–15
32. Song MH, Walkling RA (1993) The impact of managerial ownership on acquisition attempts and target shareholder wealth. J Financial Quant Analys 12:439–457
33. Steuer RE, Na P (2003) Multiple criteria decision making combined with finance: A categorized bibliographic study. Eur J Oper Res 150:496–515
34. Stevens DL (1973) Financial Characteristics of Merged Firms: A Multivariate Analysis. J Financial Quant Analys 8:149–158
35. Tartari E, Doumpos M, Baourakis G, Zopounidis C (2003) A stacked generalization framework for the prediction of corporate acquisitions. Foundat Comput Decis Sci 28:41–61
36. Tzoannos J, Samuels JM (1972) Mergers and takeovers: the financial characteristics of companies involved. J Bus Finance 4:5–16
37. Walter RM (1994) The Usefulness of Current Cost Information for Identifying Takeover Targets and Earning Above-Average Stock Returns. J Account Auditing Finance 9:349–377
38. Zanakis SH, Zopounidis C (1997) Prediction of Greek company takeovers via multivariate analysis of financial ratios. J Oper Res Soc 48:678–687
39. Zopounidis C, Doumpos M (2002) Multi-group discrimination using multi-criteria analysis: Illustrations from the field of finance. Eur J Oper Res 139:371–389

Multicriteria Sorting Methods

CONSTANTIN ZOPOUNIDIS, MICHAEL DOUMPOS
Department Production Engineering and Management, Financial Engineering Lab. Techn. University Crete University Campus, Chania, Greece

MSC2000: 90C29

Article Outline

Keywords

Sorting; Multicriteria analysis; Goal programming; Outranking relation; Preference disaggregation

Decision making problems, according to their nature, the policy of the decision maker, and the overall objective of the decision may require the choice of an alternative solution, the ranking of the alternatives from the best to the worst ones or the *sorting* of the alternatives in predefined homogeneous classes [30]. For instance, a decision regarding the location of a new power plant can be considered as a choice problem, since the objective is to select the most appropriate location according to environmental, social and investment criteria. On the other hand, an evaluation of the efficiency of the different units of a firm can be considered as a ranking problem, since the objective is to estimate the relative performance of each unit compared to the others. Finally, a credit granting decision is a sorting problem: a credit application can be accepted, rejected or submitted for further consideration, according to the business and personal profile of the applicant. Actually, a wide variety of decision problems, including financial and investment decisions, environmental decisions, medical decisions, etc., are better formulated and studied through the sorting approach.

The sorting problem, generally stated, involves the assignment of a set of observations (objects, alternatives) described over a set of attributes or criteria into predefined homogeneous classes. This type of problem can also referred to as the *'discrimination'* problem or the *'classification'* problem. Although any of these three terms can be used to describe the general objective of the problem (i. e. the assignment of observations into groups), actually, they refer to two slightly different situations: the discrimination or classification problem refers to the assignment of observations into classes which are not necessarily ordered. On the other hand, sorting refers to the problem in which the observations should be classified into classes which are ordered from the best to the worst ones. For instance, in medical diagnosis the classification of patients according to their symptoms into several possible diseases is a discrimination (classification) problem, since it is impossible to establish a preference ordering between the diseases. On the contrary, the evaluation of bankruptcy risk is a sorting problem, since the non-bankrupt firms are preferred to the bankrupt ones. In this paper the terms 'discrimination', 'classification', and 'sorting' will be used without distinction to refer to the general problem of assigning observations, objects or alternatives into classes.

The major practical interest of the sorting problem, has motivated researchers in developing an arsenal of methods for studying such problems, with the aim being the development of quantitative models achieving the higher possible classification accuracy and predicting ability. In 1936, R.A. Fisher [8] was the first to propose a framework for studying classification problems taking into account their multidimensional nature. The linear discriminant analysis (LDA) that Fisher proposed has been used for decades as the main classification technique and it is still being used at least as a reference point for comparing the performance of new techniques that are developed. C. Smith in 1947 [34] extended Fisher's linear discriminant analysis proposing quadratic discriminant analysis (QDA) in order to overcome the restrictive assumption underlying LDA that groups have equal dispersion matrices. Later on, several other statistical classification approaches have been proposed. Among them logit and probit analysis are the most widely used techniques overcoming the multivariate normality assumption of discriminant analysis (both linear and quadratic). Although these techniques overcome most of the statistical restrictions imposed in discriminant analysis, their parameters are difficult to explain, especially in multigroup discriminant problems.

The continuous advances in other fields including operations research and artificial intelligence led many scientists and researchers to exploit the new capabilities of these fields, in developing more efficient classification techniques. Among the attempts made one can mention neural networks, machine learning, fuzzy sets as well as *multicriteria decision aid* (MCDA). This article will focus on MCDA and its application in the study of classification problems with or without ordered classes. MCDA provides an arsenal of powerful and efficient nonparametric classification methods and approaches, which are free of statistical assumptions and restrictions, while furthermore they are able to incorporate the decision maker's preferences in a flexible and realistic way.

The remainder of the article is organized as follows. Section 2 provides a review of MCDA sorting approaches and techniques, outlining their basic characteristics, concepts and limitations. In section 3, a new MCDA sorting method is described and its operation is depicted through a simple illustrative example. Finally, section 4 concludes the paper and outlines some possible future research directions concerning the application of MCDA in sorting problems.

Multicriteria Sorting Methods

The MCDA methods which have been proposed for the study of sorting problems can be distinguished either according to the approach from which they are originated (multi-objective/goal programming, multiattribute utility theory, outranking relations, preference disaggregation), or according to the type of problem that they address (ordered or non-ordered classes). The review presented in this section will distinguish the methods according to their origination, but in the same time the type of problems that they address will also be discussed.

Goal Programming Approaches

The work of A. Charnes and W.W. Cooper [4] set the foundations on goal/multi-objective programming, but it can also be considered as one or the pioneering studies in the field of MCDA in general. Since then, both multi-objective and *goal programming* constitute two major fields of interest from the theoretical and practical points of view in the MCDA and operations research communities. In particular, goal programming approaches, during the 1960s and the 1970s have been used to elicit attribute weights in multiple criteria ranking decision problems [15,27,35,36]. N. Freed and F. Glover [9] were among the first to investigate the potentials of goal programming techniques in the discriminant problem. Their aim was to develop a linear discriminant model so that the minimum distance of the score of each alternative from a predefined cut-off point is maximized (maximize the minimum distance-MMD). To develop this model, they proposed the following goal programming formulation:

$$\begin{cases} \max & d \\ \text{s.t.} & \sum w_i x_{ij} + d \le c, \quad \forall i \in \text{Group } 1, \\ & \sum w_i x_{ij} - d \ge c, \quad \forall i \in \text{Group } 2, \end{cases}$$

where w_i is the weight of attribute i, x_{ij} is the evaluation of alternative j on attribute i, and c is the cut-off score (w_i and d are unrestricted in sign). Soon after proposing this model, the same authors proposed a variety of similar goal programming formulations incorporating several other discrimination criteria, such as the sum of deviations (optimize the sum of deviations-OSD), the sum of interior deviations (minimize the sum of interior deviations-MSID) and the maximum deviation [10].

These two studies attracted the interest of several operational researchers and management scientists. S.M. Bajgier and A.V. Hill [2] proposed a new goal programming approach in order to minimize the number of misclassifications using a mixed integer programming formulation (MIP) and conducted a first experimental study to compare the MMD model, the OSD model, and their MIP formulation with LDA. They concluded that the goal programming formulations are generally superior to LDA, except for the case of moderate to low overlap between groups and equal dispersion matrices, where LDA outperforms all the examined goal programming formulations.

The performance of goal programming approaches compared to statistical techniques was an issue that several researchers tried to investigate using mainly experimental data sets. Freed and Glover [11] compared MMD, MSID, OSD and LDA and they concluded that although the presence of outliers pose a greater problem for the two simpler goal programming formulations (MMD and MSID) than for LDA, generally the goal programming approaches outperform LDA.

E.A. Joachimsthaler and A. Stam [18] compared the LDA, QDA, logistic regression and OSD procedures and they concluded that these methodologies produce similar results although the misclassification rates for LDA and QDA tended to increase with highly kurtosis data and increased dispersion heterogeneity. C.A. Markowski and E.P. Markowski [22] examined the influence of qualitative attributes on the discriminating performance of MMD and LDA. Although the incorporation of qualitative attributes in LDA violates the normality assumption, the experimental study of the authors showed that the incorporation of qualitative variables improved the performance of LDA, while on the other hand MMD did not appear to be particularly well-suited for use with qualitative variables. In another experimental study conducted by P.A. Rubin [32], QDA outperformed 15 goal programming approaches, leading the author to indicate that 'if LP models are to be considered seriously as an alternative to conventional procedures, they must be shown to outperform QDA under plausible conditions, presumably involving non-Gaussian data'. These experimental studies clearly indicate the confusion concerning the discriminating performance of the goal programming formulations as opposed to well known multivariate statistical techniques. Except for this issue, the research on the field of goal programming approaches for discriminant problems, was also focus on the theoretical drawbacks which were often meet. Markowski and Markowski [23] were the first to identify two major drawbacks of the goal programming formulations (MMD and OSD) proposed by Freed and Glover [9,10]. More specifically, they proved that if each quadrant contains at least one case from the second group, unacceptable solutions will result in MMD (all coefficients in the discriminant function are zeros which leads all the observations to be classified in the same group), while furthermore they showed that the solutions (discriminant functions) obtained through the MMD and the OSD models are not stable when the data are transformed (when there is a shift from the origin). Except for these two problems, many goal programming formulations were found to suffer from two additional theoretical shortcomings [29]:

a) they produce unbounded solutions, and
b) they produce improper solutions.

A solution is considered *unbounded* if the objective function can be increased or decreased without limit, in which case the discrimination rule (function) may be meaningless, whereas a solution is improper if all observations fall on the classification hyperplane.

To overcome these problems new goal programming formulations were proposed, including hybrid models [12,13], nonlinear programming formulations [37], as well as several mixed integer programming formulations [1,3,5,20,33,38,39].

In the light of this review of goal programming approaches for discriminant problems it is possible to identify the following three characteristics of the research in this field:

1) The majority of the proposed models aim at developing a linear discrimination rule (function). The extension of the models to develop a nonlinear discriminant function leads to nonlinear programming formulations which are generally computationally intensive and difficult to solve. Among the few alternative approaches is the MSM method (multisurface method) proposed by O.L. Mangasarian [21] that leads to the construction of a piecewise linear discrimination surface between two groups (see also [26] for a revision of the method using multi-objective programming and fuzzy mathematical programming techniques).
2) Little research has been made on extending the existing framework on the multigroup discriminant problem. E.-U. Choo and W.C. Wedley [5], W. Gochet et al. [14], as well as J.M. Wilson [39] applied goal programming approaches in multigroup discriminant problems, but generally most of the studies in this field were focused on two-group discrimination trying to extend the original goal programming models of Freed and Glover [9,10] in order to achieve higher classification accuracy and predicting ability.
3) The models based on the goal programming approach can be applied in any classification problem with or without ordered classes.

Outranking Relations Approaches

In contrast to the goal programming approaches, *outranking relations* procedures study the classification problem on a completely different basis. The aim of such procedures is not to develop a discriminant function (linear or nonlinear), but instead their aim is to model the decision makers' preferences and develop a global preference model which can be used to assign the alternatives (observations) into the predefined classes. To achieve the classification of the alternatives some reference profiles are determined which can be considered as representative examples of each class. Through the comparison of each alternative with these reference profiles the classification of the alternatives is accomplished.

A representative example of MCDA sorting method based on the outranking relations approach is the ELECTRE TRI method proposed by W. Yu [40]. The aim of ELECTRE TRI is to provide a sorting of the alternatives under consideration into two or more ordered categories. In order to define the categories ELECTRE TRI uses some reference alternatives (reference profiles) r_i, $i = 1, \ldots, k-1$, which can be considered as fictitious alternatives different from the alternatives under consideration. The profile r_i is the theoretical limit between the categories C_i and C_{i+1}(C_{i+1} is preferred to C_i) and r_i is strictly better than r_{i-1} for each criterion. To provide a sorting of the alternatives in categories ELECTRE TRI makes comparisons of each alternative with the profiles.

For an alternative a and a profile r_i the concordance index $c_j(a, r_i)$ is calculated. This index expresses the strength of the affirmation 'alternative a is at least as good as profile r_i on criterion j'. In order to compare the alternative to a reference profile on the basis of more than one criteria, a global concordance index $C(a, r_i)$ is calculated. This index expresses the strength of the affirmation 'a is at least as good as r_i according to all criteria'. Setting w_j as the weight of the criterion j, $C(a, r_i)$ is constructed as the weighted average of all $c_i(a, r_i)$.

In contrast to the concordance index, the discordance index $D_j(a, r_i)$ expresses the strength of the opposition to the affirmation 'alternative a is at least as good as profile r_i according to criterion g_j'. The calculation of the discordance index is based on the definition of a veto threshold $v_j(r_i)$ for criterion j and the profile r_i. The veto threshold $v_j(r_i)$ for criterion j defines the minimum accepted difference between the values of the profile r_i and alternative a on the specific criterion so that we can say that they have totally different preference according to criterion j.

Let $\overline{F}(a, r_i)$ be the set consisted of all criteria for which the discordance index value is greater than the value of global concordance index. For each affirmation of the type: 'alternative a outranks profile r_i according to all criteria', the credibility index $\sigma_s(a, r_i)$ is calculated. If $\overline{F}(a, r_i)$ is empty then $\sigma_s(a, r_i) = C(a, r_i)$, otherwise the credibility index is calculated as follows:

$$\sigma_s(a, r_i) = C(a, r_i) \cdot \prod_{j \in \overline{F}} \frac{1 - D_j(a, r_i)}{1 - C(a, r_i)}.$$

If the value of the credibility index of the affirmation 'alternative a outranks profile r_i according to all criteria} exceeds a predefined cut-off value λ, then the proposition 'a outranks r_i' can be considered to be valid. Denoting the outranking relation as **S**, the preference (**P**), indifference (**I**) and incomparability (**R**) relations between alternative a and profile r_i can be defined as follows:

- $a\mathbf{I}r_i$ if and only if $a\mathbf{S}r_i$ and $r_i\mathbf{S}a$;
- $a\mathbf{P}r_i$ if and only if $a\mathbf{S}r_i$ and no $r_i\mathbf{S}a$;
- $r_i\mathbf{P}a$ if and only if no $a\mathbf{S}r_i$ and $r_i\mathbf{S}a$;
- $a\mathbf{R}r_i$ if and only if no $a\mathbf{S}r_i$ and no $r_i\mathbf{S}a$.

According to these relations two sorting procedures are applied: the pessimistic and the optimistic one. The sorting procedure starts by comparing alternative a to the worst profile r_1 and in the case where $a\mathbf{P}r_1$, a is compared to the second profile r_2, etc., until one of the following two situations appears:

i) $a\mathbf{P}r_i$ and $r_{i+1}\mathbf{P}a$ or $a\mathbf{I}r_{i+1}$;

ii) $a\mathbf{P}r_i$ and $a\mathbf{R}r_{i+1}, \ldots, a\mathbf{R}r_{i+k}, r_{i+k+1}\mathbf{P}a$.

If situation i) appears, then alternative a is assigned to category $i + 1$ by both pessimistic and optimistic procedures. If situation ii) appears, then a is assigned to category $i + 1$ by the pessimistic procedure and to category $i + k + 1$ by the optimistic procedure.

It is clear that the ELECTRE TRI method is a powerful tool for analyzing the decision maker's preference in sorting problems involving multiple criteria where the classes are ordered. However, the major drawback of the method is the significant amount of information that it requires by the decision maker (weights of the criteria, preference and indifference thresholds, veto thresholds, etc.). This problem can be overcame using decision instances (assignment examples) as proposed in [25].

Other MCDA sorting methods based on the outranking relations approach have been proposed in [24] (N-TOMIC method), [31] and the PROMETHEE method as it has been modified in [19]. Furthermore, P. Perny [28] extended the existing framework of the sorting methods based on the outranking relations approach in the case in which the groups are not ordered. More specifically, he proposed the construction of a fuzzy outranking relation in order to estimate the membership of each alternative for each group, and suggested two assignment procedures:

a) filtering by strict preference (the assignment rule consists of testing whether an alternative is preferred or not to a reference profile reflecting the lower limit of a group), and
b) filtering by indifference (the assignment rule consists of testing whether an alternative is indifferent or not to a reference profile representing a prototype of a group).

Overall the main characteristics of sorting methods based on the outranking relations approach of MCDA include their application to both sorting (ordered classes) as well as discrimination (non ordered classes) problems, and the significant amount of information that they require by the decision maker.

Preference Disaggregation Approaches

The *preference disaggregation* approach refers to the analysis (disaggregation) of the global preferences of the decision maker to deduce the relative importance of the evaluation criteria, using ordinal regression techniques based mainly on linear programming formulations.

In contrast to the outranking relations approach the global preference model of the decision maker is not constructed through a direct interrogation procedure between the decision analyst and the decision maker. Instead, decision instances (e. g. past decisions) are used in order to analyze the decision policy of the decision maker, to specify his/her preferences and construct the corresponding global preference model as consistently as possible.

A well known preference disaggregation method is the UTA method (UTilités Additives) proposed in [17]. Given a predefined ranking of a reference set of alternatives, the aim of the UTA method is to construct a set of additive utility functions which are as consistent as possible with the pre-ordering of the alternatives (and consequently with the decision maker's preferences). The form of the additive utility function is the following:

$$U(\overline{g}) = \sum_j u_j(g_j),$$

where $U(\overline{g})$ denotes the global utility of an alternative described over a vector of criteria $\overline{g}$, while $u_j(g_j)$ is the partial or marginal utility of an alternative on criterion g_j.

Except for the study of ranking problems, the methodological framework of the preference disaggregation approach using the UTA method is also applicable in sorting problems. The UTADIS method (UTilités Additives DIScriminantes) [6,16,17,42] is a representative example. In the UTADIS method, the sorting of the alternatives is accomplished by comparing the global utility (scores) of each alternative a, denoted as $U(a)$, with some thresholds $(u_1, \dots, u_{q-1})$ which distinguish the classes $C_1, \dots, C_q$ (the classes are ordered, so that C_1 is the class of the best alternatives and C_q is the class of the worst alternatives).

$$\begin{aligned}
&U(a) \geq u_1 \Rightarrow a \in C_1\\
&u_2 \leq U(a) < u_1 \Rightarrow a \in C_2\\
&\cdots\\
&u_k \leq U(a) < u_{k-1} \Rightarrow a \in C_k\\
&\cdots\\
&U(a) < u_{q-1} \Rightarrow a \in C_q .
\end{aligned}$$

The objective of the UTADIS method is to estimate an additive utility function and the utility thresholds in order to minimize the classification error. The classification error is measured through two error functions denoted as $\sigma^+(a)$ and $\sigma^-(a)$, representing the deviations of a misclassified alternative from the utility threshold. The estimation of both the additive utility model and the utility thresholds is achieved through linear programming techniques [6,42].

See [7] and [41] for three variants of the UTADIS method to improve the classification accuracy of the obtained additive utility models as well as their predicting ability. The first variant (UTADIS I) except for the classification errors also incorporates the distances of the correctly classified alternatives from the utility thresholds which have to be maximized. The second variant (UTADIS II) is based on a mixed integer programming formulation minimizing the number of misclassifications instead of their magnitude, while the third variant (UTADIS III) combines UTADIS I and II, and its aim is to minimize the number of misclassifications and maximize the distances of the correctly classified alternatives from the utility thresholds.

Overall the main characteristics of the application of the preference disaggregation approach in the study of sorting problems, can be summarized in the following three aspects.

1) The information that is required is minimal, since, similarly to the goal programming approaches, only a predefined classification of a reference set of alternatives is required.
2) The preference disaggregation approach is focused only on decision problems where the classes are ordered, since it is assumed that there is a strict preference relation between the classes.
3) The classification/sorting models which are developed have a nonlinear form, since the marginal utilities of the evaluation criteria are piecewise linear and consequently the global utility model is also nonlinear, in contrast to the linear discriminant models used in the goal programming approaches.

A Multigroup Hierarchical Discrimination Method

In this section a new method is presented for the study of discrimination problems with two or more ordered groups (multigroup discrimination). The proposed method is called M.H.DIS (Multigroup Hierarchical DIScrimination) and differs from most of the aforementioned MCDA approaches in two major aspects.

1) It employs a hierarchical discrimination approach: the method does not aim on the development of an overall global preference model (discriminant function) which will characterize all the observations (alternatives or objects). Instead the method is trying to distinguish the groups progressively, starting by discriminating the first group (best alternatives) from all the others, and then proceeding to the discrimination between the objects which belong to the other groups.
2) It accommodates three different discrimination criteria in a very flexible and efficient way. The most common discrimination criterion in the previous approaches is the minimization of the classification error which is measured as the deviations of the scores of the misclassified alternatives from some cut-off points. However, such an objective does not necessarily yield the optimal classification rule. For instance, consider that in a discrimination problem, three alternatives are misclassified with the following deviations from the cut-off point: [0.25, 0.25, 0.25], with the overall objective of minimizing the total classification error being 0.75. It is obvious, that this classification result is not optimal, since a classification result [0, 0, 0.75] yields the same value for the overall classification error (0.75), but there is only one misclassified alternative instead of three. Several mixed integer programming formulations have been proposed to confront this issue, but their application in real world problems is prohibited by the significant amount of time required to solve such problems. M.H.DIS employs an efficient *mixed integer programming* (MIP) formulation for minimizing the number of misclassifications, once the minimization of the classification error has been achieved. Furthermore, M.H.DIS also considers a third criterion in order to achieve the higher possible discrimination. These three discrimination criteria have been used in previous studies separately, or in hybrid models [12,13], but they have never been used through a sequential procedure. Instead, in M.H.DIS initially the classification error is minimized. Then considering only the misclassified alternatives M.H.DIS tries to 're-arrange' their classification error in order to minimize the number of misclassifications, and finally the maximum discrimination between the alternatives is attempted.

Model Formulation

Let $A = \{a_1, \dots, a_n\}$ be a set of n alternatives which should be classified into q ordered classes $C_1, \dots, C_q$.(C_1 is preferred to C_2, C_2 is preferred to C_3, etc.) Each alternative is described (evaluated) along a set $G = \{g_1, \dots, g_m\}$ of m evaluation criteria. The evaluation of each alternative a on criterion g_i is denoted as $g_i(a)$. According to the set A of alternatives, p_i different values for each criterion g_i can be distinguished. These p_i values are rank-ordered from the smallest value g_i^1 to the largest value $g_i^{p_i}$. Furthermore, among the set of criteria it is possible to distinguish two subsets: a subset G_1 consisting of m_1 criteria for which higher values indicate higher preference, and a second subset G_2 consisting of m_2 criteria for which the decision maker's preference is a decreasing function of the criterion's scale. For instance, in an investment decision problem G_1 may include criteria related to the return of an investment project (projects with higher return are preferred), while G_2 may include criteria related to the risk of the investment (projects with lower risk are preferred).

The Hierarchical Discrimination Process

The method proceeds progressively in the classification of the alternatives into the predefined classes, starting from class C_1 (best alternatives). Initially, the aim is to identify which alternatives belong in class C_1. The alternatives which are found to belong in class C_1 (either correctly or incorrectly) are excluded from further consideration. In a second stage the objective is to identify which alternatives belong in class C_2. The alternatives which are found to belong in this class (either correctly or incorrectly) are excluded from further consideration, and the same procedure continues until all alternatives have been classified in the predefined classes.

Throughout this hierarchical classification procedure, it is assumed that the decision maker's preferences are monotone functions (increasing or decreasing) on the criteria's scale. This assumption implies that in the case of a criterion $g_i \in G_1$, as the evaluation of an alternative on this criterion increases, then the decision of classifying this alternative into a higher (better) class is more favorable to a decision of classifying the alternative into a lower (worst) class. For instance, in the credit granting problem as the profitability of a firm increases, the credit analyst will be more favorable in classifying the firm as a healthy firm, rather than classifying it as a risky one. A similar implication is also made for each criterion $g_i \in G_2$.

This preference relation between the several possible decisions of classifying a specific alternative a into one of the predefined classes, imposes the following general classification rule:

> The decision concerning the classification of an alternative a into one of the predefined classes should be made in such a way that the utility (value) of such a decision for the decision maker is maximized.

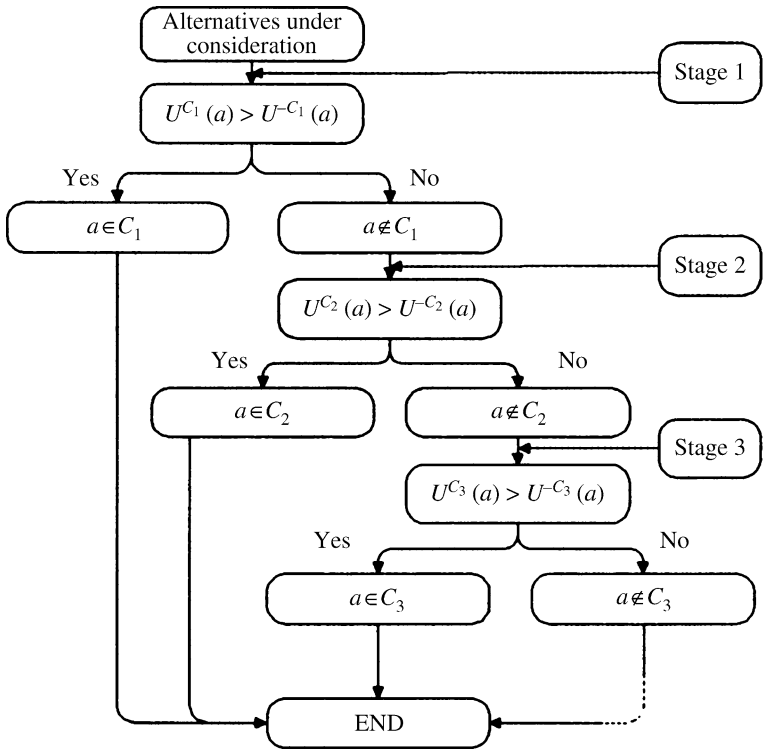
The utility of a decision concerning the classification of an alternative a into group C_j can be expressed in the form of additive utility function:

$$U^{C_j}(a) = \sum_{i=1}^{m} u_i^{C_j}[g_i(a)] \in [0,1],$$

where $u_i^{C_j}[g_i(a)]$ denotes the marginal (partial) utility of the decision concerning the classification of an alternative a into group C_j according to criterion g_i. If $g_i \in G_1$, then $u_i^{C_j}(g_i)$ will be an increasing function on the criterion's scale. On the contrary, the marginal utility of a criterion $g_i \in G_2$ regarding the classification of an alternative into a lower (worse) class C_k $(k > j)$ will be a decreasing function on the criterion's scale. For instance, consider once again the credit granting problem: since healthy firms are generally characterized by high profitability, the marginal utility for a profitability criterion for the group of healthy firms will be an increasing function, indicating that as profitability increases the preference of decision concerning the classification of a firm in the group of healthy firms in also increasing. On the other hand, for the group of risky firms the marginal utility will be a decreasing function of the criterion's (profitability) values, indicating that as profitability increases the preference of the decision concerning the classification of a firm in the group of risky firms is decreasing.

Consequently, at each stage of the hierarchical classification procedure that was described above, two utility functions are constructed. The first one corresponds to the utility of a decision concerning the classification of an alternative a into class C_k (denoted as $U^{C_k}(a)$), while the second one corresponds to the utility of a decision concerning the nonclassification of an alternative a into class C_k (denoted as $U^{-C_k}(a)$). Based on these two utility functions the aforementioned general classification rule can be expressed as follows:

$$\begin{cases} \text{if } U^{C_k}(a) > U^{-C_k}(a), & \text{then } a \in C_k, \\ \text{if } U^{C_k}(a) < U^{-C_k}(a), & \text{then } a \notin C_k. \end{cases} \tag{1}$$



Multicriteria Sorting Methods, Figure 1
The hierarchical classification procedure

Following this rule, the overall hierarchical discrimination procedure is presented in Fig. 1.

Estimation of Utility Functions

According to the hierarchical discrimination procedure which was described above, to achieve the classification of the alternatives in q classes, the number of utility functions which must be estimated is $2(q-1)$. The estimation of these utility functions in M.H.DIS is accomplished through linear programming techniques. More specifically, at each stage of the hierarchical discrimination procedure, two linear programs and one mixed integer program are solved to estimate 'optimally' the two utility functions.

LP1: Minimizing the Overall Classification Error

According to the classification rule (1), to achieve the correct classification of an alternative $a \in C_k$ at stage k (cf. Fig. 1), the estimated utility functions should satisfy the following constraint:

$$U^{C_k}(a) > U^{-C_k}(a).$$

Since, in linear programming it is not possible to use strict inequality constraints, a small positive real number s may be used as follows:

$$U^{C_k}(a) - U^{-C_k}(a) \geq s.$$

If for an alternative $a \in C_k$ the classification rule at stage k yields $U^{C_k}(a) < U^{-C_k}(a)$, then this alternative is misclassified, since it should be classified in one of the lower classes (the specific classification of the alternative will be determined in the next stages of the hierarchical discrimination process). The classification error in this case is:

$$e(a) = U^{-C_k}(a) - U^{C_k}(a) + s.$$

Similarly, to achieve the correct classification of an alternative $b \notin C_k$ at stage k, the estimated utility functions should satisfy the following constraint:

$$U^{-C_k}(b) - U^{C_k}(b) \geq s.$$

If this constraint is not satisfied for an alternative $b \notin C_k$ at stage k, then this fact implies that this alternative should be classified in class C_k and the classification error in this case is $e(b) = U^{C_k}(b) - U^{-C_k}(b) + s$.

Moreover, to achieve the monotonicity of the marginal utilities, the following constraints are imposed:

$$\text{if } g_i \in G_1 \begin{cases} u_i^{C_k}(g_i^1) = 0 \\ u_i^{-C_k}(g_i^{p_i}) = 0 \\ u_i^{C_k}(g_i^{j+1}) > u_i^{C_k}(g_i^j) \\ u_i^{-C_k}(g_i^{j+1}) < u_i^{-C_k}(g_i^j) \end{cases} \tag{2}$$

$$\text{if } g_i \in G_2 \begin{cases} u_i^{C_k}(g_i^{p_i}) = 0 \\ u_i^{-C_k}(g_i^1) = 0 \\ u_i^{C_k}(g_i^{j+1}) < u_i^{C_k}(g_i^j) \\ u_i^{-C_k}(g_i^{j+1}) > u_i^{-C_k}(g_i^j) \end{cases} \tag{3}$$

where g_i^j and g_i^{j+1} are two consecutive values of criterion $g_i (g_i^{j+1} > g_i^j$ for all $g_i \in G)$. These constraints can be simplified by setting:

$$\text{if } g_i \in G_1 \begin{cases} w_{ij,j+1}^{C_k} = u_i^{C_k}(g_i^{j+1}) - u_i^{C_k}(g_i^j) \\ w_{ij,j+1}^{-C_k} = u_i^{-C_k}(g_i^j) - u_i^{-C_k}(g_i^{j+1}) \end{cases} \tag{4}$$

$$\text{if } g_i \in G_2 \begin{cases} w_{ij,j+1}^{C_k} = u_i^{C_k}(g_i^j) - u_i^{C_k}(g_i^{j+1}) \\ w_{ij,j+1}^{-C_k} = u_i^{-C_k}(g_i^{j+1}) - u_i^{-C_k}(g_i^j) \end{cases} \tag{5}$$

The marginal utility of criterion g_i at point g_i^j can then be calculated through the following formulas:

$$\begin{aligned} u_i^{C_k}(g_i^j) &= \sum_{l=1}^{j-1} w_{il,l+1}^{C_k}, \\ u_i^{-C_k}(g_i^j) &= \sum_{l=j}^{p_i-1} w_{il,l+1}^{-C_k}. \end{aligned} \tag{6}$$

Using these transformations, constraints (2) and (3) can be rewritten as follows (a small positive number t is used to ensure the strict inequality):

$$w_{ij,j+1}^{C_k} \geq t, \quad w_{ij,j+1}^{-C_k} \geq t, \qquad \forall g_i.$$

Consequently, the initial linear program (LP1) to be solved can be formulated as follows:

$$\begin{cases} \min & F = \sum_{a \in A} e(a) \\ \text{s.t.} & U^{C_k}(a) - U^{-C_k}(a) + e(a) \geq s, \\ & \quad \forall a \in C_k, \\ & U^{-C_k}(b) - U^{C_k}(b) + e(b) \geq s, \\ & \quad \forall b \notin C_k, \\ & w_{ij,j+1}^{C_k} \geq t \\ & w_{ij,j+1}^{-C_k} \geq t \\ & \sum_i \sum_j w_{ij,j+1}^{C_k} = 1 \\ & \sum_i \sum_j w_{ij,j+1}^{-C_k} = 1 \\ & e(a), s, t \geq 0. \end{cases}$$

LP2: Minimizing the Number of Misclassifications

If after the solution of (LP1), there exist some alternatives $a \in A$ for which $e(a) > 0$, then obviously these alternatives are misclassified. However, as it has been already illustrated during the discussion of the main characteristics of M.H.DIS, it may be possible to achieve a 're-arrangement' of the classification errors which may lead to the reduction of the number of misclassifications.

In M.H.DIS this is achieved through a mixed integer programming (MIP) formulation. However, since MIP formulations are difficult to solve, especially in cases where the number or integer or binary variables is large, the MIP formulation used in M.H.DIS considers only the misclassifications occurred by solving (LP1), while retaining all the correct classifications. Let C be the set of alternatives which have been correctly classified after solving (LP1), and M be the set of misclassified alternatives for which $e(a) > 0$. The MIP formulation used in

M.H.DIS is the following (LP2):

$$\begin{cases}
\min & F = \sum_{a \in A} I(a) \\
\text{s.t.} & U^{C_k}(a) - U^{-C_k}(a) \geq s, \\
& \quad \forall a \in C_k \cap C, \\
& U^{-C_k}(b) - U^{C_k}(b) \geq s, \\
& \quad \forall b \notin C_k, b \in C, \\
& U^{C_k}(a) - U^{-C_k}(a) + I(a) \geq s, \\
& \quad \forall a \in C_k \cap M, \\
& U^{-C_k}(b) - U^{C_k}(b) + I(a) \geq s, \\
& \quad \forall b \notin C_k, b \in M, \\
& w_{ij,j+1}^{C_k} \geq t \\
& w_{ij,j+1}^{-C_k} \geq t \\
& \sum_i \sum_j w_{ij,j+1}^{C_k} = 1 \\
& \sum_i \sum_j w_{ij,j+1}^{-C_k} = 1 \\
& s, t, I(a) \text{ integer.}
\end{cases}$$

The first set of constraints is used to ensure that all the correct classifications achieved by solving (LP1) are retained. The second set of constraints is used only for the alternatives which were misclassified by (LP1). Their meaning is similar to the constraints in LP1, with the only difference being the transformation of the continuous variables $e(a)$ of LP1 (classification errors) into integer variables $I(a)$ which indicate whether an alternative is misclassified or not. The meaning of the final two constraints has already been illustrated in the discussion of the LP1 formulation. The objective of LP2 is to minimize the number of misclassifications occurred through the solution of LP1.

LP3: Maximizing the Minimum Distance

Solving LP1 and LP2 the 'optimal' classification of the alternatives has been achieved, where the term 'optimal' refers to the minimization of the number of misclassified alternatives. However, the correct classification of some alternatives may have been 'marginal', that is although they are correctly classified, their global utilities according to the two utility functions developed may have been very close. The objective of LP3 is to maximize the minimum difference between the global utilities of the correctly classified alternatives achieved according to the two utility functions.

Similarly to LP2, let C be the set of alternatives which have been correctly classified after solving LP1 and LP2, and M be the set of misclassified alternatives. LP3 can be formulated as follows:

$$\begin{cases}
\max & d \\
\text{s.t.} & U^{C_k}(a) - U^{-C_k}(a) - d \geq s, \\
& \quad \forall a \in C_k \cap C, \\
& U^{-C_k}(b) - U^{C_k}(b) - d \geq s, \\
& \quad \forall b \notin C_k, b \in C, \\
& U^{C_k}(a) - U^{-C_k}(a) \geq s, \\
& \quad \forall a \in C_k \cap M, \\
& U^{-C_k}(b) - U^{C_k}(b) \geq s, \\
& \quad \forall b \notin C_k, b \in M, \\
& w_{ij,j+1}^{C_k} \geq t \\
& w_{ij,j+1}^{-C_k} \geq t \\
& \sum_i \sum_j w_{ij,j+1}^{C_k} = 1 \\
& \sum_i \sum_j w_{ij,j+1}^{-C_k} = 1 \\
& d, s, t \geq 0.
\end{cases}$$

The first set of constraints involves only the correctly classified alternatives. In these constraints d represents the minimum absolute difference between the global utilities of each alternative in the two utility functions. The second set of constraints involves the misclassified alternatives and it is used to ensure that they will be retained as misclassified.

An Illustrative Example

To illustrate the application of the method, consider a simple example consisting of six alternatives evaluated along three evaluation criteria [25] for which higher values are preferred. The alternatives must be classified in three ordered classes. Table 1, illustrates the evaluation of the alternatives on the criteria as well as the predefined classification.

Distinguishing Between C_1 and C_2-C_3

In the first stage of the hierarchical discrimination procedure, the aim is to distinguish the alternatives belonging in class C_1 from the alternatives belonging in

Multicriteria Sorting Methods, Table 1
Data of the illustrative example (Source: [25])

	g_1	g_2	g_3	Class
a_1	70	64.75	46.25	C_1
a_2	61	62	60	C_1
a_3	40	50	37	C_2
a_4	66	40	23.125	C_2
a_5	20	20	20	C_3
a_6	15	15	30	C_3

classes C_2 and C_3. To achieve this classification two utility functions are developed, denoted as $U^{C_1}(a)$ and $U^{-C_1}(a)$.

The utility of the decision of classifying the alternative a_1 in class C_1 can be expressed as follows:

$$U^{C_1}(a_1) = u_1^{C_1}(70) + u_2^{C_1}(64.75) + u_3^{C_1}(46.25). \quad (7)$$

Since for all criteria higher values are preferred, it is possible to define the following rank-order on each criterion's scale ($p_1 = p_2 = p_3 = 6$).

g_1) $g_1^1 = 15 < 20 < 40 < 61 < 66 < 70 = g_1^{p_1}$;
g_2) $g_2^1 = 15 < 20 < 40 < 50 < 62 < 64.75 = g_2^{p_2}$;
g_3) $g_3^1 = 20 < 23.125 < 30 < 37 < 46.25 < 60 = g_3^{p_3}$.

According to relation (4), the following transformations are then applied (criterion g_1):

$$\begin{aligned} w_{11,2}^{C_1} &= u_1^{C_1}(20) - u_1^{C_1}(15),\\ w_{12,3}^{C_1} &= u_1^{C_1}(40) - u_1^{C_1}(20),\\ w_{13,4}^{C_1} &= u_1^{C_1}(61) - u_1^{C_1}(40),\\ w_{14,5}^{C_1} &= u_1^{C_1}(66) - u_1^{C_1}(61),\\ w_{15,6}^{C_1} &= u_1^{C_1}(70) - u_1^{C_1}(66). \end{aligned}$$

The same transformations are also applied to criteria g_2 and g_3. Then, according to (6), relation (7) can be re-written in the following way:

$$\begin{aligned} U^{C_1}(a) = (w_{11,2}^{C_1} + w_{12,3}^{C_1} + w_{13,4}^{C_1} + w_{14,5}^{C_1} + w_{15,6}^{C_1})\\ + (w_{21,2}^{C_1} + w_{22,3}^{C_1} + w_{23,4}^{C_1} + w_{24,5}^{C_1} + w_{25,6}^{C_1})\\ + (w_{31,2}^{C_1} + w_{32,3}^{C_1} + w_{33,4}^{C_1} + w_{34,5}^{C_1}). \end{aligned}$$

On the other hand, if a_1 is classified in class C_2 then the utility of the decision maker will be:

$$\begin{gathered} U^{-C_1}(a_1) = u_1^{-C_1}(70) + u_2^{-C_1}(64.75) + u_3^{-C_1}(46.25)\\ \Updownarrow\\ U^{-C_1}(a_1) = w_{35,6}^{-C_1}. \end{gathered}$$

Following the same methodology, the utilities concerning the classification of the rest of the alternatives are also formulated.

- Alternative a_2:

$$\begin{aligned} &U^{-C_1}(a_2) = u_1^{C_1}(61) + u_2^{C_1}(62) + u_3^{C_1}(60)\\ &\Updownarrow\\ &U^{C_1}(a_2) = (w_{11,2}^{C_1} + w_{12,3}^{C_1} + w_{13,4}^{C_1})\\ &\quad + (w_{21,2}^{C_1} + w_{22,3}^{C_1} + w_{23,4}^{C_1} + w_{24,5}^{C_1})\\ &\quad + (w_{31,2}^{C_1} + w_{32,3}^{C_1} + w_{33,4}^{C_1} + w_{34,5}^{C_1} + w_{35,6}^{C_1}),\\ &U^{-C_1}(a_2) = u_1^{-C_1}(61) + u_2^{-C_1}(62) + u_3^{-C_1}(60)\\ &\Updownarrow\\ &U^{-C_1}(a_2) = (w_{14,5}^{-C_1} + w_{15,6}^{-C_1}) + (w_{25,6}^{-C_1}). \end{aligned}$$

- Alternative a_3:

$$\begin{aligned} &U^{-C_1}(a_3) = u_1^{C_1}(40) + u_2^{C_1}(50) + u_3^{C_1}(37)\\ &\Updownarrow\\ &U^{C_1}(a_3) = (w_{11,2}^{C_1} + w_{12,3}^{C_1})\\ &\quad + (w_{21,2}^{C_1} + w_{22,3}^{C_1} + w_{23,4}^{C_1})\\ &\quad + (w_{31,2}^{C_1} + w_{32,3}^{C_1} + w_{33,4}^{C_1}),\\ &U^{-C_1}(a_3) = u_1^{-C_1}(40) + u_2^{-C_1}(50) + u_3^{-C_1}(37)\\ &\Updownarrow\\ &U^{-C_1}(a_3) = (w_{13,4}^{-C_1} + w_{14,5}^{-C_1} + w_{15,6}^{-C_1})\\ &\quad + (w_{24,5}^{-C_1} + w_{25,6}^{-C_1}) + (w_{34,5}^{-C_1} + w_{35,6}^{-C_1}). \end{aligned}$$

- Alternative a_4:

$$\begin{aligned} &U^{-C_1}(a_4) = u_1^{C_1}(66) + u_2^{C_1}(40) + u_3^{C_1}(23.125)\\ &\Updownarrow\\ &U^{C_1}(a_4) = (w_{11,2}^{C_1} + w_{12,3}^{C_1} + w_{13,4}^{C_1} + w_{14,5}^{C_1}) \end{aligned}$$

$$+ (w^{C_1}_{21,2} + w^{C_1}_{22,3}) + (w^{C_1}_{31,2}),$$

$$U^{-C_1}(a_4) = u_1^{-C_1}(66) + u_2^{-C_1}(40) + u_3^{-C_1}(23.125)$$

$$\Updownarrow$$

$$U^{-C_1}(a_4) = (w^{-C_1}_{15,6}) + (w^{-C_1}_{23,4} + w^{-C_1}_{24,5} + w^{-C_1}_{25,6}) + (w^{-C_1}_{32,3} + w^{-C_1}_{33,4} + w^{-C_1}_{34,5} + w^{-C_1}_{35,6}).$$

- Alternative a_5:

$$U^{-C_1}(a_5) = u_1^{C_1}(20) + u_2^{C_1}(20) + u_3^{C_1}(20)$$

$$\Updownarrow$$

$$U^{C_1}(a_5) = (w^{C_1}_{11,2}) + (w^{C_1}_{21,2}),$$

$$U^{-C_1}(a_5) = u_1^{-C_1}(20) + u_2^{-C_1}(20) + u_3^{-C_1}(20)$$

$$\Updownarrow$$

$$U^{-C_1}(a_5) = (w^{-C_1}_{12,3} + w^{-C_1}_{13,4} + w^{-C_1}_{14,5} + w^{-C_1}_{15,6}) + (w^{-C_1}_{22,3} + w^{-C_1}_{23,4} + w^{-C_1}_{24,5} + w^{-C_1}_{25,6}) + (w^{-C_1}_{31,2} + w^{-C_1}_{32,3} + w^{-C_1}_{33,4} + w^{-C_1}_{34,5} + w^{-C_1}_{35,6}).$$

- Alternative a_6:

$$U^{-C_1}(a_6) = u_1^{C_1}(15) + u_2^{C_1}(15) + u_3^{C_1}(30)$$

$$\Updownarrow$$

$$U^{C_1}(a_6) = (w^{C_1}_{31,2} + w^{C_1}_{32,3}),$$

$$U^{-C_1}(a_6) = u_1^{-C_1}(15) + u_2^{-C_1}(15) + u_3^{-C_1}(15)$$

$$\Updownarrow$$

$$U^{-C_1}(a_6) = (w^{-C_1}_{11,2} + w^{-C_1}_{12,3} + w^{-C_1}_{13,4} + w^{-C_1}_{14,5} + w^{-C_1}_{15,6}) + (w^{-C_1}_{21,2} + w^{-C_1}_{22,3} + w^{-C_1}_{23,4} + w^{-C_1}_{24,5} + w^{-C_1}_{25,6}) + (w^{-C_1}_{33,4} + w^{-C_1}_{34,5} + w^{-C_1}_{35,6}).$$

According to these expressions of the global utility of the decision to classify an alternative into class C_1 or into one of the classes C_2 and C_3, the LP1 formulation is used to minimize the classification error ($s = 0.001$, $t = 0.0001$).

$$\left\{\begin{array}{ll}
\min & F = e(a_1) + e(a_2) + e(a_3) + e(a_4) \\
 & \quad + e(a_5) + e(a_6) \\
\text{s.t.} & U^{C_1}(a_1) - U^{-C_1}(a_1) + e(a_1) \geq 0.001 \\
 & U^{C_1}(a_2) - U^{-C_1}(a_2) + e(a_2) \geq 0.001 \\
 & U^{-C_1}(a_3) - U^{C_1}(a_3) + e(a_3) \geq 0.001 \\
 & U^{-C_1}(a_4) - U^{C_1}(a_4) + e(a_4) \geq 0.001 \\
 & U^{-C_1}(a_5) - U^{C_1}(a_5) + e(a_5) \geq 0.001 \\
 & U^{-C_1}(a_6) - U^{C_1}(a_6) + e(a_6) \geq 0.001 \\
 & w^{C_1}_{ij,j+1} \geq 0.0001, \quad w^{-C_1}_{ij,j+1} \geq 0.0001, \\
 & \sum_{i=1}^{3}\sum_{j=1}^{5} w^{C_1}_{ij,j+1} = 1, \\
 & \sum_{i=1}^{3}\sum_{j=1}^{5} w^{-C_1}_{ij,j+1} = 1, \\
 & \forall i = 1,2,3, \quad \forall j = 1,\ldots,6, \\
 & e(a_1), e(a_2), e(a_3) \geq 0, \\
 & e(a_4), e(a_5), e(a_6) \geq 0.
\end{array}\right.$$

The obtained solution is presented in Table 2. According to this solution, the marginal utilities are calculated.

- Criterion g_1:
  - $u_1^{C_1}(15) = 0$,
  - $u_1^{-C_1}(15) = w^{-C_1}_{11,2} + w^{-C_1}_{12,3} + w^{-C_1}_{13,4} + w^{-C_1}_{14,5} + w^{-C_1}_{15,6} = 0.25937$,

Multicriteria Sorting Methods, Table 2
Results obtained through the solution of LP1

$w^{C_1}_{11,2}$	0.00010	$w^{-C_1}_{11,2}$	0.03708
$w^{C_1}_{12,3}$	0.00010	$w^{-C_1}_{12,3}$	0.03708
$w^{C_1}_{13,4}$	0.09872	$w^{-C_1}_{13,4}$	0.07406
$w^{C_1}_{14,5}$	0.00010	$w^{-C_1}_{14,5}$	0.03708
$w^{C_1}_{15,6}$	0.09872	$w^{-C_1}_{15,6}$	0.07406
$w^{C_1}_{21,2}$	0.00010	$w^{-C_1}_{21,2}$	0.03708
$w^{C_1}_{22,3}$	0.00010	$w^{-C_1}_{22,3}$	0.03708
$w^{C_1}_{23,4}$	0.09872	$w^{-C_1}_{23,4}$	0.07406
$w^{C_1}_{24,5}$	0.13570	$w^{-C_1}_{24,5}$	0.11104
$w^{C_1}_{25,6}$	0.09872	$w^{-C_1}_{25,6}$	0.07406
$w^{C_1}_{31,2}$	0.00010	$w^{-C_1}_{31,2}$	0.03708
$w^{C_1}_{32,3}$	0.09872	$w^{-C_1}_{32,3}$	0.07406
$w^{C_1}_{33,4}$	0.09872	$w^{-C_1}_{33,4}$	0.07406
$w^{C_1}_{34,5}$	0.13570	$w^{-C_1}_{34,5}$	0.11104
$w^{C_1}_{35,6}$	0.13570	$w^{-C_1}_{35,6}$	0.11104

- $u_1^{C_1}(20) = w_{11,2}^{C_1} = 0.0001$,
- $u_1^{-C_1}(20) = w_{12,3}^{-C_1} + w_{13,4}^{-C_1} + w_{14,5}^{-C_1} + w_{15,6}^{-C_1} = 0.22229$,
- $u_1^{C_1}(40) = w_{11,2}^{C_1} + w_{12,3}^{C_1} = 0.0002$,
- $u_1^{-C_1}(40) = w_{13,4}^{-C_1} + w_{14,5}^{-C_1} + w_{15,6}^{-C_1} = 0.18521$,
- $u_1^{C_1}(61) = w_{11,2}^{C_1} + w_{12,3}^{C_1} + w_{13,4}^{C_1} = 0.09892$,
- $u_1^{-C_1}(61) = w_{14,5}^{-C_1} + w_{15,6}^{-C_1} = 0.11114$,
- $u_1^{C_1}(66) = w_{11,2}^{C_1} + w_{12,3}^{C_1} + w_{13,4}^{C_1} + w_{14,5}^{C_1} = 0.09902$,
- $u_1^{-C_1}(66) = w_{15,6}^{-C_1} = 0.07406$,
- $u_1^{C_1}(70) = w_{11,2}^{C_1} + w_{12,3}^{C_1} + w_{13,4}^{C_1} + w_{14,5}^{C_1} + w_{15,6}^{C_1} = 0.19773$,
- $u_1^{-C_1}(70) = 0$;

- Criterion g_2:
  - $u_2^{C_1}(15) = 0$,
  - $u_2^{-C_1}(15) = w_{21,2}^{-C_1} + w_{22,3}^{-C_1} + w_{23,4}^{-C_1} + w_{24,5}^{-C_1} + w_{25,6}^{-C_1} = 0.33333$,
  - $u_2^{C_1}(20) = w_{21,2}^{C_1} = 0.0001$,
  - $u_2^{-C_1}(20) = w_{22,3}^{-C_1} + w_{23,4}^{-C_1} + w_{24,5}^{-C_1} + w_{25,6}^{-C_1} = 0.29625$,
  - $u_2^{C_1}(40) = w_{21,2}^{C_1} + w_{22,3}^{C_1} = 0.0002$,
  - $u_2^{-C_1}(40) = w_{23,4}^{-C_1} + w_{24,5}^{-C_1} + w_{25,6}^{-C_1} = 0.25917$,
  - $u_2^{C_1}(50) = w_{21,2}^{C_1} + w_{22,3}^{C_1} + w_{23,4}^{C_1} = 0.09892$,
  - $u_2^{-C_1}(50) = w_{24,5}^{-C_1} + w_{25,6}^{-C_1} = 0.18511$,
  - $u_2^{C_1}(62) = w_{21,2}^{C_1} + w_{22,3}^{C_1} + w_{23,4}^{C_1} + w_{24,5}^{C_1} = 0.23462$,
  - $u_2^{-C_1}(62) = w_{25,6}^{-C_1} = 0.07406$,
  - $u_2^{C_1}(64.75) = w_{21,2}^{C_1} + w_{22,3}^{C_1} + w_{23,4}^{C_1} + w_{24,5}^{C_1} + w_{25,6}^{C_1} = 0.33333$,
  - $u_2^{-C_1}(64.75) = 0$;
- Criterion g_3:
  - $u_3^{C_1}(20) = 0$,
  - $u_3^{-C_1}(20) = w_{31,2}^{-C_1} + w_{32,3}^{-C_1} + w_{33,4}^{-C_1} + w_{34,5}^{-C_1} + w_{35,6}^{-C_1} = 0.40730$,
  - $u_3^{C_1}(23.125) = w_{31,2}^{C_1} = 0.0001$,
  - $u_3^{-C_1}(23.125) = w_{32,3}^{-C_1} + w_{33,4}^{-C_1} + w_{34,5}^{-C_1} + w_{35,6}^{-C_1} = 0.37021$,
  - $u_3^{C_1}(30) = w_{31,2}^{C_1} + w_{32,3}^{C_1} = 0.09882$,
  - $u_3^{-C_1}(30) = w_{33,4}^{-C_1} + w_{34,5}^{-C_1} + w_{35,6}^{-C_1} = 0.29615$,
  - $u_3^{C_1}(37) = w_{31,2}^{C_1} + w_{32,3}^{C_1} + w_{33,4}^{C_1} = 0.19753$,
  - $u_3^{-C_1}(37) = w_{34,5}^{-C_1} + w_{35,6}^{-C_1} = 0.22209$,
  - $u_3^{C_1}(46.25) = w_{31,2}^{C_1} + w_{32,3}^{C_1} + w_{33,4}^{C_1} + w_{34,5}^{C_1} = 0.33323$,
  - $u_3^{-C_1}(46.25) = w_{35,6}^{-C_1} = 0.11104$,
  - $u_3^{C_1}(60) = w_{31,2}^{C_1} + w_{32,3}^{C_1} + w_{33,4}^{C_1} + w_{34,5}^{C_1} + w_{35,6}^{C_1} = 0.46893$,
  - $u_3^{-C_1}(60) = 0$;

Multicriteria Sorting Methods, Table 3
Global utilities obtained through the solution of LP1 (stage 1)

	$U^{C_1}(a)$	$U^{-C_1}(a)$
a_1	0.8643	0.1110
a_2	0.8025	0.1852
a_3	0.2967	0.5924
a_4	0.0993	0.7034
a_5	0.0002	0.9258
a_6	0.0988	0.8889

According to these marginal utilities, the global utilities are calculated based on the expressions that have already been presented. Table 3, illustrates the obtained global utilities according to the two utility functions that were developed.

It is clear that a_1 and a_2 are classified in class C_1, since the global utility of a decision concerning the classification of these two alternatives in class C_1 is greater than the utility concerning their classification in classes C_2 or C_3. Similarly, alternatives a_3, a_4, a_5 and a_6 are not classified in class C_1, but instead they belong in one of the classes C_2 or C_3 (their specific classification will be determined in the next stage of the hierarchical discrimination process).

Since the correct discrimination between the alternatives belonging in class C_1 and the alternative not belonging in this class has been achieved through LP1, it is not necessary to proceed in LP2 (minimization of the number of misclassifications). Hence, the procedure proceeds in the formulation and solution of LP3 in order to achieve the higher possible discrimination:

$$\begin{cases} \max & d \\ \text{s.t.} & U^{C_1}(a_1) - U^{-C_1}(a_1) - d \geq 0.001 \\ & U^{C_1}(a_2) - U^{-C_1}(a_2) - d \geq 0.001 \\ & U^{-C_1}(a_3) - U^{C_1}(a_3) - d \geq 0.001 \\ & U^{-C_1}(a_4) - U^{C_1}(a_4) - d \geq 0.001 \\ & U^{-C_1}(a_5) - U^{C_1}(a_5) - d \geq 0.001 \\ & U^{-C_1}(a_6) - U^{C_1}(a_6) - d \geq 0.001 \\ & w_{ij,j+1}^{C_1} \geq 0.0001, \quad w_{ij,j+1}^{-C_1} \geq 0.0001 \\ & \sum_{i=1}^{3}\sum_{j=1}^{5} w_{ij,j+1}^{C_1} = 1, \quad \sum_{i}^{3}\sum_{j=1}^{5} w_{ij,j+1}^{-C_1} = 1, \\ & \forall i = 1,2,3, \quad \forall j = 1,\ldots,6, \quad d \geq 0. \end{cases}$$

According to the obtained solution and following the same procedure for calculating the marginal utili-

Multicriteria Sorting Methods, Table 4
Global utilities obtained through the solution of LP3 (stage 1)

	$U^{C_1}(a)$	$U^{-C_1}(a)$
a_1	0.9985	0.0001
a_2	0.9987	0.0003
a_3	0.0008	0.9992
a_4	0.0009	0.9993
a_5	0.0002	0.9998
a_6	0.0002	0.9998

ties, the global utilities of Table 4 are obtained. Obviously, this new solution provides a better discrimination of the alternatives, compared to the initial solution obtained by LP1.

Distinguishing Between C_2 and C_3

After the solution of LP3, the first stage of the hierarchical discrimination process is completed, with the correct classification of a_1 and a_2 in class C_1. Consequently, these two alternatives are excluded from further consideration (second stage). In the second stage, the aim is to determine the specific classification of the alternatives a_3, a_4, a_5 and a_6. The following rank-order is defined on the scale of the three evaluation criteria ($p_1 = p_2 = p_3 = 4$).

g_1) $g_1^1 = 15 < 20 < 40 < 66 = g_1^{p_1}$;
g_2) $g_2^1 = 15 < 20 < 40 < 50 = g_2^{p_2}$;
g_3) $g_3^1 = 20 < 23.125 < 30 < 37 = g_2^{p_2}$.

Then, following the procedure illustrated in the previous stage, the variables $w_{ij,j+1}^{C_1}$ and $w_{ij,j+1}^{-C_1}$ are formulated, and the new form of the LP1 problem is the following ($s = 0.001$, $t = 0.0001$):

$$\left\{\begin{array}{ll}
\min & F = e(a_3) + e(a_4) + e(a_5) + e(a_6) \\
\text{s.t.} & U^{C_2}(a_3) - U^{-C_2}(a_3) + e(a_3) \geq 0.001 \\
& U^{C_2}(a_4) - U^{-C_2}(a_4) + e(a_4) \geq 0.001 \\
& U^{-C_2}(a_5) - U^{C_2}(a_5) + e(a_5) \geq 0.001 \\
& U^{-C_2}(a_6) - U^{C_2}(a_6) + e(a_6) \geq 0.001 \\
& w_{ij,j+1}^{C_2} \geq 0.0001, \quad w_{ij,j+1}^{-C_2} \geq 0.0001 \\
& \sum_{i=1}^{3} \sum_{j=1}^{3} w_{ij,j+1}^{C_2} = 1, \\
& \sum_{i}^{3} \sum_{j=1}^{3} w_{ij,j+1}^{-C_2} = 1, \\
& \forall i = 1, 2, 3, \quad \forall j = 1, \ldots, 4, \\
& e(a_3), e(a_4), e(a_5), e(a_6) \geq 0.
\end{array}\right.$$

Table 5 presents the global utilities of the alternatives according to the solution obtained by LP1 in this second stage.

The alternatives are correctly classified in their original classes, and therefore, it is not necessary to proceed with LP2 (similarly to the first stage). Instead, the method proceeds in solving LP3 to achieve better discrimination of the alternatives.

$$\left\{\begin{array}{ll}
\max & d \\
\text{s.t.} & U^{C_2}(a_3) - U^{-C_2}(a_3) - d \geq 0.001 \\
& U^{C_2}(a_4) - U^{-C_2}(a_4) - d \geq 0.001 \\
& U^{-C_2}(a_5) - U^{C_2}(a_5) - d \geq 0.001 \\
& U^{-C_2}(a_6) - U^{C_2}(a_6) - d \geq 0.001 \\
& w_{ij,j+1}^{C_2} \geq 0.0001, \quad w_{ij,j+1}^{-C_2} \geq 0.0001, \\
& \sum_{i=1}^{3} \sum_{j=1}^{3} w_{ij,j+1}^{C_2} = 1 \\
& \sum_{i=1}^{3} \sum_{j=1}^{3} w_{ij,j+1}^{-C_2} = 1, \\
& \forall i = 1, 2, 3, \quad \forall j = 1, \ldots, 4, \\
& d \geq 0.
\end{array}\right.$$

Table 6 presents the global utilities calculated according to the solution of LP3.

In this point the hierarchical discrimination procedure ends, since all the alternatives have been classified in the three predefined classes. Moreover, this classification is correct. In particular, in stage 1 a_1 and a_2 have

Multicriteria Sorting Methods, Table 5
Global utilities obtained through the solution of LP1 (stage 2)

	$U^{C_2}(a)$	$U^{-C_2}(a)$
a_3	0.8944	0.1000
a_4	0.7333	0.2501
a_5	0.2111	0.8000
a_6	0.1612	0.7500

Multicriteria Sorting Methods, Table 6
Global utilities obtained through the solution of LP3 (stage 2)

	$U^{C_2}(a)$	$U^{-C_2}(a)$
a_3	0.9999	0.0005
a_4	0.9997	0.0003
a_5	0.0002	0.9996
a_6	0.0005	0.7949

been correctly classified in class C_1, while in stage 2 a_3 and a_4 have been correctly classified in class C_2, and a_5 and a_6 have been classified into the final class C_3 (cf. Table 6).

Concluding Remarks and Future Perspectives

The focal point of interest in this article was the application of MCDA in the study of sorting or more generally discrimination (classification) problems. Such types of problems have major practical interest in several fields including finance, environmental and energy policy and planning, marketing, medical diagnosis, robotics (pattern recognition), etc. The multivariate statistical classification techniques have been used for decades to study such problems. However, their inability to provide a realistic and flexible approach to support real world decision making problems in situations where classification is required, led operational researchers, management scientists as well as practitioners towards the exploitation of the recent advances in the fields of operations research, management science, and artificial intelligence.

Among these 'alternative' approaches for the study of classification problem, MCDA provides an arsenal of tools and methods to develop classification (sorting) models within a realistic and flexible context. This article outlined the main MCDA classification techniques, both from the specific type of classification problems that they address (ordered or non-ordered classes), as well as from the MCDA approach that they employ (goal programming, outranking relations, preference disaggregation).

Furthermore, a new MCDA approach has been proposed. The M.H.DIS method, extends the common two-group classification framework, through a hierarchical multigroup discrimination procedure, taking into account three main discrimination criteria through a sequential process. In this way the classification problem is studied globally, in order to achieved the higher possible classification accuracy. Except for the illustrative example used in this paper, the M.H.DIS method has already been used in several financial classification problems, including the evaluation of bankruptcy risk, portfolio selection and management, the evaluation of bank branches efficiency, the assessment of country risk, company mergers and acquisitions, etc. [43], providing very encouraging results compared to well known statistical techniques (discriminant analysis, logit and probit analysis), and MCDA preference disaggregation techniques (family of UTADIS methods).

An interesting further research direction would be the exploration of a possible combination of M.H.DIS with artificial intelligence techniques such as fuzzy sets, in order to consider the fuzziness which may exist on the evaluation of alternatives on each evaluation criterion, or on the classification of the alternatives.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Abad PL, Banks WJ (1993) New LP based heuristics for the classification problem. Europ J Oper Res 67:88–100
2. Bajgier SM, Hill AV (1982) A comparison of statistical and linear programming approaches to the discriminant problem. Decision Sci 13:604–618
3. Banks WJ, Abad PL (1991) An efficient optimal solution algorithm for the classification problem. Decision Sci 22:1008–1023
4. Charnes A, Cooper WW (1961) Managem. models and industrial applications of linear programming. Wiley, New York
5. Choo E-U, Wedley WC (1985) Optimal criterion weights in repetitive multicriteria decision-making. J Oper Res Soc 36(11):983–992
6. Devaud JM, Groussaud G, Jacquet–Lagrèze E (1980) UTADIS: Une méthode de construction de fonctions d'utilité additives rendant compte de jugements globaux. Europ Working Group on Multicriteria Decision Aid, Bochum
7. Doumpos M, Zopounidis C (1998) The use of the preference disaggregation analysis in the assessment of financial risks. Fuzzy Economic Rev 3(1):39–57
8. Fisher RA (1936) The use of multiple measurements in taxonomic problems. Ann Eugenics 7:179–188
9. Freed N, Glover F (1981) A linear programming approach to the discriminant problem. Decision Sci 12:68–74
10. Freed N, Glover F (1981) Simple but powerful goal programming models for discriminant problems. Europ J Oper Res 7:44–60
11. Freed N, Glover F (1986) Evaluating alternative linear programming models to solve the two-group discriminant problem. Decision Sci 17:151–162
12. Glover F (1990) Improved linear programming models for discriminant analysis. Decision Sci 21:771–785
13. Glover F, Keene S, Duea B (1988) A new class of models for the discriminant problem. Decision Sci 19:269–280
14. Gochet W, Stam A, Srinivasan V, Chen S (1997) Multigroup discriminant analysis using linear programming. Oper Res 45(2):213–225
15. Horsky D, Rao MR (1984) Estimation of attribute weights from preference comparisons. Managem Sci 30(7):801–822
16. Jacquet–Lagrèze E (1995) An application of the UTA discriminant model for the evaluation of R&D projects. In: Pardalos PM, Siskos Y, Zopounidis C (eds) Advances in Multicriteria Analysis. Kluwer, Dordrecht, pp 203–211
17. Jacquet–Lagrèze E, Siskos Y (1982) Assessing a set of additive utility functions for multicriteria decision making, the UTA method. Europ J Oper Res 10:151–164
18. Joachimsthaler EA, Stam A (1988) Four approaches to the classification problem in discriminant analysis: An experimental study. Decision Sci 19:322–333
19. Khoury NT, Martel JM (1990) The relationship between risk-return characteristics of mutual funds and their size. Finance 11(2):67–82
20. Koehler GJ, Erenguc SS (1990) Minimizing misclassifications in linear discriminant analysis. Decision Sci 21:63–85
21. Mangasarian OL (1968) Multisurface method for patter separation. IEEE Trans Inform Theory IT-14(6):801–807
22. Markowski CA, Markowski EP (1987) An experimental comparison of several approaches to the discriminant problem with both qualitative and quantitative variables. Europ J Oper Res 28:74–78
23. Markowski EP, Markowski CA (1985) Some difficulties and improvements in applying linear programming formulations to the discriminant problem. Decision Sci 16:237–247
24. Massaglia M, Ostanello A (1991) N-TOMIC: A decision support for multicriteria segmentation problems. In: Korhonen P (ed) Internat. Workshop Multicriteria Decision Support, vol 356. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 167–174
25. Mousseau V, Slowinski R (1998) Inferring an ELECTRE-TRI model from assignment examples. J Global Optim 12(2):157–174
26. Nakayama H, Kagaku N (1992) Pattern classification by linear goal programming and its extensions. J Global Optim 12(2):111–126
27. Pekelman D, Sen SK (1974) Mathematical programming models for the determination of attribute weights. Managem Sci 20(8):1217–1229
28. Perny P (1998) Multicriteria filtering methods based on concordance and non-discordance principles. Ann Oper Res 80:137–165
29. Ragsdale CT, Stam A (1991) Mathematical programming formulations for the discriminant problem: An old dog does new tricks. Decision Sci 22:296–307
30. Roy B (1985) Méthodologie multicritère d'aide à la décision. Economica, Paris
31. Roy B, Moscarola J (1977) Procédure automatique d'examem de dossiers fondée sur une segmentation trichotomique en présence de critéres multiples. RAIRO Rech Opérat 11(2):145–173
32. Rubin PA (1990) A comparison of linear programming and parametric approaches to the two- group discriminant problem. Decision Sci 21:373–386
33. Rubin PA (1990) Heuristic solution procedures for a mixed-integer programming discriminant model. Managerial and Decision Economics 11:255–266
34. Smith C (1947) Some examples of discrimination. Ann Eugenics 13:272–282
35. Srinivasan V, Shocker AD (1973) Estimating the weights for multiple attributes in a composite criterion using pairwise judgements. Psychometrika 38(4):473–493
36. Srinivasan V, Shocker AD (1973) Linear programming techniques for multidimensional analysis of preferences. Psychometrika 38(3):337–396

37. Stam A, Joachimsthaler EA (1989) Solving the classification problem via linear and nonlinear programming methods. Decision Sci 20:285–293
38. Stam A, Joachimsthaler EA (1990) A comparison of a robust mixed-integer approach to existing methods for establishing classification rules for the discriminant problem. Europ J Oper Res 46:113–122
39. Wilson JM (1996) Integer programming formulation of statistical classification problems. OMEGA Internat J Management Sci 24(6):681–688
40. Yu W (1992) ELECTRE TRI: Aspects methodologiques et manuel d'utilisation. Document du Lamsade (Univ Paris-Dauphine) 74
41. Zopounidis C, Doumpos M (1997) A multicriteria decision aid methodology for the assessment of country risk. In: Zopounidis C, Garcia Vázquez JM (eds) Managing in Uncertainty. Proc VI Internat Conf AEDEM, AEDEM Ed, pp 223–236
42. Zopounidis C, Doumpos M (1997) Preference disaggregation methodology in segmentation problems: The case of financial distress. In: Zopounidis C (ed) New Operational Approaches for Financial Modelling. Physica Verlag, Heidelberg, pp 417–439
43. Zopounidis C, Doumpos M (1998) A multi-group hierarchical discrimination method for managerial decision problems: The M.H.DIS method. In: Paper Presented at the EURO XVI Conf.: Innovation and Quality of Life, Brussels, 12–15 July

Multidimensional Assignment Problem

MAP

Alla R. Kammerdiner
Department of Industrial and Systems Engineering, University of Florida, Gainesville, USA

MSC2000: 90C10, 90C27

Article Outline

Keywords and Phrases

Multidimensional assignment problem; Zero-one integer programming; Combinatorial optimization

Introduction

The multidimensional assignment problem (MAP) can be viewed as a higher-dimensional extension of the linear assignment problem (LAP). While the LAP is often explained as assigning each person in a group a specific job so that for each job there is only one person who does it, and for each person there is only one assigned job. The MAP generalizes evidently two-dimensional (people, jobs) LAP by allowing additional dimensions (space, time, etc.) Hence, the previous example of scheduling people to jobs can be extended to scheduling people to jobs at various time intervals in different locations, so that each specific parameter (say, time interval) is coupled with its own unique three other parameters (person, job, location) and none of them are in any other assignment (of a person, a job, a time slot and a person). Such a modified assignment problem is an example of a MAP in four dimensions.

Obviously, the LAP is a special case of the MAP in two dimensions. On the other hand, the MAP (sometimes referred to as multi-index assignment problem) is a special case of the multi-index transportation problem, just like the LAP is a particular instance of the more general transportation problem.

Interestingly, a broader class of multidimensional transportation problems was originally considered about a decade before the LAP was first given its multidimensional generalization. In fact, a three-dimensional case of the multi-index transportation problem was first introduced by Schell in 1955 [33], and later by Haley [19] in 1963. The MAP was initially presented by Pierskalla [26] in 1966, through first extending the LAP to its three-dimensional case, and then (in 1968) as a general formulation of MAP in n dimensions [27].

Despite the fact that the LAP can be solved in polynomial time, the MAP of dimensionality $d \geq 3$ is known to be NP-hard in general (the latter statement follows from a reduction of the matching problem in three dimensions) [16]. In fact, the size of the MAP increases extremely fast with an increase in dimensions. To be more precise, the size of problem grows by products of factorials. As a result of the inherent complexity of the problem, only small to medium-sized instances of the MAP can be solved routinely at the moment. Most of the exact and heuristic algorithms developed for this problem are enumerative in nature and/or utilize some form of local neighborhood search. Although many real-life applications of the MAP, including the data association problem in target tracking, require solving general problems of dimensions higher than three, most of the proposed solution methods deal with widely studied three-dimensional versions such as axial 3-MAP and planar 3-MAP.

Formulation

Several alternative formulations of the MAP have been given since Pierskalla introduced it as a 0-1 integer programming problem as follows.

Given $1 \leq p_1 \leq \ldots \leq p_d \leq n$, a finite sequence of positive integers, we want to

$$\begin{aligned}
\text{minimize} \quad & \sum_{1\leq i_1\leq p_1} \cdots \sum_{1\leq i_d\leq p_d} c_{i_1\ldots i_d} \cdot x_{i_1\ldots i_d} \\
\text{subject to} \quad & \sum_{1\leq i_2\leq p_2} \cdots \sum_{1\leq i_d\leq p_d} x_{i_1\ldots i_d} \\
& \qquad = 1,\ 1 \leq i_1 \leq p_1\,, \\
& \sum_{1\leq i_1\leq p_1} \cdots \sum_{1\leq i_{k-1}\leq p_{k-1}} \sum_{1\leq i_{k+1}\leq p_{k+1}} \cdots \\
& \qquad \sum_{1\leq i_d\leq p_d} x_{i_1\ldots i_d} = 1\,, \\
& \qquad 1 \leq i_k \leq p_k\,, \quad 2 \leq k \leq d-1\,, \\
& \sum_{1\leq i_1\leq p_1} \cdots \sum_{1\leq i_{d-1}\leq p_{d-1}} x_{i_1\ldots i_d} = 1\,, \\
& \qquad 1 \leq i_d \leq p_d\,, \\
& x_{i_1\ldots i_d} \in \{0,\,1\}, \quad 1 \leq i_k \leq p_k,\ 1 \leq k \leq d\,,
\end{aligned} \tag{1}$$

where $c_{i_1\ldots i_d}$ are the cost coefficients.

By introducing dummy variables, we can assume without loss of generality that $p_1 = \ldots = p_d = n$; then the d-dimensional assignment problem can be reformulated as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_{1\leq i_1\leq n} \cdots \sum_{1\leq i_d\leq n} c_{i_1\ldots i_d} \cdot x_{i_1\ldots i_d} \\
\text{subject to} \quad & \sum_{1\leq i_2\leq n} \cdots \sum_{1\leq i_d\leq n} x_{i_1\ldots i_d} = 1\,,\ \ 1 \leq i_1 \leq n\,, \\
& \sum_{1\leq i_1\leq n} \cdots \sum_{1\leq i_{k-1}\leq n} \\
& \sum_{1\leq i_{k+1}\leq n} \cdots \sum_{1\leq i_d\leq n} x_{i_1\ldots i_d} = 1\,, \\
& 1 \leq i_k \leq n\,, \quad 2 \leq k \leq d-1\,, \\
& \sum_{1\leq i_1\leq n} \cdots \sum_{1\leq i_{d-1}\leq n} x_{i_1\ldots i_d} = 1\,,\ \ 1 \leq i_d \leq n\,, \\
& x_{i_1\ldots i_d} \in \{0,\,1\}\,,\ \ 1 \leq i_k \leq n\,,\ \ 1 \leq k \leq d\,.
\end{aligned} \tag{2}$$

The MAP (2) also has an interesting interpretation as a problem of combinatorial optimization:

Given a d-dimensional cubic matrix, one must find the permutation of its columns and rows with the minimum sum of the diagonal elements. In other words, this is an equivalent characterization of (2) in terms of $d-1$ permutations $\pi_1, \pi_2, \ldots, \pi_{d-1}$ of the set $\{1, 2, \ldots n\}$:

$$\begin{aligned}
\text{minimize} \quad & \sum_{1\leq i\leq n} c_{i\pi_1(i)\ldots\pi_{d-1}(i)}\,, \\
& \text{subject to } \pi_1,\ \pi_2\,,\ \ldots\,,\ \pi_{d-1} \in \Pi^n\,,
\end{aligned} \tag{3}$$

where Π^n is the set of all permutations of $\{1, 2, \ldots n\}$.

Spieksma [34] gives an alternative compact formulation of the MAP as follows:

Given d sets $A_1, A_2, \ldots, A_d$, each of size n, let $A = \otimes_{i=1}^{d} A_i = A_1 \times A_2 \times \ldots \times A_d$. In other words, A is a set of all d-tuples $a = (a(1), a(2), \ldots, a(d)) \in A$. Let x_a denote a variable for each $a \in A$. Then, given assignment costs c_a for all $a \in A$, the objective function is written as $\sum_{a\in A} c_a x_a$.

Given a positive integer k, such that $1 \leq k \leq d-1$, let Q denote the set of all $(d-k)$-element subsets of $\{1, 2, \ldots, d\}$. Each subset F from Q corresponds to the set of "fixed" indices. Given such F, let $A_F = \otimes_{f\in F} A_f$. Next, given some $g \in A_F$, let $A(F, g) = \{a \in A | a(f) = g(f),\ \forall f \in F\}$ denote the set of all d-tuples that coincide with g on the set F of "fixed" indices.

Then the multi-index assignment problem can be written as:

$$\begin{aligned} \text{maximize/minimize} \quad & \sum_{a \in A} c_a x_a \\ \text{subject to} \quad & \sum_{a \in A(F,\, g)} x_a = 1, \\ & \text{for all } g \in A_F,\ F \in Q\,, \\ & x_a \in \{0,\ 1\},\ \text{for all } a \in A\,. \end{aligned} \tag{4}$$

Similarly to the linear assignment formulation by means of a bipartite graph, the MAP can also be stated using the graph theory terminology in the subsequent fashion [7]:

Given a complete d-partite graph $G = (V_1, V_2, \dots, V_d;\, E)$, where $V_i, |V_i| = n,\ i \in \{1, 2, \dots, d\}$, denote mutually disjoint vertex sets, and E is the set of edges in the graph, a subset of the vertex set $V = \cup_{i=1}^{d} V_i$ is said to be a *clique* if it meets every set V_i in exactly one vertex. A d-dimensional assignment is a partition of V into n pairwise disjoint cliques. Given a real-valued cost function c defined on the set of cliques of d-partite graph G, the d-dimensional assignment problem asks for a d-dimensional assignment, which minimizes c.

Cases

A special case of the MAP that is based on the graph theory formulation for MAP was considered by Bandelt et al. [6]. The cost function in this particular case can be represented using some type of function of elementary costs defined on the edges of the d–partite graph, whereas a general formulation of the MAP using graphs allows for the cost function to be defined arbitrarily on the set of cliques. In particular, the clique costs can be decomposed using such functions of edge costs as a sum of costs (i. e., a sum of the lengths of all the edges in a given clique), a tour cost (i. e., minimum cost of a traveling salesman tour in a given clique), a star cost (i. e., minimum length of a spanning star in a given clique), and a tree cost (i. e., minimum cost of a spanning tree). By using the decomposed costs, one can construct the worst-case bounds on the ratio between the solution costs found by a simple heuristic, as well as find the cost of the optimal solution. Specifically, Crama and Spieksma [10] considered a case of three-dimensional assignment problem, where the lengths of the edges of the underlying three-partite graph satisfy the triangle inequality, and the objective function is defined as the cost of the triangle formed by three vertices (each from a different mutually disjoint vertex subsets of the three-partite graph). When the triangle cost is defined as the length of the triangle (i. e., sum of the lengths of all its sides), then there exists a heuristic that gives a feasible solution that is within 3/2 from the optimum. The latter bound is decreased to 4/3 in the case when the triangle cost is defined as the sum of the two shortest sides.

As mentioned earlier, owing to the exponential increase in the size of the problem with an increase in the number of parameters, it becomes computationally difficult to solve MAP instances of higher dimensionality. As a result most solution methods for the MAP are constructed for three-dimensional versions of the problem. Two important types of the three-dimensional assignment problem are the *axial* three-dimensional assignment problem and the *planar* three-dimensional assignment problem. The distinction between two types lies in constraints and can be easily explained using the following simple geometric interpretation [7].

Let each solution be represented by a three-dimensional 0-1 array of size $n \times n \times n$. To visualize such an array of zeros and ones, let us fix a vertex and draw lines or axes along three dimensions. Next, we partition each axis onto n intervals. This partition splits the array into n^3 cells so that each cell contains either a 0 or a 1. Given an axis, say j, each of n intervals on j has a corresponding two-dimensional level surface that consists of $n \times n$ cells and goes through a given interval of j. Alternatively, the interval partition of each axis divides a three-dimensional solution array into n two-dimensional surfaces or "slices" corresponding to each interval on the axis. The constraints imposed in the axial case guarantee that for each axis and all of its intervals, the $n \times n$ cells in each two-dimensional slice through the interval sum up to 1. In other words, each axial interval is assigned a value of 1, which constitutes the sum of all cells that can be projected on that axial interval. This explains the name "axial."

In contrast, the constraints of the planar MAP deal with three planes formed by each possible pair of axes. For example, consider the plane formed by axes j and k. Using the above partition, this plane is divided into $n \times n$ squares. For each square on the plane, there is a corresponding stack of cells that goes along the i axis. Each cell in the stack is projected onto its square (j^*, k^*)

by axis i. The planar constraints require that for each plane and every square the sum of the cells in an associated stack is equal to 1.

Integer programming formulations for each of type of three-index MAP are given below.

Given a set of n^3 cost coefficients c_{ijk}, the axial three-dimensional assignment problem is defined as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}x_{ijk} \\ \text{subject to} \quad & \sum_{j=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1\,, \quad 1 \le i \le n\,, \\ & \sum_{i=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1\,, \quad 1 \le j \le n\,, \\ & \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ijk} = 1\,, \quad 1 \le k \le n\,, \\ & x_{ijk} \in \{0,\, 1\}\,, \quad 1 \le i,\, j,\, k \le n\,. \end{aligned} \tag{5}$$

Given n^3 cost coefficients c_{ijk}, the planar three-dimensional assignment problem can be written in the following fashion:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}x_{ijk} \\ \text{subject to} \quad & \sum_{i=1}^{n} x_{ijk} = 1, \quad 1 \le j, k \le n\,, \\ & \sum_{j=1}^{n} x_{ijk} = 1\,, \quad 1 \le i,\, k \le n\,, \\ & \sum_{k=1}^{n} x_{ijk} = 1\,, \quad 1 \le i,\, k \le n\,, \\ & x_{ijk} \in \{0,\, 1\}\,, \quad 1 \le i,\, j,\, k \le n\,. \end{aligned} \tag{6}$$

The axial three-index MAP given by (5) can also be formulated using n permutations σ and π as a combinatorial optimization problem:

$$\text{minimize} \sum_{i=1}^{n} c_{i\sigma(i)\pi(i)}\,, \text{ subject to } \sigma,\, \pi \in \Pi_n\,. \tag{7}$$

Note that the planar three-dimensional assignment problem has a different combinatorial interpretation in terms of Latin squares of order n.

Although both axial and planar three-dimensional assignment problems (just as the general MAP) are generally NP-hard, there exist a number of polynomially solvable special cases. Particularly, in the case when the cost coefficients form a so-called Monge array [8], the MAP is solved by $d-1$ identity-n permutations $\{1,\, 2,\, \dots,\, n\} \to \{1,\, 2,\, \dots,\, n\}$. Another case of the polynomially solvable MAP is the axial three-dimensional assignment problem, where the cost coefficient can be represented as a product of nonnegative index factors $c_{ijk} = p_i \cdot q_j \cdot r_k$, and the objective function is maximized [9].

Methods

All known exact methods for solving this generally NP-hard problem are enumerative in nature, and as a result of the inherent complexity of the problem such methods are too slow for practical applications of the MAP. Hence, researchers often use heuristic approaches to find suboptimal solutions of different MAPs. In fact, one of the earliest solution methods for the MAP was a suboptimal method of trisubstitution proposed by Pierskalla [26] in 1966 to solve a three-dimensional assignment problem. Later Frieze and Yadegar [15] developed a suboptimal procedure for the three-index MAP using Lagrangian relaxation. Their technique utilized information contained in the relaxed solution to recover a feasible solution. The key advantage of the Lagrangian relaxation approach is that it allows for computing both upper and lower bounds on the optimum solution, and therefore this method can be employed to evaluate solution quality. Consequently, the Lagrangian relaxation technique was widely used to propose numerous modifications of the original three-dimensional method by extending it to the general multidimensional case [12,28,29]. For example, one of such algorithms presented by Poore and Robertson [29] in 1997 works by relaxing a d-dimensional assignment problem to a two-dimensional problem, then maximizing with regard to the relaxed Lagrangian multipliers, and next formulating the recovery procedure as a $(d-1)$-dimensional problem. These three steps are repeated successively until the recovery procedure can be formulated as a two-dimensional problem, which is solved optimally in polynomial time, and the algorithm terminates.

Most exact methods for solving the MAP are devised primarily for its three-dimensional case. One of the earliest exact approaches for the axial three-dimensional assignment problem was suggested by Pierskalla [27] in 1968. His approach works by enumerating all feasible solutions using a tree structure, and utilizing the branch and bound method as follows. For a given node of the feasible solutions tree, a lower bound is calculated from the corresponding dual subproblem, before proceeding further on outgoing branches from the node. If the lower bound is greater than the known lowest bound, then the outgoing branches are eliminated, since it is impossible to obtain a better solution along such branches. Otherwise when the lower bound obtained is less than the known lowest bound, we continue further from this node, because it might still be possible to improve our solution in that direction. Although this branch and bound algorithm can easily be generalized to the multidimensional case, it is too slow to work effectively for the general MAP.

Since Pierskalla introduced his branch and bound procedure for the axial three-dimensional assignment problem, many other branch and bound based approaches have been developed. Most of them branch the current problem onto two subproblems by setting one variable $x_{ijk} = 0$ or $x_{ijk} = 1$. Then the size of the subproblems is decreased. In contrast, a branch and bound scheme proposed by Balas and Saltzman [5] permits fixing several variables at once at each branching node by incorporating a special branching strategy that takes advantage of the problem structure.

The planar three-index MAP can also be solved using variations of branch and bound. One of the first applications of this method to the planar case was given by Vlach [35] in 1967. The algorithm obtains lower bounds by means of row and column reductions that are similar to the ones in the axial case. A method for solving the planar three-dimensional assignment problem based on a clever combination of branch and bound with a relaxation heuristic and Lagrangian relaxation was developed by Magos and Miliotis [24]. The upper bounds are calculated by first applying the relaxation heuristic and then decomposing the remaining problem into n linear sum assignment problems. The lower bounds are computed by either a heuristic or a Lagrangian relaxation depending on the current problem.

The method introduced by Hansen and Kaufman [20] for solving the axial three-dimensional assignment problem employs a primal-dual method comparable to the well-known Hungarian method for the LAP.

There have been a number of investigations of a convex hull of feasible solutions of the three-dimensional assignment problem. Euler et al. [14] examined the polyhedral structure of the solution polytope for the planar three-index MAP through its connection to Latin squares. Euler [13] also studied the axial polytope by investigating the role of odd cycles for a class of facets of the polytope. The structure of the axial three-index assignment polytope was also analyzed by Balas et al. [3,4,32]. They developed linear-time separation algorithms for different classes of facets induced by specific cliques, and then constructed a polyhedral procedure for solving the axial three-index MAP.

Clemons et al. [11] applied a simulated annealing algorithm for solving the MAP. Several local neighborhood search procedures were implemented for the MAP. Greedy randomized adaptive search procedures (GRASP) were applied by Murphey et al. [25] for solving the general MAP and later by Lidstrom et al. [22] and by Aiex et al. [1] for finding solutions of the axial three-dimensional assignment problem. A tabu search for the planar three-dimensional assignment problem was employed by Magos [23] to obtain suboptimal solutions of the planar thee-index assignment problem.

Grundel and Pardalos [18] developed a test problem generator for testing exact and suboptimal solution methods for the axial MAP. Several recent studies investigated various asymptotic properties of the MAPs with randomly generated assignment cost coefficients. In particular, Grundel et al. [17] established the lower and upper bounds for the expected number of local minima of the MAPs with random costs.

Applications

The MAP can be used to solve various real-life problems arising in such important areas as capital investment, dynamic facility location, and satellite launching [30]. Other applications of the MAP include circuit board assembly and production planning of goods, which can be modeled using the axial three-dimensional assignment problem [34]. The planar three-dimensional assignment problem has also found many interesting applications, for instance, school timetables and experimental design [21], as well as modeling of satellite launching [2].

Furthermore, it was shown that such a complex problem as tracking elementary particles can be investigated using the five-dimensional assignment problem as a mathematical model [31]. By solving this complex case of the MAP, one can reconstruct the paths of charged elementary particles produced by the Large Electron–Positron Collider.

Many important applications of the general MAP arise in data association, resource allocation, air traffic control, surveillance, etc. In particular, Poore [28] has shown that the data association problem arising in a large class of multiple target tracking and sensor fusion problems can be formulated as a MAP by partitioning the set of observations into false reports and tracks, and then maximizing the likelihood of selecting the true partition.

See also

- ► Assignment and Matching
- ► Integer Programming: Branch and Bound Methods
- ► Multi-index Transportation Problems

References

1. Aiex RM, Resende MGC, Pardalos PM, Toraldo G (2005) GRASP with path relinking for the three-index assignment. INFORMS J Comput 17(2):224–247
2. Balas E, Landweer P (1983) Traffic Assignment in Commu-nication Satelites. Oper Res Lett 2:141–147
3. Balas E, Qi L (1993) Linear-time separation algorithms for three-index assignment polytope. Discr Appl Math 43: 1–12
4. Balas E, Saltzman MJ (1989) Facets of the three-index assignment polytope. Discr Appl Math 23:201–229
5. Balas E, Saltzman MJ (1991) An algorithm for three-index assignment problem. Oper Res 39:150–161
6. Bandelt HJ, Crama Y, Spieksma FCR (1994) Approximation algorithms for multidimensional assignment problems with decomposable costs. Discr Appl Math 49:25–50
7. Burkard RE, Çela E (1999) Linear Assignment Problems and Extensions. In: Pardalos P, Du D-Z (eds) The Handbook of Combinatorial Optimization. Kluwer, Dordrecht, pp 75–149
8. Burkard RE, Klinz B, Rudolf R (1996) Perspectives of Monge properties in optimization. Discr Appl Math 70:95–161
9. Burkard RE, Rudolf R, Woeginger GJ (1996) Three-dimensional axial problem with decomposable cost coefficients. Discr Appl Math 65:123–169
10. Crama Y, Spieksma FCR (1992) Approximation algorithms for three-dimensional assignment problems with triangle inequalities. Eur J Oper Res 60:273–279
11. Clemons W, Grundel D, Jeffcoat D (2004) Applying simulated annealing to the multidimensional assignment problem. In: Grundel DA, Pardalos PM, Murphey R (eds) Theory and Algorithms for Cooperative Systems. World Scientific, Singapore
12. Deb S, Pattipatti KR, Bar-Shalom Y (1992), A s-dimensional assignment algorithm for track initiation. Proceedings of the IEEE International Systems Conference, Kobe, Japan, September 127–130
13. Euler R (1987) Odd cycles and a class of facets of the axial 3-index assignment polytope. Applicationes mathimaticae (Zastowania Matematyki) 19:375–386
14. Euler R, Burkard RE, Grommes R (1986) On Latin squares and facial structure of related polytopes. Discr Math 62:155–181
15. Frieze AM, Yadegar J (1981) An algorithm for solving 3-dimensional assignment problems with application to scheduling a teaching practice. J Oper Res Soc 32:969–995
16. Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, San Francisco
17. Grundel D, Krokhmal P, Oliveira C, Pardalos P (2007) On the number of local minima for the multidimensional assignment problem. J Comb Optim 13(1):1–18
18. Grundel D, Pardalos P (2005) Test Problem Generator for the Multidimensional Assignment Problem. Comput Optim Appl 30(2):133–146
19. Haley KB (1963) The Multi-index Problem. Oper Res 11:368–379
20. Hansen P, Kaufman L (1973) A primal-dual algorithm for three-dimensional assignment problem. Cahiers du CERO 15:327–336
21. Hilton A (1980) The reconstruction of Latin Squares with Applications to School Timetabling and to Experimental Desin
22. Lidstrom N, Pardalos P, Pitsoulis L, Toraldo G (1996) An approximation algorithm for the three-index assignment problem, Technical Report
23. Magos D (1996) Tabu search for the planar three-dimensional assignment problem. J Global Optim 8:35–48
24. Magos D, Miliotis P (1994) An algorithm for the planar three-index assignment problem. Eur J Oper Res 77:141–153
25. Murphey R, Pardalos P, Pitsoulis L (1998) A greedy randomized adaptive search procedure for the multitarget multisensor tracking problem. DIMACS Ser Am Math Soc 40:277–302
26. Pierskalla WP (1967) The tri-substitution method for the three-multidimensional assignment problem. CORS J 5:71–81
27. Pierskalla WP (1968) The multidimensional assignment problem. Oper Res 16:422–431

28. Poore AB (1994) Multidimensional assignment formulation of data association problems arising from multitarget and multisensor tracking. Comput Optim Appl 3:27–54
29. Poore AB, Robertson AJ (1997) A New Lagrangian Relaxation Based Algorithm for a Class of Multidimensional Assignment Problems. Comput Optim Appl 8:129–150
30. Pierskalla WP (1967) The tri-substitution method for the three-dimensional assignment problem. J Canadian Oper Res Soc 5:71–81
31. Pusztaszeri JF, Rensing PE, Liebling TM (1996) Tracking Elementary Particles Near Their Primary Vertex: A Combinatorial Approach. J Global Optim 9(1):41–64
32. Qi L, Balas E, Gwan G (1994) A new facet class and a polyhedral method for three-index assignment problem. In: Du D-Z (ed) Advances in Optimization. Kluwer, Dordrecht, pp 256–274
33. Schell E (1955) Distribution of a Product by Several Properties, Proc. Second Symposium in Linear Programming, Washington, D.C., January 27-29, vol 1–2, pp 615–642
34. Spieksma FCR (2000) Multi index assignment problems: complexity, approximation, applications. In: Pitsoulis L, Pardalos P (eds) Nonlinear Assignment Problems, Algorithms and Applications. Kluwer, Dordrecht, pp 1–12
35. Vlach M (1967) Branch and bound method for the three-index assignment problem. Economicko-Matematicky Obzor 3:181–191

Multidimensional Knapsack Problems

John E. Beasley
The Management School, Imperial College,
London, England

MSC2000: 90C27, 90C10

Article Outline

Keywords

Multidimensional knapsack; Multiconstraint knapsack; Multiple choice knapsack; Combinatorial optimization

The *multidimensional knapsack problem* (MKP) can be formulated as:

$$\begin{cases} \max & \displaystyle\sum_{j=1}^{n} p_j x_j \\ \text{s.t.} & \displaystyle\sum_{j=1}^{n} r_{ij} x_j \le b_i, \quad i = 1, \dots, m, \\ & x_j \in \{0,1\}, \quad j = 1, \dots, n, \end{cases} \tag{1}$$

where $b_i \geq 0$, $i = 1, \dots, m$, and $r_{ij} \geq 0$, $i = 1, \dots, m$, $j = 1, \dots, n$.

Each of the m constraints in (1) is called a *knapsack constraint*, so the MKP is also called the *m-dimensional knapsack problem*.

Other names given to this problem in the literature are the *multiconstraint knapsack problem*, the *multi-knapsack problem* and the *multiple knapsack problem*. Some authors also include the term 'zero-one' in their name for the problem, e. g., the *multidimensional zero-one knapsack problem*. Historically the majority of authors have used the name *multidimensional knapsack problem* and so we also use that phrase to refer to the problem. The special case corresponding to $m = 2$ is known as the *bidimensional knapsack problem* or the *bi-knapsack problem*.

Many practical problems can be formulated as a MKP, for example, the capital budgeting problem where project j has profit p_j and consumes r_{ij} units of resource i. The goal is to find a subset of the n projects such that the total profit is maximised and all resource constraints are satisfied. Other applications of the MKP include allocating processors and databases in a distributed computer system [24], project selection and cargo loading [53], and cutting-stock problems [26].

The MKP can be regarded as a general statement of any zero-one integer programming problem with non-negative coefficients. Indeed much of the early work on the MKP (e. g., [32,35,52,59]) viewed the problem in this way.

Most of the research on knapsack problems deals with the much simpler single constraint version ($m = 1$). For the single constraint case the problem is not strongly *NP*-hard and effective approximation algorithms have been developed for obtaining near-optimal solutions. A good review of the single constraint knapsack problem and its associated exact and heuristic algorithms is given by S. Martello and P. Toth [42].

Below we give a very brief overview of the literature relating to the MKP. A more detailed literature review can be found in [10].

Exact Algorithms

There have been relatively few exact algorithms presented in the literature.

W. Shih [53] presented a branch and bound algorithm (cf. also ► Integer programming: Branch and bound methods) for the MKP with an upper bound obtained by computing the objective function value associated with the optimal fractional solution for each of the m single constraint knapsack problems separately and selecting the minimum objective function value among those as the upper bound.

Another branch and bound algorithm was presented in [25] with various relaxations of the problem, including Lagrangian, surrogate and composite relaxations being used to compute bounds. Y. Crama and J.B. Mazzola [11] showed that although the bounds derived from these relaxations are stronger than the bounds obtained from the linear programming (LP) relaxation, the improvement in the bound that can be realized using these relaxations is limited.

Statistical/Asymptotic Analysis

There have been a few papers considering a statistical/asymptotic analysis of the MKP.

An asymptotic analysis was presented by K.E. Schilling [51] who computed the asymptotic ($n \to \infty$ with m fixed) objective function value for the MKP where the r_{ij}'s and p_j's were uniformly (and independently) distributed over the unit interval and where $b_i = 1$. K. Szkatula [54] generalized that analysis to the case where $b_i \neq 1$ (see also [55]).

A statistical analysis was conducted by J.F. Fontanari [18], who investigated the dependence of the objective function on b_i and on m, in the case when $p_j = 1$ and the r_{ij}'s were uniformly distributed over the unit interval.

Early Heuristic Algorithms

Early heuristic algorithms for the MKP were typically based upon simple constructive heuristics.

S.H. Zanakis [59] gave detailed results comparing three algorithms from [32,35] and [52]. R. Loulou and E. Michaelides [40] presented a greedy-like method based on *Toyoda's primal heuristic* [57]. Primal heuristics start with a zero solution, after which a succession of variables are assigned the value one, according to a given rule, as long as the solution remain feasible.

Bound Based Heuristics

Bound based heuristics make use of an upper bound on the optimal solution to the MKP.

M.J. Magazine and O. Oguz [41] presented a heuristic algorithm that combines the ideas of S. Senju and Toyoda's dual heuristic [52] with *Everett's generalized Lagrange multiplier approach* [17]. Dual heuristics start with the all-ones solution, variables are then successively set to zero according to heuristic rules until a feasible solution is obtained. Their algorithm computes an approximate solution and uses the multipliers generated to obtain an upper bound.

H. Pirkul [45] presented a heuristic algorithm which makes use of surrogate duality. The m knapsack constraints were transformed into a single knapsack constraint using surrogate multipliers. A feasible solution was obtained by packing this single knapsack in decreasing order of profit/weight ratios. These ratios were defined as $p_j / \sum_{i=1}^{m} \omega_i r_{ij}$, where ω_i is the surrogate multiplier for constraint i. Surrogate multipliers were determined using a descent procedure.

J.S. Lee and M. Guignard [36] presented a heuristic that combined Toyoda's primal heuristic [57] with variable fixing, LP and a complementing procedure from [6].

A. Volgenant and J.A. Zoon [58] extended the heuristic in [41] in two ways:

1) in each step, not one, but more, multiplier values are computed simultaneously; and
2) at the end of the procedure the upper bound is sharpened by changing some multiplier values.

A. Freville and G. Plateau [21] presented an efficient preprocessing algorithm for the MKP, based on [20], which provided sharp lower and upper bounds on the optimal value, and also a tighter equivalent representation by reducing the continuous feasible set and by eliminating constraints and variables.

They also [22] presented a heuristic for the bidimensional knapsack problem which includes problem reduction, a bound based upon surrogate relaxation and partial enumeration.

Tabu Search Heuristics

Tabu search (TS) heuristics are based on tabu search concepts (see [1,29,46]).

F. Dammeyer and S. Voß [12] presented a TS heuristic based on reverse elimination. R. Aboudi and K. Jörnsten [2] combined TS with the pivot and complement heuristic [6] in a heuristic that they applied to the MKP (see also [39]). R. Battiti and G. Tecchiolli [7] presented a heuristic based on reactive TS (essentially TS but with the length of the tabu list varied over the course of the algorithm).

F. Glover and G.A. Kochenberger [28] presented a TS heuristic with a flexible memory structure that integrates recency and frequency information keyed to 'critical events' in the search process. Their method was enhanced by a strategic oscillation scheme that alternates between constructive (current solution feasible) and destructive (current solution infeasible) phases. See also [30].

A. Løkketangen and Glover [37] presented a heuristic based on probabilistic TS (essentially TS but with the acceptance/rejection of a potential move controlled by a probabilistic process). They also [38] presented a TS heuristic designed to solve general zero-one mixed integer programming problems which they applied to the MKP.

Genetic Algorithm Heuristics

Genetic algorithm (GA) heuristics are based on genetic algorithm concepts (see [1,8,43,46]).

In the GA of [34] infeasible solutions were allowed to participate in the search and a simple fitness function which uses a graded penalty term was used. In [56] simple heuristic operators based on local search algorithms were used, and a hybrid algorithm based on combining a GA with a TS heuristic was suggested.

In [48,49] a GA was presented where parent selection is not unrestricted (as in a standard GA) but is restricted to be between 'neighboring' solutions. Infeasible solutions were penalized as in [34]. An adaptive threshold acceptance schedule (motivated by [14,15]) for child acceptance was used.

In the GA of [33] only feasible solutions were allowed. P.C. Chu and J.E. Beasley [10] presented a GA based upon a simple repair operator to ensurethat all solutions were feasible.

Analysed Heuristics

Analysed heuristics have some theoretical underlying analysis relating to their worst-case or probabilistic performance.

A.M. Frieze and M.R.B. Clarke [23] described a polynomial approximation scheme based on the use of the dual simplex algorithm for LP, and analysed the asymptotic properties of a particular random model.

In [47] a class of generalized greedy algorithms is proposed in which items are selected according to decreasing ratios of their p_j's and a weighted sum of their r_{ij}'s. These heuristics were subjected to both a worst-case, and a probabilistic, performance analysis.

I. Averbakh [5] investigated the properties of several dual characteristics of the MKP for different probabilistic models. He also presented a fast statistically efficient approximate algorithm with linear running time complexity for problems with random coefficients.

Other Heuristics

G.E. Fox and G.D. Scudder [19] presented a heuristic based on starting from setting all variables to zero(one) and successively choosing variables to set to one(zero). See [13] for a heuristic based upon simulated annealing (SA). See [27] for a heuristic based on ghost image processes. S. Hanafi and others [31] presented a simple multistage algorithm within which a number of different local search procedures (such as greedy, SA, threshold accepting [14,15] and noising [9]) can be used. They also presented two TS heuristics.

Multiple–Choice Problems

One problem that is related to the MKP is the *multidimensional multiple-choice knapsack problem* (MMKP). Suppose that $\{1, \dots, n\}$ is divided up into K sets S_k, $k = 1, \dots, K$, which are mutually exclusive $S_k \cap S_l = \emptyset$, $\forall k \neq l$, and exhaustive $\cup_{k=1}^{K} S_k = \{1, \dots, n\}$. If we then add to the formulation of the MKP given previously the constraint

$$\sum_{j \in S_k} x_j = 1, \quad k = 1, \dots, K, \tag{2}$$

we obtain the MMKP. Equation (2) ensures that exactly one variable is chosen from each of the sets S_k, $k = 1, \dots, K$.

See [44] for a heuristic for MMKP based on the MKP heuristic of Magazine and Oguz [41].

The special case of the MMKP corresponding to $m = 1$ is known as the *multiple-choice knapsack problem* (MCKP) and its LP relaxation as the *linear multiple-choice knapsack problem* (LMCKP). Work on MCKP includes [16], which presented a hybrid dynamic programming tree search algorithm incorporating a Lagrangian relaxation bound; [4], which presented a heuristic based upon SA; and [3], which presented a tree search algorithm incorporating a Lagrangian relaxation bound. For work on LMCKP see [50]. Earlier work on MCKP and LMCKP is cited in [3, 4,16,50].

See also

- ► Integer Programming
- ► Quadratic Knapsack

References

1. Aarts EHL, Lenstra J (eds) (1997) Local search in combinatorial optimization. Wiley, New York
2. Aboudi R, Jörnsten K (1994) Tabu search for general zero-one integer programs using the pivot and complement heuristic. ORSA J Comput 6:82–93
3. Aggarwal V, Deo N, Sarkar D (1992) The knapsack problem with disjoint multiple-choice constraints. Naval Res Logist 39:213–227
4. Al-Sultan K (1994) A new approach to the multiple-choice knapsack problem. Proc 16th Internat Conf Computers and Industr Engineering, pp 548–550
5. Averbakh I (1994) Probabilistic properties of the dual structure of the multidimensional knapsack problem and fast statistically efficient algorithms. Math Program 65:311–330
6. Balas E, Martin CH (1980) Pivot and complement – a heuristic for 0–1 programming. Managem Sci 26:86–96
7. Battiti R, Tecchiolli G (1995) Local search with memory: Benchmarking RTS. OR Spektrum 17:67–86
8. Bäck T, Fogel DB, Michalewicz Z (eds) (1997) Handbook of evolutionary computation. Oxford Univ. Press, Oxford
9. Charon I, Hudry O (1993) The noising method: A new method for combinatorial optimization. Oper Res Lett 14:133–137
10. Chu PC, Beasley JE (1998) A genetic algorithm for the multidimensional knapsack problem. J Heuristics 4:63–86
11. Crama Y, Mazzola JB (1994) On the strength of relaxations of multidimensional knapsack problems. INFOR 32:219–25
12. Dammeyer F, Voss S (1993) Dynamic tabu list management using reverse elimination method. Ann Oper Res 41:31–46
13. Drexl A (1988) A simulated annealing approach to the multiconstraint zero-one knapsack problem. Computing 40:1–8
14. Dueck G (1993) New optimization heuristics: the grand deluge algorithm and the record-to-record travel. J Comput Phys 104:86–92
15. Dueck G, Scheuer T (1990) Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. J Comput Phys 90:161–175
16. Dyer ME, Riha WO, Walker J (1995) A hybrid dynamic programming/branch-and-bound algorithm for the multiple-choice knapsack problem. J Comput Appl Math 58:43–54
17. Everett H (1963) Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. Oper Res 11:399–417
18. Fontanari JF (1995) A statistical analysis of the knapsack problem. J Phys A: Math Gen 28:4751–4759
19. Fox GE, Scudder GD (1985) A heuristic with tie breaking for certain 0–1 integer programming models. Naval Res Logist Quart 32:613–623
20. Freville A, Plateau G (1986) Heuristics and reduction methods for multiple constraints 0–1 linear programming problems. Europ J Oper Res 24:206–215
21. Freville A, Plateau G (1994) An efficient preprocessing procedure for the multidimensional 0–1 knapsack problem. Discrete Appl Math 49:189–212
22. Freville A, Plateau G (1997) The 0–1 bidimensional knapsack problem: toward an efficient high-level primitive tool. J Heuristics 2:147–167
23. Frieze AM, Clarke MRB (1984) Approximation algorithms for the m-dimensional 0–1 knapsack problem: worst-case and probabilistic analysis. Europ J Oper Res 15:100–109
24. Gavish B, Pirkul H (1982) Allocation of databases and processors in a distributed computing system. In: Akoka J (ed) Managem. of Distributed Data Processing. North-Holland, Amsterdam, pp 215–231
25. Gavish B, Pirkul H (1985) Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality. Math Program 31:78–105
26. Gilmore PC, Gomory RE (1966) The theory and computation of knapsack functions. Oper Res 14:1045–1075
27. Glover F (1994) Optimization by ghost image processes in neural networks. Comput Oper Res 21:801–822
28. Glover F, Kochenberger GA (1996) Critical event tabu search for multidimensional knapsack problems. In: Os-

man IH, Kelly JP (eds) Meta–Heuristics: Theory and Applications. Kluwer, Dordrecht, pp 407–427
29. Glover FW, Laguna M (1997) Tabu search. Kluwer, Dordrecht
30. Hanafi S, Freville A (1998) An efficient tabu search approach for the 0–1 multidimensional knapsack problem. Europ J Oper Res 106:659–675
31. Hanafi S, Freville A, AbedellaouiEl A (1996) Comparison of heuristics for the 0–1 multidimensional knapsack problem. In: Osman IH, Kelly JP (eds) Meta–Heuristics: Theory and Applications. Kluwer, Dordrecht, pp 449–465
32. Hillier FS (1969) Efficient heuristic procedures for integer linear programming with an interior. Oper Res 17:600–637
33. Hoff A, Løkketangen A, Mittet I (1996) Genetic algorithms for 0/1 multidimensional knapsack problems. Working Paper, Molde College, Molde
34. Khuri S, Bäck T, Heitkötter J (1994) The zero/one multiple knapsack problem and genetic algorithms. Proc. 1994 ACM Symp. Applied Computing (SAC'94), ACM, New York, pp 188–193
35. Kochenberger GA, McCarl BA, Wymann FP (1974) A heuristic for general integer programming. Decision Sci 5:36–44
36. Lee JS, Guignard M (1988) An approximate algorithm for multidimensional zero-one knapsack problems - a parametric approach. Managem Sci 34:402–410
37. Løkketangen A, Glover F (1996) Probabilistic move selection in tabu search for zero-one mixed integer programming problems. In: Osman IH, Kelly JP (eds) Meta-Heuristics: Theory and Applications. Kluwer, Dordrecht, pp 467–487
38. Løkketangen A, Glover F (1997) Solving zero-one mixed integer programming problems using tabu search. Europ J Oper Res 106:624–658
39. Løkketangen A, Jörnsten K, Storøy S (1994) Tabu search within a pivot and complement framework. Internat Trans Oper Res 1:305–316
40. Loulou R, Michaelides E (1979) New greedy-like heuristics for the multidimensional 0–1 knapsack problem. Oper Res 27:1101–1114
41. Magazine MJ, Oguz O (1984) A heuristic algorithm for the multidimensional zero-one knapsack problem. Europ J Oper Res 16:319–326
42. Martello S, Toth P (1990) Knapsack problems: Algorithms and computer implementations. Wiley, New York
43. Mitchell M (1996) An introduction to genetic algorithms. MIT, Cambridge
44. Moser M, Jokanovic DP, Shiratori N (1997) An algorithm for the multidimensional multiple-choice knapsack problem. IEICE Trans Fundam Electronics, Commun and Computer Sci E80A:582–589
45. Pirkul H (1987) A heuristic solution procedure for the multiconstraint zero-one knapsack problem. Naval Res Logist 34:161–172
46. Reeves CR (1993) Modern heuristic techniques for combinatorial problems. Blackwell, Oxford
47. Rinnooy Kan AHG, Stougie L, Vercellis C (1993) A class of generalized greedy algorithms for the multi-knapsack problem. Discrete Appl Math 42:279–290
48. Rudolph G, Sprave J (1995) A cellular genetic algorithm with self-adjusting acceptance threshold. Proc. First IEE/IEEE Internat. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications, IEEE, New York, pp 365–372
49. Rudolph G, Sprave J (1996) Significance of locality and selection pressure in the grand deluge evolutionary algorithm. In: Voigt HM, Ebeling W, Rechenberg I, Schwefel HP (eds) Parallel Problem Solving from Nature IV Proc Internat Conf Evolutionary Computation, Lecture notes Computer Sci. Springer, Berlin, pp 686–694
50. Sarin S, Karwan MH (1989) The linear multiple choice knapsack problem. Oper Res Lett 8:95–100
51. Schilling KE (1990) The growth of m-constraint random knapsacks. Europ J Oper Res 46:109–112
52. Senju S, Toyoda Y (1968) An approach to linear programming with 0–1 variables. Managem Sci 15:196–207
53. Shih W (1979) A branch and bound method for the multiconstraint zero-one knapsack problem. J Oper Res Soc 30:369–378
54. Szkatula K (1994) The growth of multi-constraint random knapsacks with various right-hand sides of the constraints. Europ J Oper Res 73:199–204
55. Szkatula K (1997) The growth of multi-constraint random knapsacks with large right-hand sides of the constraints. Oper Res Lett 21:25–30
56. Thiel J, Voss S (1994) Some experiences on solving multiconstraint zero-one knapsack problems with genetic algorithms. INFOR 32:226–242
57. Toyoda Y (1975) A simplified algorithm for obtaining approximate solutions to zero-one programming problems. Managem Sci 21:1417–1427
58. Volgenant A, Zoon JA (1990) An improved heuristic for multidimensional 0–1 knapsack problems. J Oper Res Soc 41:963–970
59. Zanakis SH (1977) Heuristic 0–1 linear programming: An experimental comparison of three methods. Managem Sci 24:91–104

Multidisciplinary Design Optimization

MDO

LAYNE T. WATSON
Virginia Polytechnic Institute and State University,
Blacksburg, USA

MSC2000: 65F10, 65F50, 65H10, 65K10

Article Outline

Keywords

Collaborative; Concurrent subspace; Multidisciplinary design; Multipoint approximation; Response surface; DACE

Modern large scale vehicle design (aircraft, ships, automobiles, mass transit) requires the interaction of multiple disciplines, traditionally processed in a sequential order. *Multidisciplinary optimization* (MDO), a formal methodology for the integration of these disciplines, is evolving toward methods capable of replacing the traditional sequential methodology of vehicle design by concurrent algorithms, with both an overall gain in product performance and a decrease in design time. The obstacles to MDO becoming a production methodology, in the same sense as quality control, are numerous and formidable. In aircraft design, for instance, typical disciplines involved would be aerodynamics, structures, thermodynamics, controls, propulsion, manufacture, and economics. Detailed analyses in each of these disciplines could involve tens to hundreds of subroutines and tens of thousands of lines of code. Managing the software libraries and data alone is a daunting task.

Codes from different disciplines typically are grossly incompatible, but even within disciplines, data structures and solution representations may be incompatible, requiring 'translation' routines or recoding. This incompatibility is particularly acute when stand-alone packages with interactive interfaces are involved. Most disciplinary codes, designed years ago for small serial computers, are very ill-suited to modern parallel architectures, even with a coarse grained approach.

Detailed, highly accurate disciplinary analyses are very expensive, requiring sometimes hours on a supercomputer, even when run in parallel. The import of this is that, regardless of the dimension of the design space, it can be sampled for accurate function values at only a relatively small number of points. Other obstacles to achieving true MDO include model verification, noisy function values, and flawed parallel optimization methodologies.

Almost every conceivable strategy for MDO has been proposed. A good recent summary of hierarchical approaches can be found in [4], and [9] pioneered nonhierarchical or concurrent approaches. The basic idea of concurrent methods, and a particular variant known as *concurrent subspace optimization* (CSSO), is to simultaneously and independently optimize each of the disciplines (or 'contributing analyses', as they are called), and then perform a global coordination that brings the entire system closer to a globally feasible and optimal point. *Collaborative optimization* differs from CSSO in how the global coordination is managed. An excellent discussion of these approaches is in the proceedings [2]. While concurrent methods are intuitively appealing and naturally parallelizable, they are not guaranteed to converge [8].

Trust region model management [1] is a rigorous approach to MDO that shows promise, and aspects of CSSO when combined with an extended Lagrangian and response surface approximations, can lead to a provably convergent MDO method (J.F. Rodríguez, J.E. Renaud and L.T. Watson, [6]). A noteworthy aspect of the *Rodríguez method* [6] is that the convergence proof covers variable fidelity data, which is crucial in practice.

In a taxonomy of MDO approaches, one distinction would be between hierarchic or nonhierarchic. Another distinction is whether parallelism is achieved between disciplines (concurrent disciplinary computation) or within disciplines (multipoint, response surface, local/global computation). If response surface approximations are used, two prevalent approximation methods are classical least squares and *DACE* (Design and Analysis of Computer Experiments).

S. Burgee, A.A. Giunta, V. Balabanov, B. Grossman, W.H. Mason, R. Narducci, R.T. Haftka, and Watson [3] has a detailed discussion of the multipoint, classical least squares approach to response surface construction, and of the use of parallelism within disciplines (the pipelined MDO paradigm of Burgee is also provably convergent). The tack of this approach is to use classical design of experiments theory, regression statistics, and low order polynomial approximation models.

The DACE [7] model posits that the output of a computer analysis program is

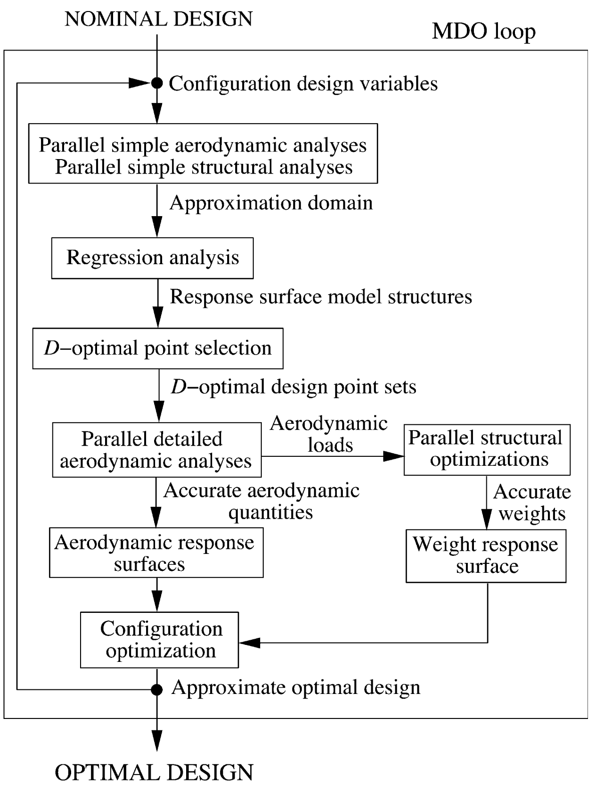
$$Y(x) = \beta + Z(x),$$

where $Z(x)$ is a zero mean stationary Gaussian process. (This is clearly a fiction since computer output is deterministic. The issue is whether the model has predictive power.) Using Bayesian statistics, the best unbiased predictor is

$$\widehat{Y}(x) = \widehat{\beta} + r(x,S)R^{-1}(Y_S - 1 \cdot \widehat{\beta}),$$

where S is a set of observation sites, Y_S is the vector of observations at S, $r(x,S)$ is the correlation of x with sites S, R is the correlation matrix between sites S, and $\widehat{\beta}$ is the estimate of the mean. Some parametrized functional form for the correlation is assumed, and then these correlation parameters and $\widehat{\beta}$ are computed as maximum likelihood estimates.

DACE models are more flexible than polynomial models, but with sparse data in high dimensions neither DACE nor polynomial models have much predictive power. To appreciate the problem, observe that a cube in 30 dimensions has $2^{30} \approx 10^9$ vertices, and to even evaluate an algebraic formula at each vertex requires supercomputer power.


NOMINAL DESIGN
MDO loop
Configuration design variables
Parallel simple aerodynamic analyses
Parallel simple structural analyses
Approximation domain
Regression analysis
Response surface model structures
D–optimal point selection
D–optimal design point sets
Parallel detailed aerodynamic analyses
Aerodynamic loads
Parallel structural optimizations
Accurate aerodynamic quantities
Accurate weights
Aerodynamic response surfaces
Weight response surface
Configuration optimization
Approximate optimal design
OPTIMAL DESIGN

Multidisciplinary Design Optimization, Figure 1
MDO paradigm

MDO Paradigm Example

As an illustration, an MDO paradigm for aircraft design is presented here. The MDO algorithm is a repeat loop, with a nominal design as its starting point, approximate optimal designs as loop iterates, and an optimal design as its ending point (see Fig. 1). At the start of each loop, aerodynamic shape and mission variables are obtained from either the nominal starting design or the intermediate approximate optimal design. These shape and mission variables are then used in the parallel simple aerodynamic and structural analyses.

The simple aerodynamic analyses are performed on a regular grid of points in the design space. Simple aerodynamic calculations evaluate the (aerodynamic) feasibility of each grid point using tolerances on the constraints and move limits on the objective function, eliminating grossly infeasible points, and generating an approximation domain. The simple structural analyses use the aerodynamic shape and mission variables in basic weight equations to calculate approximate weights needed by the objective function and constraints, further refining the approximation domain.

Using the relatively abundant data from the simple analyses, regression analysis and analysis of variance are used to identify less important terms in the polynomial response surface models. Once the less important terms are eliminated, the structure of the reduced-term polynomial regression models is known, and can be used later in the generation of response surface approximations of the optimal weight and necessary aerodynamic quantities over the approximation domain.

A genetic algorithm (GA; cf. ► Genetic Algorithms) is used to find sets of approximate D-optimal design points in the approximation domain obtained from the parallel simple analyses. The structure of a response surface model is embodied in the regression matrix $\mathbf{X}$, which defines the GA merit function $|\mathbf{X}^{\mathsf{T}}\mathbf{X}|$ (maximized by a set of points called D-optimal). These D-optimal design points are input to the detailed aerodynamic analysis code, which performs detailed analyses at each of the D-optimal design points in parallel. The analyses result in accurate aerodynamic quantities, such as wave drag and other drag components, and accurate aerodynamic loads.

The accurate aerodynamic quantities are used to generate reduced-term polynomial response surface models for each of the expensive quantities (such as wave drag). An aerodynamic load calculated in the detailed aerodynamic analyses is used in a detailed structural optimization to calculate an accurate optimal weight for that particular aerodynamic load. This structural optimization is done (in parallel) for each aerodynamic load generated in the detailed aerodynamic analyses. The accurate optimal weights calculated in the structural optimization are used to generate a reduced-term polynomial response surface model for the optimal weight.

All the response surface models are then used in a configuration optimization to generate an approximate optimal design, which will be used as the starting design for the next iteration of the MDO loop. The grid spacing may possibly be refined for the simple analyses. When some convergence criterion is satisfied, the MDO loop exits with an optimal design.

Note that the *source of parallelism in the present MDO paradigm is the multipoint approximations within each discipline*, where the disciplines are visited sequentially in a pipeline. This contrasts sharply with CSSO MDO paradigms, where the *source of the parallelism is processing the disciplines in parallel.*

See also

- ▶ Bilevel Programming: Applications in Engineering
- ▶ Design Optimization in Computational Fluid Dynamics
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Multilevel Methods for Optimal Design
- ▶ Optimal Design of Composite Structures
- ▶ Optimal Design in Nonlinear Optics
- ▶ Structural Optimization: History

References

1. Alexandrov N (1996) Robustness properties of a trust region framework for managing approximations in engineering optimization. Proc. 6th AIAA/NASA/USAF Multidisciplinary Analysis and Optimization Symposium, 96-4102 AIAA, 1056–1059
2. Alexandrov N, Hussaini MY (eds) (1997) Multidisciplinary design optimization, state-of-the-art. SIAM, Philadelphia
3. Burgee S, Giunta AA, Balabanov V, Grossman B, Mason WH, Narducci R, Haftka RT, Watson LT (1996) A coarse grained parallel variable-complexity multidisciplinary optimization paradigm. Internat J Supercomputer Appl High Performance Comput 10:269–299
4. Cramer EJ, Dennis JE Jr, Frank PD, Lewis RM, Shubin GR (1994) Problem formulation for multidisciplinary optimization. SIAM J Optim 4:754–776
5. Renaud JE, Gabriele GA (1991) Sequential global approximation in non-hierarchic system decomposition and optimization. Adv Design Automation 1:191–200
6. Rodríguez JF, Renaud JE, Watson LT (1998) Convergence of trust region augmented Lagrangian methods using variable fidelity approximation data. Structural Optim 15:141–156
7. Sacks J, Welch WJ, Mitchell TJ, Wynn HP (1989) Design and analysis of computer experiments. Statistical Sci 4:409–435
8. Shankar J, Ribbens CJ, Haftka RT, Watson LT (1993) Computational study of a nonhierarchical decomposition algorithm. Comput Optim Appl 2:273–293
9. Sobieszczanski-Sobieski J (1988) Optimization by decomposition: A step from hierarchic to non-hierarchic systems. Second NASA/Air Force Symposium on Recent Advances in Multidisciplinary Analysis and Optimization

Multifacility and Restricted Location Problems
MFR

Horst W. Hamacher, Stefan Nickel
Fachbereich Math., Universität Kaiserslautern,
Kaiserslautern, Germany

MSC2000: 90B85

Article Outline

Keywords

Location theory; Weber problem; Weber–Rawls problem; multifacility Weber problem; multiWeber problem; Gauge; Convex polytope; Linear programming; NP-hard; Voronoi diagram; Restricted location problem; Finite dominating set; Discretization; Barrier location problems

In location planning one is typically concerned with finding a good location for one or several new facilities with respect to a given set of existing facilities (clients). The two most common models in planar location theory are the *Weber problem*, where the average (weighted) distance of the new to the existing facilities is taken into account and the *Weber–Rawls problem*, where the maximum (weighted) distance of the new to the existing facilities is taken into account.

More precisely, one is given a finite set $Ex = \{Ex_1, \dots, Ex_M\}$ of existing facilities (represented by their geographical coordinates) in the plane $\mathbf{R}^2$ and distance functions d_m assigned to each existing facility $m \in \mathcal{M} := \{1, \dots, M\}$. The set of locations for the N new facilities one is looking for is denoted $X = \{X_1, \dots, X_N\}$. The distance between the new facilities is measured by a common distance d. Additionally, a value w_{mn} is assigned to each pair (Ex_m, X_n), for $m \in \mathcal{M}$, $n \in \mathcal{N} := \{1, \dots, N\}$ and a value v_{rs} assigned to each pair (X_r, X_s), for $r, s \in \mathcal{N}$, $s > r$, reflecting the *level of interaction*.

With these definitions the *multifacility Weber objective function* can be written as

$$\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} w_{mn} d_m(Ex_m, X_n) + \sum_{\substack{r,s\in\mathcal{N}\\ s>r}} v_{rs} d(X_r, X_s) := f(X_1, \dots, X_N)$$

and the *multifacility Weber–Rawls objective function* can be written as

$$\max\left\{\max_{\substack{m\in\mathcal{M}\\ n\in\mathcal{N}}} w_{mn} d_m(Ex_m, X_n),\ \max_{\substack{r,s\in\mathcal{N}\\ s>r}} v_{rs} d(X_r, X_s)\right\} := g(X_1, \dots, X_N).$$

In the corresponding optimization problems we may additionally assume a feasible region $\mathcal{F}$ and we look for

$$\min_{\{X_1,\dots,X_N\}\subset\mathcal{F}} f(X_1, \dots, X_N),$$

and

$$\min_{\{X_1,\dots,X_N\}\subset\mathcal{F}} g(X_1, \dots, X_N).$$

In the first part of this survey it is assumed that $\mathcal{F} = \mathbf{R}^2$ whereas $\mathcal{F}$ will be a restricted set later on.

The models above implicitly assume that the new facilities can be distinguished, that the amount of interaction between each new and existing facility is known and that the new facilities have mutual communication. Note, that problems without communication between the new facilities can be separated into N independent 1-facility problems which can be easily solved by suitable algorithms. Also, in many applications we want to locate a number of indistinguishable facilities to serve the overall demand. This implies that we are not only locating facilities, but we are also allocating existing facilities (clients) to the new ones. This variation of the problem is called *multiWeber* or *multiWeber–Rawls problem* and the objective functions can be written as

$$\sum_{m\in\mathcal{M}} w_m d_m(Ex_m, \{X_1, \dots, X_N\}) = \widehat{f}(X)$$

and

$$\max_{m\in\mathcal{M}} \{w_m d_m(Ex_m, \{X_1, \dots, X_N\})\} = \widehat{g}(X),$$

respectively, where $d_m(Ex_m, \{X_1, \dots, X_N\}) := \min_{Y\in X_1,\dots,X_n\}} d_m(Ex_m, Y)$.

In order to discuss solution methods, suitable types of distance functions d_m, $m \in \mathcal{M}$, are specified next.

Let B be a compact convex set in the plane containing the origin in its interior and let Y be a point in the plane. The *gauge* of Y (with respect to B) is then defined as

$$\gamma_B(Y) := \inf\{\lambda > 0\colon\ Y \in \lambda B\}\,.$$

This definition dates back to [25]. The distance from Ex_m to Y induced by γ_B is

$$d_m(Ex_m, Y) := \gamma_{B_m}(Y - Ex_m) \quad \text{for } m \in \mathcal{M}.$$

In the case where all B_m are convex polytopes with extreme points $\text{Ext}(B_m) := \{e_1^m, \dots, e_G^m\}$ we can define halflines l_i^m starting at Ex_m and going through e_i^m. For the 1-facility case it was proved in [6] for the Weber problem that there always exists an optimal solution in the set of intersection points of the halflines l_i^m for $i = 1, \dots, G^m$ and $m \in \mathcal{M}$. This result carries over to multi-(facility) Weber problems when each B_m has no more than 4 extreme points [24]. For more than 4 extreme points it is in general wrong (see [24] for a counterexample).

In the case where all B_m are polytopes we can give linear programming formulations for the multifacility

Weber as well as the multifacility Weber–Rawls problem [34] using B_m^0, the polar set of B_m, $m \in \mathcal{M}$.

$$\begin{cases} \min & \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}} w_{mn} z_{mn} + \sum_{\substack{r,s\in\mathcal{N} \\ s>r}} v_{rs} z'_{rs} \\ \text{s.t.} & \left\langle Ex_m - X_n, e_m^0 \right\rangle \le z_{mn}, \\ & \quad \forall m \in \mathcal{M},\ n \in \mathcal{N} e_m^0 \in \operatorname{Ext}(B_m^0), \\ & \left\langle X_s - X_r, e^0 \right\rangle \le z'_{rs}, \\ & \quad \forall s, r \in \mathcal{N},\ s > r,\ e^0 \in \operatorname{Ext}(B^0), \end{cases}$$

$$\begin{cases} \min & z \\ \text{s.t.} & w_{mn} \left\langle Ex_m - X_n, e_m^0 \right\rangle \le z, \\ & \quad \forall m \in \mathcal{M},\ n \in \mathcal{N},\ e_m^0 \in \operatorname{Ext}(B_m^0), \\ & v_{rs} \left\langle X_s - X_r, e^0 \right\rangle \le z, \\ & \quad \forall s, r \in \mathcal{N},\ s > r,\ e^0 \in \operatorname{Ext}(B^0). \end{cases}$$

Even without polyhedral structure we still have a convex optimization problem for which several solution techniques are available (see [11,12,21,32] and references therein).

In the case where we also have to deal with the allocation problem we still can apply discretization results from the 1-facility case. The allocation part makes the problem however *NP*-hard (see [22,23]; cf. also ► Complexity Theory; ► Complexity Classes in Optimization). Nevertheless, constructs from computational geometry (e. g. Voronoi diagrams; cf. also ► Voronoi Diagrams in Facility Location) can be used to tackle the allocation part efficiently and allow iterative heuristics producing in general satisfactory results (see [2,30]).

Further extensions are possible and already investigated including location with attraction and repulsion, hub location, etc. (see [32] for further references).

A problem common to all forms of multi-(facility) location problems is, that in an optimal solution locations of different new facilities may coincide with each other or with existing facilities. This raises at least two issues:

- A priori detection of coincidences which result in a reduction of the dimension of the problem and allow the exploitation of differentiability are discussed in [7,20,31].
- If coincidence is excluded, the theory of restricted location can be used which is discussed next.

So far, the set $\mathcal{F}$ for placing new facilities was the whole plane $\mathbf{R}^2$. Now, the feasibility set $\mathcal{F} = \mathbf{R}^2 \setminus \operatorname{int}(\mathcal{R})$ is considered, where $\mathcal{R} \subseteq \mathbf{R}^2$ is the restricting set assumed to be connected in $\mathbf{R}^2$. This problem is more complicated than the unrestricted one, since $\mathcal{F}$ is in general not convex. But from a practical point of view it is a necessary extension of the classical location model, since forbidden regions appear everywhere: nature reserves, lakes, exclusion of coincidence in multifacility, etc. These problems are called *restricted location problems* and have been developed in [1,12,14,15] and [26]. In the following we exclude the trivial case and assume that none of the optimal solutions of the unrestricted problem is a feasible solution of the restricted one.

If the objective function h of the location problem is convex it can be shown that optimal solutions of the restricted problem can be found on the boundary of $\mathcal{R}$. Therefore, level curves

$$L_=(z) := \{X \in \mathbb{R}^n\colon\ h(X) = z\}$$

and level sets

$$L_\le(z) := \{X \in \mathbb{R}^n\colon\ h(X) \le z\}$$

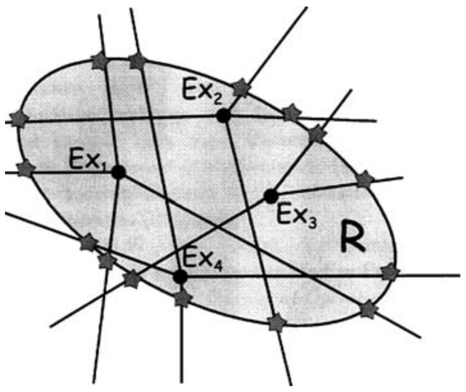
can be used to reformulate the restricted location problem as

$$\min\{z\colon\ L_=(z) \cap \partial\mathcal{R} \ne \emptyset \text{ and } L_\le(z) \subseteq \mathcal{R}\}\,.$$

A resulting search algorithm was formulated in [11], but proved to be inefficient in practical applications.

An efficient approach originally presented in [12,14,15] identifies *finite dominating set* (FDS) on the boundary $\mathcal{R}$, i. e. a finite set of locations on $\partial\mathcal{R}$ which contains an optimal solution. Using this discretization, problems with gauge distance and convex forbidden region can be solved by considering as FDS the intersection points of l_i^m and the boundary of $\mathcal{R}$ (see [15,26,28] and the illustration in the following figure).

The discretization also works for restricted center problems [16] and can be extended to nonconvex forbidden regions (see [15,26]) and also to the case of attraction and repulsion (negative weights are allowed), see [29]. The concept of forbidden regions has been successfully applied to a problem in PCB assembly, where the bins holding the parts to be inserted into the PCB have to be stored [10]. Of course, the PCB itself has to be forbidden for placing a bin. A solution approach, where also the issue of space requirements in a multifacility setting is addressed can be found in [9,15]. A more gen-



Multifacility and Restricted Location Problems, Figure 1
Example of a restricted location problem with 4 existing facilities and an elliptic forbidden region

eral case where the new facility is a line has been considered in [33]. Algorithms for multifacility problems with forbidden regions can be found in [8,15,27].

Another type of restricted location problem is one, where not only placement, but also tresspassing of regions is forbidden. These problems are called *barrier location problems*. The corresponding models are mathematically challenging, since the distance functions (and thus also the objective functions) are no longer convex. [17] considers Euclidean distances and one circle as forbidden region. [1] and [4] develop heuristics for l_p distances and barriers that are closed polygons. [19] and [3] obtain discretization results for l_1 distances and arbitrary shaped barriers by showing an equivalence of the barrier problem to a network location problem. In the more general context of gauge distances an FDS is given in [13] for median problems and in [5] for center problems. Finally, [18] considers barrier problems if the distance is an arbitrary norm and the barrier consists of a line with finitely many passages.

See also

- ► Combinatorial Optimization Algorithms in Resource Allocation Problems
- ► Competitive Facility Location
- ► Facility Location with Externalities
- ► Facility Location Problems with Spatial Interaction
- ► Facility Location with Staircase Costs
- ► Global Optimization in Weber's Problem with Attraction and Repulsion
- ► MINLP: Application in Facility Location-allocation
- ► Network Location: Covering Problems
- ► Optimizing Facility Location with Rectilinear Distances
- ► Production-distribution System Design Problem
- ► Resource Allocation for Epidemic Control
- ► Single Facility Location: Circle Covering Problem
- ► Single Facility Location: Multi-objective Euclidean Distance Location
- ► Single Facility Location: Multi-objective Rectilinear Distance Location
- ► Stochastic Transportation and Location Problems
- ► Voronoi Diagrams in Facility Location
- ► Warehouse Location Problem

References

1. Aneja YP, Parlar M (1994) Algorithms for Weber facility location in the presence of forbidden regions and/or barriers to travel. Transport Sci 28:70–76
2. Aurenhammer F (1991) Voronoi diagrams – A survey of a fundamental geometric data structure. ACM Computing Surveys 23:345–405
3. Batta R, Ghose A, Palekar US (1989) Locating facilities on the Manhattan metric with arbitrarily shaped barriers and convex forbidden regions. Transport Sci 23:26–36
4. Butt SE, Cavalier TM (1996) An efficient algorithm for facility location in the presence of forbidden regions. Europ J Oper Res 90:56–70
5. Dearing PM, Hamacher HW, Klamroth K (1998) Center problems with barriers. Techn Report Depts Math Univ Kaiserslautern and Clemson Univ
6. Durier R, Michelot C (1985) Geometrical properties of the Fermat–Weber problem. Europ J Oper Res 20:332–343
7. Fliege J (1997) Nondifferentiability detection and dimensionality reduction in minisum multifacility location problems. J Optim Th Appl 94
8. Fliege J, Nickel S (2000) An interior point method for multifacility location problems with forbidden regions. Studies in Location Anal 14:23–45
9. Foulds LR, Hamacher HW (1993) Optimal bin location and sequencing in printed circuit board assembly. Europ J Oper Res 66:279–290
10. Francis RL, Hamacher HW, Lee C-Y, Yeralan S (1994) Finding placement sequences and bin locations for Cartesian robots. Trans Inst Industr Eng (IIE):47–59
11. Francis RL, McGinnis LF Jr, White JA (1992) Facility layout and location: An analytical approach, 2nd edn. Prentice-Hall, Englewood Cliffs
12. Hamacher HW (1995) Mathematische Lösungsverfahren für planare Standortprobleme. Vieweg, Braunschweig, Wiesbaden
13. Hamacher HW, Klamroth K (1997 2000) Planar location problems with barriers under polyhedral gauges. Report in Wirtschaftsmath 21, Dept Math, Univ Kaiserslautern

14. Hamacher HW, Nickel S (1994) Combinatorial algorithms for some 1-facility median problems in the plane. Europ J Oper Res 79:340–351
15. Hamacher HW, Nickel S (1995) Restricted planar location problems and applications. Naval Res Logist 42:967–992
16. Hamacher HW, Schöbel A (1997) A note on center problems with forbidden polyhedra. Oper Res Lett 20:165–169
17. Katz IN, Cooper L (1981) Facility location in the presence of forbidden regions, I: Formulation and the case of Euclidean distance with one forbidden circle. Europ J Oper Res 6:166–173
18. Klamroth K (1996) Planar location problems with line barriers. Report in Wirtschaftsmath 13, Dept Math, Univ Kaiserslautern
19. Larson RC, Sadiq G (1983) Facility locations with the Manhattan metric in the presence of barriers to travel. Oper Res 31:652–669
20. Lefebvre O, Michelot C, Plastria F (1991) Sufficient conditions for coincidence in minisum multifacility location problems with a general metric. Oper Res 39:437–442
21. Love RF, Morris JG, Wesolowsky GO (1988) Facilities location: Models and methods. North-Holland, Amsterdam
22. Masuyama S, Ibaraki T, Hasegawa T (1981) The computational complexity of the M-center problems on the plane. Trans IECE Japan E64:57–64
23. Megiddo N, Supowit KJ (1984) On the complexity of some common geometric location problems. SIAM J Comput 13:182–196
24. Michelot C (1987) Localization in multifacility location theory. Europ J Oper Res 31:177–184
25. Minkowski H (1967) Gesammelte Abhandlungen, vol 2. Chelsea, New York
26. Nickel S (1995) Discretization of planar location problems. Shaker, Aachen
27. Nickel S (1997) Bicriteria and restricted 2-facility Weber problems. Math Meth Oper Res 45(2):167–195
28. Nickel S (1998) Restricted center problems under polyhedral gauges. Europ J Oper Res 104(2):343–357
29. Nickel S, Dudenhöffer E-M (1997) Weber's problem with attraction and repulsion under polyhedral gauges. J Global Optim 11:409–432
30. Okabe A, Boots B, Sugihara K (1992) Spatial tesselations. Concepts and applications of Voronoi diagrams. Wiley, New York
31. Plastria F (1992) When facilities coincide: exact optimality conditions in multifacility Location. J Math Anal Appl 169:476–498
32. Plastria F (1995) Continuous location problems. In: Drezner Z (ed) Facility Location – A Survey of Applications and Methods. Springer, Berlin, pp 225–262
33. Schöbel A (1999) Locating lines and hyperplanes: Theory and algorithms. Kluwer, Dordrecht
34. Ward JE, Wendell RE (1985) Using block norms for location modeling. Oper Res 33:1074–1090

Multi-index Transportation Problems

MITP

MAURICE QUEYRANNE[1], FRITS SPIEKSMA[2]
[1] University British Columbia, Vancouver, Canada
[2] Maastricht University, Maastricht, The Netherlands

MSC2000: 90C35

Article Outline

Keywords

Transportation problem; Three-dimensional transportation problem; Greedy algorithm; Monge property; Approximation algorithms

An ordinary *transportation problem* has variables with two indices, typically corresponding to sources (or origins, or supply points) and destinations (or demand points). A *multi-index transportation problem* (MITP) has variables with three or more indices, corresponding to as many different types of points or resources or other factors. Multi-index transportation problems were considered by T. Motzkin [22] in 1952; an application involving the distribution of different types of soap was presented by E. Schell [35] in 1955. MITPs are also known as *multidimensional transportation problems* [4]. There are several versions and special cases of MITPs:

- The number k of dimensions may be fixed to a small value; the resulting MITP is called a *k-index transportation problem*, k ITP. Quite naturally, the best studied cases are the *three-index transportation problems* (3ITPs), also known as *three-dimensional*, or *3D transportation problems*.

- The type of constraints is determined by an integer m with $0 < m < k$, defining m-fold k ITPs (called *symmetric MITPs* in [16]; see also [41, Chapt. 8]). The most common cases are *axial MITPs*, when $m = k-1$; and *planar MITPs*, when $m = 1$; see below for details.
- Integer solutions may or may not be required. Integrality requirements, which give rise to *integer MITPs*, may be necessary since MITPs lack the integrality property enjoyed by ordinary transportation problems (but see [22] for an exception).
- Unit right-hand sides, in conjunction with integrality requirements, give rise to *multi-index assignment problems* (MIAPs). (Some authors use this term for integer MITPs with integer right-hand sides; the present terminology, consistent with that for ordinary assignment and transportation problems, seems preferable.) MIAPs are hard to solve: the 3IAP is already *NP*-hard by reduction from the 3-dimensional matching problem [17]. Even worse [6]: no polynomial time algorithm for the 3IAP can achieve a constant performance ratio, unless $P = NP$.
- The objective function is usually a simple linear combination of the variables, normally a total cost to be minimized as in equation (1) below. Alternatives, not considered in this article, may include bottleneck objectives [11,36], more general nonlinear objectives such as in [34], or multicriteria problems [38].
- There may be additional constraints, such as upper bounds on the variables, (capacitated MITPs), variables fixed to the value zero (MITPs with forbidden cells), or constraints on certain partial sums of variables (MITPs with generalized capacity constraints).

MITPs with linear objectives and without integrality restrictions are linear programming problems with a special structure. The most extensively studied *integer MITPs* are *three-index assignment problems* (3IAPs); see also Three-index Assignment Problem.

Formulations

The following compact notation [31,34] avoids multiple summations and multiple layers of subindices. Let $k \geq 3$ denote the number of dimensions or indices, and $K = \{1, \dots, k\}$. For $i \in K$ let A_i denote the set of values of the ith index. Let $A = \otimes_{i \in K} A_i = A_1 \times \cdots \times A_k$ denote the Cartesian product of these index sets, that is, the set of all joint indices (k-tuples) $a = (a(1), \dots, a(k))$ with $a(i) \in A_i$ for all $i \in K$. One variable x_a is associated with each joint index $a \in A$. Thus, for example in a 3ITP with index sets I, J and L, the variable x_a stands for $x_{ij\ell}$ when the joint index is $a = (i, j, \ell)$.

Given unit costs $c_a \in \mathbf{R}$ for all $a \in A$, a linear objective function is

$$\min \sum_{a \in A} c_a x_a \tag{1}$$

and the variables are usually restricted to be nonnegative:

$$x_a \geq 0 \quad \text{for all } a \in A. \tag{2}$$

Given the integer m with $0 < m < k$, the demand constraints of the m-fold k ITP are defined as follows. Let $\binom{K}{k-m}$ denote the set of all $(k-m)$-element subsets of K; an $F \in \binom{K}{k-m}$ is interpreted as a set of $k-m$ 'fixed indices'. Given such an F and a $(k-m)$-tuple $g \in A_F = \otimes_{f \in F} A_f$ of 'fixed values', let

$$A(F, g) = \{a \in A \colon\ a(f) = g(f),\ \forall f \in F\}$$

be the set of k-tuples which coincide with g on the fixed indices. The m-fold demand constraints are

$$\begin{gathered}\sum_{a \in A(F,g)} x_a = d_{Fg} \\ \text{for all } F \in \binom{K}{k-m},\ g \in A_F,\end{gathered} \tag{3}$$

where the right-hand sides d_{Fg} are given positive demands associated with the values g for fixed index subset F. These 'demands' may also denote supplies or capacities when the indices represent sources or some other resource type. When some of these resources are in excess, the equality in constraints (3) may be replaced with inequalities. Problem (1)–(3) is a k ITP. Adding the integrality restrictions

$$x_a \in \mathbb{N} \quad \text{for all } a \in A, \tag{4}$$

yields an integer MITP.

As mentioned above, the most common cases are $m = k-1$, defining axial MITPs; and $m = 1$, defining planar MITPs. For the axial problems, the notation may simplified by letting $d_{ig} = d_{Fg}$ when $F = \{ i \}$. Note that each variable x_a appears in the same number k of axial and planar demand constraints; however there are only $\sum_{i \in K} |A_i|$ axial constraints, versus $\sum_{i \in K} \prod_{f \in K \setminus \{ i \}} |A_f|$ planar constraints. Of course, it is possible to combine demand constraints with different values of m, so as to formulate different types of restrictions (e. g., see [5] and [16]).

Reductions between MITPs are presented in [16], where it is shown in particular that an m-fold k ITP can be reduced to a 1-fold k ITP for any m (with $0 < m < k$), thereby generalizing a result in [14]. Thus, an algorithm that solves planar k ITPs is in principle capable of solving m-fold k ITPs for any m (with $0 < m < k$).

Notice that any MITP with arbitrary right hand sides can be transformed to a MITP with right hand sides 1. This is a (pseudopolynomial) transformation and simply involves duplicating a resource with a supply of q units by q unit-supply resources. There seems to be little advantage in doing so, except perhaps in converting an integer MITP into one with 0–1 variables.

Another issue is the existence of feasible solutions. For an axial MITP the requirement of equal total demands $\sum_g d_{ig} = \sum_g d_{jg}$ for all $i, j \in K$ is a necessary and sufficient condition for the existence of feasible solutions. Feasibility conditions are more complicated for nonaxial problems; see [40] for a review of results for planar problems. See also [41, Chapt. 8] for properties of polytopes associated with (integer) MITPs, including issues of degeneracy.

Applications

Transportation and Logistics

MITPs are used to model transportation problems that may involve different goods; such resources as vehicles, crews, specialized equipment; and other factors such as alternative routes or transshipment points. Thus index sets A_1 and A_2 may represent destinations and sources, respectively, and the other sets $A_3, A_4, \ldots$ these additional factors. The type of 'demand' constraints used will reflect the availability of these factors and their interactions. Thus, for example, an axial demand constraint (3) with right-hand side d_{3i} will be used for a vehicle type $i \in A_3$ of which d_{3i} units are globally available (at identical cost) to all sources and destinations, while a constraint with $F = \{2, 3\}$ will be used if there are d_{Fg} vehicles of type $g(3)$ available at the different sources $g(2)$.

Interesting cases arise when each resource or factor $\ell \in A_i$ corresponds to a point $P_{i,\ell}$ in a *metric space*, i. e., a set with a distance δ, and the unit costs c_a are 'decomposable' as defined below. Each joint index $a \in A$ may be interpreted as a *cluster* of points among which transportation and other activities are conducted. The unit cost c_a reflects the within-cluster transportation costs associated with these activities; it is *decomposable* if it can be expressed as a function of the distances between pairs of points in the cluster a. Examples include the *diameter* $\max_{i,j} \delta(P_{i,a(i)}, P_{j,a(j)})$, when all these activities are performed simultaneously; the sum costs $\sum_{i,j} \delta(P_{i,a(i)}, P_{j,a(j)})$ when all activities are performed sequentially; and the *Hamiltonian path* or *path* costs, when all points $P_{i\,\ell}$ in the cluster have to be visited in a shortest sequence.

Other interesting cases arise when one of the indices denotes time. A simple *dynamic location problem* [27] may be modeled as an axial k ITP, where index set A_1 may denote the set of facilities (say, warehouses) to be located; A_2 that of candidate locations; and A_3 that of time periods. The costs c_{ijt} may include discounted construction and operating costs of these facilities. See [38] and [33] for other applications of this type.

Timetabling

Other problems involving time and which can be formulated as MITPs arise in timetabling or staffing applications. To illustrate, consider the following generic situation. Given are N employees (index i), each of which can be assigned to one of M tasks (index j) during each of T time periods (index k). Moreover, for each pair consisting of a task and a time period a number r_{jk} is given denoting the number of employees required for task j in period k. Also, a number r_{ij} is given denoting the number of periods that task j requires employee i. An employee can only be assigned to one task during each time period. Finally, there is a cost-coefficient c_{ijk} which gives the cost of employee i performing task j in period k. This problem is called the *multiperiod assignment problem* in [21] (see also the references contained therein). To model this as a planar 3ITP, let A_1 be the set of employees; A_2 the set of tasks; A_3 the set of time periods;

$$d_{Fg} = \begin{cases} r_{jk} & \text{for } F = \{2,3\}, \quad \forall g = (j,k); \\ 1 & \text{for } F = \{1,3\}, \quad \forall g = (i,k); \\ r_{ij} & \text{for } F = \{1,2\}, \quad \forall g = (i,j); \end{cases}$$

and require the decision variables to be in $\{0,1\}$. A special case arises when $r_{jk} = 1$ for all j, k and $N = M$. The polyhedral structure of the resulting planar 3ITP is investigated in [7]. Other references dealing with timetabling problems formulated as MITPs are [10,15] and [12].

Multitarget Tracking

Consider the following (idealized) situation. N objects move along straight lines in the plane. At each of T time instants a *scan* has been made, and the approximate position of each object is observed and recorded. From such a scan it is not possible to deduce which object generated which observation. Also, a small error may be associated with each observation. A *track* is defined as a T-tuple of observations, one from each scan. For each possible track a cost is computed based on a least squares criterion associated with the observations in the track. The problem is now to identify N tracks while minimizing the sum of the costs of these tracks. This problem is called the *data-association problem* in [25]. It can be modeled as an axial integer TIAP as follows: let A_i be the set of observations in scan i, $i = 1, \dots, T$, and let $d_{ig} = 1$, $i = 1, \dots, T$, $g = 1, \dots, N$. Not surprisingly, this problem is *NP*-hard already for $T = 3$ (see [37]; notice however that this does not follow from the *NP*-hardness of 3IAP due to the structure present in the cost-coefficients in the objective function of multitarget tracking problems). Other references dealing with target tracking problems formulated as axial MIAPs are [23] and [24]; see also [20].

Tables with Given Marginals

Other statistical applications of MITPs require finding multidimensional tables with given sums across rows or higher-dimensional planes, as specified in constraints (3). The right-hand sides d_{Fg} of such constraints are often known as *marginals*. In a simple application [3] arising in the *integration of surveys* and *controlled selection*, each index set represents a population from which a sample is to be drawn. A (joint) sample is a k-tuple, one from each population. The marginals are specified marginal probability distributions over each population, giving rise to axial demand constraints. Given sample costs c_a, the problem is to find a joint probability distribution, defined by (x_a), of all the samples, consistent with these marginal distributions and of minimum expected cost (1).

In contrast, problems of *updating input-output matrices* (see [34] and references therein) typically have nonlinear objectives. In such problems, given are a k-dimensional array B of data (for example, past input-output coefficients) and arrays d of marginals (for example, forecast aggregate coefficients) with appropriate dimensions. The problem is to determine values x_a, the updated array entries, satisfying the demand constraints corresponding to the given marginals, and such that the resulting updated array $X = (x_a)$ differs as little as possible from the given array B, as specified by an appropriate (nonlinear) objective function. A (nonlinear) MITP arises when the values x_a are constrained to be nonnegative, a natural requirement in many contexts.

Other Applications

include an axial integer 3ITP model for planning the launching of weather satellites [27], and an axial integer 5IAP arising in routing meshes in circuit design [9].

Solution Methods

As noted above, MITPs are linear programming problems with a special structure. There are several proposals for extensions of LP (transportation) algorithms to MITPs (e. g., [4,13] for 3ITPs and [1] for a 4ITP).

As also mentioned earlier, integer MITPS are hard to solve. Exact algorithms have been proposed for the axial integer 3IAP (see Three-index Assignment Problem) and for the planar integer 3IAP (see [39] and [19]). Other exact approaches for integer MITPs rely on structure that is present in the particular application considered (see, e. g., [12]).

Several methods have been proposed to obtain good approximate solutions to integer MITPs. In [21]results are reported for a *rounding heuristic* on some medium-sized planar integer 3ITPs. A *tabu search* algorithm for this problem is described in [18]. Heuristic solution approaches based on *Lagrangian relaxation* are proposed in [26,28] and [29] for multitarget tracking problems.

One major difficulty with these exact or approximate solution methods may be the sheer size of MITP formulations; if, for example, all $|A_i| = n$ then an m-fold k ITP has n^k variables and $\binom{k}{m} n^{k-m}$ constraints. In contrast, the two approaches sketched below yield feasible solutions to *axial* MITPs much more quickly than simply writing down all the cost coefficients. In particular, these algorithms only produce the nonzero variables x_a and their values; all other variables are zero in the solution. In addition, this solution is integral if all demands are integral. Of course, the effectiveness of these methods relies on some assumptions on the cost coefficients c_a, assumptions which are verified in several applications.

A Greedy Algorithm for Axial MITPs

The *greedy algorithm* below (a multi-index extension of the *North–West corner rule*) finds a feasible solution to axial MITPs in $O(k \sum_i |A_i|)$ time, which is (for fixed k) linear in the size of the demand data d_{ig}. This solution is in fact optimal if the cost coefficients are known to satisfy a 'Monge property' [3,31,32] defined below. (For $k = 3$, this greedy algorithm is already described in [4] to obtain a basic feasible solution).

Consider the axial k ITP with equality constraints (3) and assume that each $A_i = \{1, \dots, |A_i|\}$. Recalling that the demands are denoted d_{ig}, assume that $\sum_{g \in A_i} d_{ig} = \sum_{g \in A_1} d_{1g}$ for all $i \in K$, a necessary and sufficient condition for the problem to be feasible.

```
PROCEDURE greedy MITP algorithm
    WHILE ($\sum_{g\in A_i} d_{ig} > 0$ for all $i \in K$) DO
        let $a(i) = \min\{g \in A_i : d_{ig} > 0\}$;
        let $\Delta = \min\{d_{i,a(i)} : i \in K\}$;
        let $x_a = \Delta$;
        FOR $i \in K$ DO let $d_{i,a(i)} = d_{i,a(i)} - \Delta$;
RETURN $x$
END
```

A greedy algorithm for axial MITPs

A Monge Property

The *join* $a \vee b$ and *meet* $a \wedge b$ of $a, b \in A$ are

$$(a \vee b)_i = \max\{a(i), b(i)\},$$
$$(a \wedge b)_i = \min\{a(i), b(i)\} \quad \text{for all } i \in K.$$

The cost coefficients (c_a) satisfy the *Monge property* if

$$c_{a \vee b} + c_{a \wedge b} \leq c_a + c_b \quad \text{for all } a, b \in A.$$

Note that this is just the *submodularity* of the function $c: A \to \mathbf{R}$ defined on the product lattice A, see [3,31,32]. These references show that the above greedy algorithm returns an optimal solution for *all* feasible demands if and only if the cost function satisfies the Monge property. The latter two references also extend the greedy algorithm

i) to the case of forbidden cells when the nonforbidden cells form a *sublattice* of A; and
ii) so that it returns an optimal dual solution.

They also show that optimizing a linear function over a *submodular polyhedron* is special case of the dual problem. It is shown in [32] that the primal problems are equivalent to the 'submodular linear programs on forests' of [8].

Cost functions c with the Monge property include typical decomposable costs (as defined above) when all the points are located on a same line or on parallel lines (one line for each factor type A_i). For these problems, the greedy algorithm above amounts to a 'left to right sweep' across the points.

Hub Heuristics for Axial MITPs

The basic idea ([30], extending earlier work on axial 3IAPS [6] and MIAPs [2] with decomposable costs) is to solve a small number of ordinary transportation problems and to expand their solutions into a feasible solution to the original MITP. For a large collection of decomposable costs arising from applications, the objective value of this feasible solution is provably within a constant factor of the optimum.

Given an index h, called the *hub*, determine, for each index $i \neq h$, a feasible solution to the ordinary transportation problem defined by supplies $(d_{ij})_{j \in A(i)}$ and $(d_{hg})_{g \in A(h)}$. The Expand procedure below then takes as inputs these solutions $y^{(h)} = (y^i)_{i \neq h}$ and expands them into a feasible solution $x^{(h)}$ to the axial MITP. Its running time is $O(|A_h| \sum_{i \neq h} |A_i|)$.

```
FOR $k = 1, \dots, n$,
PROCEDURE Expand$(h, y^{(h)})$
FOR $g := 1$ TO $n_h$ DO
    $q := 0$;
    $a(i) := 1$ for $i \in K \setminus h$;
    WHILE$(q < d_{h,g})$ DO
        let $\ell$ be such that
        $y^{\ell}_{a(\ell),g} = \min\{y^{r}_{a(r),g} : r \neq h\}$;
        $x^{(h)}_a := y^{\ell}_{a(\ell),g}$;
        $y^{r}_{a(r),g} := y^{r}_{a(r),g} - x^{(h)}_a$ for all $r \in K \setminus h$;
        $a(\ell) := a(\ell) + 1$;
        $q := q + x^{(h)}_a$;
RETURN $x^{(h)}$
END
```

The Expand procedure for axial MITPs

In the hub heuristics for decomposable costs, the ordinary transportation problems use as cost coefficients the distances $\delta(P_{ij}, P_{hg})$ between the corresponding points P_{ij} and P_{hg} in the metric space. The expanded MITP solution x^h would be optimum if the cost function was that of the star with center h, namely if $c_a = \sum_{i \neq h} \delta(P_{i,a(i)}, P_{h,a(h)})$. The *triangle-inequality* property of the distance δ allows one to bound the cost penalty from using this h-star cost function instead of the actual decomposable cost function.

In the *single hub heuristic*, one chooses a hub $h \in K$; solves these $k-1$ transportation problems; inputs their solutions $y^{(h)}$ to Expand; and simply outputs the resulting MITP solution $x^{(h)}$. If the distance δ satisfies the triangle inequality, the cost of this solution $x^{(h)}$ is no more than $k-1$ times the optimal cost, in the worst case, for many common decomposable cost functions. The *multiple-hub heuristic* is an obvious extension whereby one performs the single-hub heuristic k times, once for each $h \in K$, and retains the best solution. This amounts to solving $\binom{K}{2}$ ordinary transportation problems. Under the same assumptions as above and for many common decomposable cost functions, the cost of the resulting solution is less than twice the optimum cost in the worst case.

See also

► Generalized Assignment Problem
► Stochastic Transportation and Location Problems

References

1. Bammi D (1978) A generalized-indices transportation problem. Naval Res Logist Quart 25:697–710
2. Bandelt H-J, Crama Y, Spieksma FCR (1994) Approximation algorithms for multidimensional assignment problems with decomposable costs. Discrete Appl Math 49:25–50
3. Bein WW, Brucker P, Park JK, Pathak PK (1995) A Monge property for the d-dimensional transportation problem. Discrete Appl Math 58:97–109
4. Corban A (1964) A multidimensional transportation problem. Rev Roumaine Math Pures et Appl IX:721–735
5. Corban A (1966) On a three-dimensional transportation problem. Rev Roumaine Math Pures et Appl XI:57–75
6. Crama Y, Spieksma FCR (1992) Approximation algorithms for three-dimensional assignment problems with triangle inequalities. Europ J Oper Res 60:273–279
7. Euler R, Le Verge H (1996) Time-tables, polyhedra and the greedy algorithm. Discrete Appl Math 65:207–222
8. Faigle U, Kern W (1996) Submodular linear programs on forests. Math Program 72:195–206
9. Fortin D, Tusera A (1994) Routing in meshes using linear assignment. In: Bachem A, Derigs U, Jünger M, Schrader R (eds) Operation Research 93, pp 169–171
10. Frieze AM, Yadegar J (1981) An algorithm for solving 3-dimensional assignment problems with application to scheduling a teaching practice. J Oper Res Soc 32:989–995
11. Geetha S, Vartak MN (1994) The three-dimensional bottleneck assignment problem with capacity constraints. Europ J Oper Res 73:562–568
12. Gilbert KC, Hofstra RB (1987) An algorithm for a class of three-dimensional assignment problems arising in scheduling applications. IIE Trans 8:29–33
13. Haley KB (1962) The solid transportation problem. Oper Res 10:448–463
14. Haley KB (1963) The multi-index problem. Oper Res 11:368–379
15. Junginger W (1972) Zurückführung des Stundenplanproblems auf einen dreidimensionales Transportproblem. Z Oper Res 16:11–25
16. Junginger W (1993) On representatives of multi-index transportation problems. Europ J Oper Res 66:353–371
17. Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of Computer Computations. Plenum, New York, pp 85–103
18. Magos D (1996) Tabu search for the planar three-index assignment problem. J Global Optim 8:35–48
19. Magos D, Miliotis P (1994) An algorithm for the planar three-index assignment problem. Europ J Oper Res 77:141–153
20. Mavridou T, Pardalos PM, Pitsoulis L, Resende MGC (1998) A GRASP for the biquadratic assignment problem. Europ J Oper Res 105/3:613–621

21. Miller JL, Frank LS (1996) A binary-rounding heuristic for multi-period variable-task duration assignment problems. Comput Oper Res 23:819–828
22. Motzkin T (1952) The multi-index transportation problem. Bull Amer Math Soc 58:494
23. Murphey R, Pardalos PM, Pitsoulis L (1998) A GRASP for the multitarget multisensor tracking problem. In: DIMACS, vol 40. Amer Math Soc, Providence, pp 277–302
24. Murphey R, Pardalos PM, Pitsoulis L (1998) A parallel GRASP for the data association multidimensional assignment problem. In: IMA Vol Math Appl, vol 106. Springer, Berlin, pp 159–180
25. Pattipatti KR, Deb S, Bar-Shalom Y, Washburn RB Jr (1990) Passive multisensor data association using a new relaxation algorithm. In: Bar-Shalom Y (ed) Multitarget-multisensor tracking: Advances and applications, p 111
26. Pattipatti KR, Deb S, Bar—Shalom Y, Washburn RB Jr (1992) A new relaxation algorithm passive sensor data association. IEEE Trans Autom Control 37:198–213
27. Pierskalla WP (1968) The multidimensional assignment problem. Oper Res 16:422–431
28. Poore AB (1994) Multidimensional assignment formulation of data-association problems arising from multitarget and multisensor tracking. Comput Optim Appl 3:27–57
29. Poore AB, Rijavec N (1993) A Lagrangian relaxation algorithm for multidimensional assignment problems arising from multitarget tracking. SIAM J Optim 3:544–563
30. Queyranne M, Spieksma FCR (1997) Approximation algorithms for multi-index transportation problems with decomposable costs. Discrete Appl Math 76:239–253
31. Queyranne M, Spieksma FCR, Tardella F (1993) A general class of greedily solvable linear programs. In: Rinaldi G, Wolsey L (eds) Proc Third IPCO Conf (Integer Programming and Combinatorial Optimization), pp 385–399
32. Queyranne M, Spieksma FCR, Tardella F (1998) A general class of greedily solvable linear programs. Math Oper Res 23(4):892–908
33. Rautman CA, Reid RA, Ryder EE (1993) Scheduling the disposal of nuclear waste material in a geologic repository using the transportation model. Oper Res 41:459–469
34. Romero D (1990) Easy transportation-like problems on K-dimensional arrays. J Optim Th Appl 66:137–147
35. Schell E (1955) Distribution of a product by several properties. In: Directorate of Management Analysis (ed) Second Symposium in Linear Programming 2. DCS/Comptroller HQ, US Air Force, Washington DC, pp 615–642
36. Sharma JK, Sharup K (1977) Time-minimizing multidimensional transportation problem. J Eng Production 1:121–129
37. Spieksma FCR, Woeginger GJ (1996) Geometric three-dimensional assignment problems. Europ J Oper Res 91:611–618
38. Tzeng G, Teodorović D, Hwang M (1996) Fuzzy bicriteria multi-index transportation problems for coal allocation planning of Taipower. Europ J Oper Res 95:62–72
39. Vlach M (1967) Branch and bound method for the three-index assignment problem. Ekonomicko–Matematicky Obzor 3:181–191
40. Vlach M (1986) Conditions for the existence of solutions of the three-dimensional planar transportation problem. Discrete Appl Math 13:61–78
41. Yemelichev VA, Kovalev MM, Kratsov MK (1984) Polytopes, graphs and optimization. Cambridge Univ Press, Cambridge

Multilevel Methods for Optimal Design

NATALIA M. ALEXANDROV
NASA Langley Res. Center, Hampton, USA

MSC2000: 49M37, 65K05, 65K10, 90C30, 93A13

Article Outline

Keywords

Nonlinear optimization; Multilevel; Bilevel; Hierarchical; Multidisciplinary design

Multilevel, or *hierarchical*, programming problems (MLP) are constrained optimization programs in which subsets of the solution set are themselves solution sets of other, lower-level optimization programs. Several general MLP problem statements exist. They differ from one another in the specifics of optimization variable distribution among the levels and the definition of the objectives and constraints at particular levels.

Given a set of objectives $\{f_i\}_{i=1,\dots,M}$ with $f_i\colon \mathbf{R}^n \to \mathbf{R}$ and a vector of variables $x \in \mathbf{R}^n$, partitioned into subsets

$x = (x_1, \dots, x_M)$ for some integer M denoting the number of subsystems, a prototypical form of MLP may be stated as follows:

$$\begin{cases} \min_{x_1 \in S_1} & f_1(x) \\ \text{s.t.} & x_2 \in \operatorname{argmin}_{x_2 \in S_2} \{f_2(x)\} \\ & \vdots \\ & x_M \in \operatorname{argmin}_{x_M \in S_M} \{f_M(x)\}, \end{cases}$$

where the optimization problem at each level i controls its own subset of variables x_i, while the other subsets of variables $x_1, \dots, x_{i-1}, x_{i+1}, x_M$ serve as parameters. The constraint set for each level is $S_i \equiv \{x \colon h_i(x) = 0, g_i(x) \geq 0\}$ with $h_i \colon \mathbf{R}^n \to \mathbf{R}^{m_{h_i}}$ and $g_i \colon \mathbf{R}^n \to \mathbf{R}^{m_{g_i}}$ for some integers m^{h_i}, m^{g_i}.

This form of MLP inspired by the work of H. Stackelberg [92] can be viewed as an M-player Stackelberg game [18,84]. Its interpretation is that of M autonomous players or decision makers seeking to minimize their (possibly constrained) objective functions while manipulating subsets of decision or design variables disjoint from those of other decision makers. The higher-level problems are implicit in the variables of the lower-level problems. This formulation has been studied widely in the bilevel case. See, for example, [15] and the references therein. In general, all problem levels, but the outermost one, may contain a number of concurrent optimization problems.

A related variant of the problem, known as the *generalized bilevel programming problem*, represents the reaction of the lower-level problem to decisions made by the upper-level problem via a solution of an equilibrium problem stated as a variational inequality:

$$\begin{cases} \min\limits_{\substack{x \in X, \\ y \in Y(x)}} & f_1(x, y) \\ \text{s.t.} & \langle f_2(x, y), y - z \rangle \leq 0 \quad \text{for all } z \in Y(x), \end{cases}$$

where the upper-level domain X is such that the lower-level domain $Y(x)$ is not empty. This formulation was introduced by P. Marcotte in [63] and studied in [45,64], and [71].

Multilevel problems may be partitioned into two classes with respect to another criterion [100]. In one of the classes, upper-level optimization problems depend on the corresponding lower-level ones through the *optimal value functions* (or the *marginal functions*) of the lower-level problems. An optimal value function represents the value of a lower-level objective function at a solution of that lower-level problem. In the other class, upper-level problems depend on the corresponding lower-level problems through the actual optimal solutions of the latter. An example of two such formulations in engineering design optimization will be given further.

Multilevel programming problems arise in numerous applications where the structure of the application involves hierarchical *decision making* or where the sheer size and complexity of the problem necessitates partitioning of the system and processing the subsystems in a hierarchical fashion. Information on applications of multilevel optimization in such varied areas as power systems, water resource systems, urban traffic systems, and river pollution control can be found in [36,50,51,52,62,69,70,85], and many other references. The use of multilevel algorithms in engineering control is well documented, for instance, in [46] and [57].

The broad area of *multidisciplinary design optimization*(MDO) – a term that denotes a large set of research subjects and practical techniques for the design of complex coupled engineering systems – is particularly amenable to the use of multilevel methods, due to the extreme computational expense and the organizational complexity of the field. For instance, the design of aircraft involves aerodynamics, structural analysis, control, weights, propulsion, and cost, to list a few disciplines. The complexity and expense of each discipline have assured that most disciplines have developed into vast, autonomous fields of study, so that practically feasible optimization methods that involve the contributing disciplines must take into account such an *autonomy* and the hierarchical organization. Maintaining disciplinary autonomy while accounting for interdisciplinary subsystem couplings and allowing for integrated system optimization with respect to system and interdisciplinary objectives is one of the tasks of MDO. Overviews of multidisciplinary optimization may be found in [6] and [90].

Practitioners of engineering have been using multilevel methods, in some form, since optimization algorithms made their appearance in engineering problems. The seminal works [60,65], and [98] contributed to a systematic development and understanding of hierarchical optimization. Multilevel methods have been studied extensively in application to multidisciplinary design ([16,17,22,96,97]) and single-discipline design areas that give rise to large problems, such as structural optimization (e. g., [74,87,91]). Engineering multilevel optimization has always had a strong connection with *multi-objective optimization* (e. g., [53]).

Problem Formulation

The procedure of formulating an engineering design problem as a multilevel or a bilevel problem is difficult and depends on the complexity and size of the problem. The general components in formulating a multilevel optimization problem are as follows:

- The original problem is studied to determine its structure. Structure is of paramount importance in deciding to adopt a particular formulation. For instance, most formulations assume that the problem subsystems share only a relatively small number of variables, i. e., that the *bandwidth of interdisciplinary coupling* is relatively small.
- The problem is partitioned into a system (or upper-level) problem and subsystem (or lower-level) problems. Decisions are made on inclusions of particular variables and constraints into the system and subsystems. Decisions are also made on the form of the system and subsystem objectives.
- Finally, algorithms are selected for solving the system and subsystem optimization problems. One must distinguish a formulation of the problem from the algorithm used to solve that formulation. While some of the multilevel formulations can be easily shown to be mathematically equivalent to the original problem with respect to solution sets, they may not be equivalent with respect to other attributes, such as constraint qualifications and optimality conditions. Hence the numerical properties of algorithms applied to different formulations vary widely [8,9,10].

Problem *decomposition* constitutes a special area of study. In general, decomposition techniques take advantage of the problem structure and depend on the strength and bandwidth of couplings among the subsystems. *Separable* and partially separable problems are particularly amenable to decomposition.

Two types of decomposition may be considered in design optimization. Coarse-grained decomposition with respect to disciplines presents no difficulty, because the design problem initially consists of autonomous parts. The difficulty at this level of problem formulation is in *integration* or *synthesis*. However, in realistic applications, even though the coarse-grained decomposition is frequently obvious, the complexity of the problem requires that a *dependence analysis* be performed in order to determine the most advantageous arrangement or sequencing of the disciplinary subsystems in the optimization procedure. Automatic techniques based on graph-theoretic foundations may be found in [78] and [79], for instance.

Finer-grained decomposition within a particular discipline may be addressed by a multitude of techniques for decomposition of mathematical programs. Extensive references on decomposition in general mathematical programming, beginning with [19] and [31], and extended in [49] and many others, can be found in [42] and [43]. Further references to decomposition techniques aimed specifically at design problems can be found in [95].

General multilevel programming presents an exceedingly difficult problem, and many multilevel formulations and algorithms of engineering design rely more on heuristics than on theoretically substantiated foundations. There are exceptions, for instance, such as those in [12,29,68], and [75]. While many engineering multilevel approaches have enjoyed success when applied to specific problems, insufficient analytical foundation and the difficulty of the problem usually mean that the approaches are not robust, and extensive 'fine-tuning' of heuristic parameters is required for each new problem or instance of a problem. Hence, recent years have seen renewed interest in systematic, analytically substantiated approaches to MLP. Many such developments have taken place in *bilevel optimization*.

Bilevel Optimization

Although bilevel optimization problems (BLP) form the simplest case of multilevel optimization, they are very difficult to solve and constitute a fertile research area. A survey of the field can be found in [28]. A large bibliography with an emphasis on theoretical developments is also provided in [94].

The conventional general bilevel problem may be posed as follows:

$$\begin{cases} \min_{x\in X} & f_1(x,y) \\ \text{s.t.} & h_1(x,y)=0 \\ & g_1(x,y)\geq 0, \end{cases}$$

where y solves for fixed x:

$$\begin{cases} \min_{y\in Y} & f_2(x,y) \\ \text{s.t.} & h_2(x,y)=0 \\ & g_2(x,y)\geq 0. \end{cases}$$

The cases of linear and convex problem functions have been studied widely. A popular class of methods for the linear bilevel problem (extreme point algorithms) computes global solutions by enumerating extreme points of the lower-level feasible set (e. g., [27]). Convex bilevel problems are often solved by branch and bound methods (e. g., [15]). A survey of methods for linear and convex bilevel programming can be found in [11].

The considerably more difficult case of nonlinear and nonconvex problem functions has inspired much research activity as well but has, to date, led to few computationally successful algorithms. The existing approaches to nonlinear bilevel optimization can be classified into several categories.

Penalty-Based Methods

This category uses penalty methods. In some algorithms (e. g., [1]), a barrier function penalizes the lower-level objective. In double-penalty methods, both the lower-level problem and the upper-level problem are approximated by sequences of unconstrained optimization problems [56,61,64]. Single or double-penalty methods are, in general, expected to converge slowly, especially for highly nonlinear problems. Thus using these methods for the usually large and nonlinear design optimization problems may be difficult.

KKT-Based Methods

The algorithms of this category convert the bilevel problem into a nonconvex, single-level optimization problem by using the *Karush—Kuhn—Tucker conditions* (KKT conditions) of the lower-level problem as constraints on the upper-level problem [14,15,20,37,44]. If the lower-level problem is convex, the KKT formulation is equivalent to the original formulation [14]. However, even in this case, the KKT conditions on the lower-level problem include the *complementarity slackness* condition as a constraint. The form of the complementarity condition makes the single-level problem difficult to solve. The KKT formulation suffers from an additional difficulty. Namely, it is well known from the study of the sensitivity and stability of nonlinear programming (e. g., [40]) that even if the lower-level problem behaves exceedingly well in that it satisfies such stringent assumptions as *strong second order sufficiency* and *regularity* as a *constraint qualification*, the *feasible set* of the single-level problem will generally not be differentiable with respect to x. Hence, the performance of gradient-based solvers on the transformed problem may be adversely affected.

Descent-Based Methods

Another category of algorithms is based on solving subproblems that result in descent for the upper-level problem with gradient information of the lower-level problem used in a number of ways [34,39,59,83].

The remainder of the article will be devoted to a more detailed description of two specific approaches to nonlinear, nonconvex problems that arose from the need to solve engineering design problems. One approach is a bilevel formulation, the other is an algorithm for solving multilevel formulations.

Examples: Collaborative Optimization

Collaborative optimization (CO) is a general approach to solving multidisciplinary design optimization problems by formulating them as nonlinear bilevel programs of special structure. CO comprises a number of methods. Its antecedents can be traced to earlier hierarchical approaches, as in [60] and [98]. The underlying idea of CO appeared in [13,80,81,82,88] and [96,97]. The approach has recently received attention under the name of *collaborative optimization* [22,23,86,93].

Given that MDO problems are naturally partitioned into subsystems along disciplinary lines, CO suggests an intuitively attractive way to formulate the optimization problem so that the autonomy of the disciplinary subsystem computations is preserved. However, the approach presents a problem that is difficult to solve by means of conventional nonlinear programming software [7,58]. The analytical and computational aspects of CO were addressed in [9], of which the following discussion is an abstract. As a complete description of CO is lengthy, only an abbreviated version is considered here.

It is assumed that the original system is composed of a number, say M, of interdependent but autonomous systems, each of which is described by a disciplinary analysis A_i, $i = 1, \dots, M$, expressed in the form

$$A_i(x_i, y_i(x_i)) = 0,$$

where, given a vector of disciplinary design variables x_i, the analysis (frequently represented by a numerical differential equation solver or simulator) is performed to yield the vector of state variables or responses $y_i(x_i)$. The sets of disciplinary variables x_i are not necessarily disjoint. The disciplinary constraints are usually represented by inequalities

$$c_i(x_i, y_i(x_i)) \geq 0.$$

Once the system objective and variables and the subsystem constraints and variables are identified, the bilevel problem is formed as follows:

The constraints of the system problem comprise the 'consistency' (or 'coupling' or 'matching') conditions that are used to drive the discrepancy among the inputs and outputs shared by the subsystems to zero. The values of the constraints are computed by solving the subsystem optimization problems, and the number of *consistency constraints* is related to the number of subsystems and variables shared among the subsystems. The form of the consistency constraints determines a particular implementation of CO.

Let ξ and η represent system-level variables corresponding to inputs and outputs of subsystems, respectively. Then, given M subsystems, the abbreviated system program is

$$\begin{cases} \min & F(\xi, \eta) \\ \text{s.t.} & G(\xi, \eta) = 0, \end{cases} \qquad (1)$$

where

$$G(\xi, \eta) = \begin{pmatrix} g_1(\xi, \eta) \\ \vdots \\ g_M(\xi, \eta) \end{pmatrix}$$

is the set of system consistency constraints obtained by solving lower-level subproblems, each of which is of the form

$$\begin{cases} \min & \frac{1}{2}\left[\|\xi_i - x_i\|^2 + \|\eta_i - y(x_i)\|^2\right] \\ \text{s.t.} & c_i(x_i, y(x_i)) \geq 0, \end{cases} \qquad (2)$$

where i is the number of the subsystem. Thus, the objective of a subsystem optimization problem is always to minimize the discrepancy between the shared variables of the subsystems, in a least squares sense, subject to satisfying the disciplinary constraints, which do not depend explicitly on the system variables passed down to the subsystems as parameters. The subsystems remain feasible during optimization, while *interdisciplinary feasibility* is gradually attained at the system level via the consistency constraints. Maintaining disciplinary feasibility is extremely important from the design perspective.

The problem now consists of a set of decoupled subproblems that can be solved independently and in parallel.

One instance of the system-level consistency conditions gives rise to the form in which CO is usually presented: namely, the consistency condition is intended to drive to zero the value function of the subproblem (2). That is,

$$g_i(\xi, \eta) = \frac{1}{2}\|\xi - x_*\|^2 + \|\eta - y(x_*)\|^2, \qquad (3)$$

where x_* solves the subsystem optimization problem.

Another instance of system-level consistency conditions matches the system-level variables with their subsystem counterparts computed in subproblem

$$g_i(\xi, \eta) = (\xi - x_*, \eta - y(x_*)). \qquad (4)$$

The behavior of optimization algorithms applied to the original and CO formulations will differ greatly, as the formulations are not equivalent with respect to constraint qualifications or optimality conditions.

In general, value functions are not differentiable, and this may cause difficulties for optimization algorithms applied to the system-level problem. However, under a number of strong assumptions, the constraints are locally differentiable and can usually be computed.

Derivatives of the system-level constraints with respect to the system-level design variables are the sensitivities of the minima or the solutions of the subsystem-level optimization problems to parameters. The area of *sensitivity in nonlinear programming* has been studied extensively. Relevant results can be found in [40] and [41]. In particular, under the assumptions of sufficient smoothness, *second order sufficiency*, regularity as constraint qualification, and *strict complementarity slackness*, the basic sensitivity theorem (BST) proves the existence of a unique, local, continuously differentiable solution-multiplier triple for the perturbed problem. Moreover, locally, the set of active constraints remains unchanged and regularity and strict complementarity hold, allowing one to compute derivatives locally. In fact, under a number of assumptions, stronger statements can be made about the differentiability of the value function [30,77].

Under the conditions of BST, local first order derivatives of the consistency constraints (3) have a particularly simple form because, in the case of CO, the constraints of the lower-level problems do not depend on parameters. On the other hand, the first order sensitivities of solutions of the lower-level problem that form the derivatives of the consistency constraints (4), while of closed form, are expensive to compute and involve second order derivatives of the subsystem Lagrangians.

There is another feature of the CO formulation with compatibility constraints (3) that will cause difficulties for nonlinear programming algorithms applied to the system-level problem: *Lagrange multipliers* will almost never exist for the equality constrained system level problem, with all the ensuing consequences. The nonexistence of Lagrange multipliers is due to the description of the feasible region that causes the Jacobian of the system-level constraints to vanish at a solution. The formulation with compatibility constraints (4) aims to address this problem. However, the computation of derivatives for this formulation is clearly expensive, as it not only involves solving a system of equations, but also requires the computation of second order information for the subsystems. The difficulties are addressed in detail in [9].

In summary, CO is an appealing approach to design optimization; however, the bilevel nature of the problem formulation will cause difficulties for conventional nonlinear programming algorithms applied to the system-level problem. Variations, special algorithms for solution, and alternatives can be found in, e. g. [33,54,55].

Example: MAESTRO, a Class of Multilevel Algorithms

As mentioned earlier, most multilevel formulations and algorithms for engineering design problems assume that the bandwidth of coupling among the subsystems comprised by the multilevel system is small. While many problems may be stated in this way, it is becoming increasingly important to consider problems with large bandwidth of coupling where, to use an MDO expression, 'everything affects everything else'. MAESTRO (a class of multilevel algorithms for constrained optimization; [2]) is intended for solving large nonlinear programming problems with arbitrary couplings among the naturally occurring subsystems, i. e., a particular instance of MDO problems with a single objective. The class was extended in, e. g., [5] to include a large class of steps for the nonlinear programming problem and in [3,4] to incorporate general nonlinear objectives. The class makes no assumptions on the structure of the problem, such as convexity or separability.

The algorithms of the class are based on *trust region methodology* (see, e. g., [35,38,67]) and are proven to converge under reasonable assumptions.

The idea of the MAESTRO algorithms is to attain sequential predicted *sufficient decrease conditions* for all the constrained objectives, and is a direct extension of the multilevel ideas for the equality constrained optimization problem. The approach can be summarized as follows. Given an initial approximation to the solution of the multilevel problem, the *trial step* for the multilevel problem is computed as a sum of a sequence of substeps, each of which predicts sufficient (or optimal) decrease in the quadratic model of the objective of a given subproblem, subject to maintaining predicted decrease in the models of the previous objectives. For instance, in the case of the unconstrained bilevel problem, the trial step for the bilevel problem is a sum of two substeps. The first substep is computed to predict sufficient decrease, via the quadratic model of the innermost objective f_2, for the subproblem of approximately optimizing

$$m_{f_2}(s) \equiv f_2(x_c) + \nabla f_2(x_c)^\top s + \frac{1}{2}s^\top H_2(x_c)s,$$

in the trust region of size δ_{f_2} to produce the substep s_{f_2}, where x_c is the current approximation to the solution and H_2 is the current approximation to the Hessian of f_2. The second step s_{f_1} would then approximately minimize the quadratic model of the outermost objective f_1, constructed at $x_c + s_{f_2}$, in the trust region of size δ_{f_1}, subject to constraints that enforce the preservation of the predicted sufficient or optimal decrease for f_1. The total trial step is evaluated by using the merit function designed to account for the sequential processing of the objectives. The algorithm is shown to converge to critical points of the bilevel or multilevel problem. Thus, the essential difference between this approach and the classical approaches to bilevel optimization is that instead of starting from the optimality conditions for the bilevel or multilevel problem, the approach attempts to obtain decrease on the sequence of subproblem models, while preserving predicted decrease for the previously processed subproblems, and to measure progress via the use of an appropriate merit function with rigorously updated penalty parameters. It is important to emphasize that the merit function is used only to evaluate the steps, and not to compute them.

The ongoing work is concerned with practical implementation issues and applications to engineering design problems.

Summary

Multilevel optimization has been an active research field, both in applied mathematics and in engineering design. Many open questions remain, in particular, in the area of practical computational algorithms for bilevel and multilevel problems. Overviews of some recent developments can be found in [66].

Understanding the behavior of specific, nonlinear programming algorithms applied to the system-level problem of the bilevel or multilevel formulations will present an interesting and difficult area of inquiry, and would benefit from the techniques of *nonsmooth analysis and optimization* [32,36,47], unconventional notions of constraint qualifications [24,25], and optimality [99,100].

To facilitate research and testing in the area of algorithms, one may find automatic bilevel and multilevel problem generators, as well as other sources of multilevel problems, described in [26,72,73].

See also

- ▶ Bilevel Fractional Programming
- ▶ Bilevel Linear Programming
- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Bilevel Programming
- ▶ Bilevel Programming: Applications
- ▶ Bilevel Programming: Applications in Engineering
- ▶ Bilevel Programming: Implicit Function Approach
- ▶ Bilevel Programming: Introduction, History and Overview
- ▶ Bilevel Programming in Management
- ▶ Bilevel Programming: Optimality Conditions and Duality
- ▶ Design Optimization in Computational Fluid Dynamics
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Multidisciplinary Design Optimization
- ▶ Multilevel Optimization in Mechanics
- ▶ Optimal Design of Composite Structures
- ▶ Optimal Design in Nonlinear Optics
- ▶ Stochastic Bilevel Programs
- ▶ Structural Optimization: History

References

1. Aiyoshi E, Shimizu K (1981) Hierarchical decentralized system and its new solution by a barrier method. IEEE Trans Syst, Man Cybern 11:444–449
2. Alexandrov NM (1993) Multilevel algorithms for nonlinear equations and equality constrained optimization. PhD Thesis Rice Univ.
3. Alexandrov NM (1996) Multilevel and multiobjective optimization in multidisciplinary design. AIAA Paper no. 96-4122
4. Alexandrov NM (2000) A trust-region algorithm for nonlinear bilevel optimization. in preparation
5. Alexandrov NM, Dennis JE (1995) Multilevel algorithms for nonlinear optimization. In: Borggaard J, Burkardt J, Gunzburger M, Peterson J (eds) Optimal Design and Control. Birkhäuser, Basel, pp 1–22
6. Alexandrov NM, Hussaini MY (eds) (1997) Multidisciplinary design optimization: State of the art. SIAM, Philadelphia
7. Alexandrov NM, Kodiyalam S (1998) Initial results of an MDO method evaluation study. AIAA Paper no. 98-4884

8. Alexandrov NM, Lewis RM (2000) Algorithmic perspectives on problem formulations in MDO. AIAA Paper no. 2000-4719
9. Alexandrov NM, Lewis RM (2002) Analytical and computational aspects of collaborative optimization for multidisciplinary design. AIAA J 40:301–309
10. Alexandrov NM, Lewis RM (2000) Analytical and computational properties of distributed approaches to MDO. AIAA Paper no. 2000-4718
11. Anadalingam G, Friesz TL (1992) Hierarchical optimization: An introduction. Ann Oper Res 34:1–11
12. Badhrinath K, Rao JRJ (1994) Bilevel models for optimum designs which are insensitive to perturbations in variables and parameters. Techn Report Univ Houston UH-ME-SDL-94-03
13. Balling RJ, Sobieszczanski-Sobieski J (1994) An algorithm for solving the system-level problem in multilevel optimization. In: Fifth AIAA/USAF/NASA/ISSMO Symp. Multidisciplinary Analysis and Optimization (Panama Beach, Florida, Sept. 7-9, 1994), AIAA Paper no. 94-4333 (1994)
14. Bard JF (1983) An algorithm for solving the general bilevel programming program. Math Oper Res 8:260–272
15. Bard JF, Falk JE (1982) An explicit solution to the multilevel programming problem. Comput Oper Res 9:77–100
16. Barthelemy J-FM (1988) Engineering applications of heuristic multilevel optimization methods. NASA TM-101504
17. Barthelemy J-FM, Riley MF (1988) Improved multilevel optimization approach for the design of complex engineering systems. AIAA J 26:353–360
18. Basar T, Selbuz H (1979) Closed loop Stackelberg strategies with applications in optimal control of multilevel systems. IEEE Trans Autom Control AC-24:166–178
19. Benders JF (1962) Partitioning procedures for solving mixed variables programming problems. Numerische Math 4:238–252
20. Bialas WF, Karwan MH (1982) On two-level optimization. IEEE Trans Autom Control AC-27:211–214
21. Bracken J, McGill J (1973) Mathematical programs with optimization problems in the constraints. Oper Res 21:37–44
22. Braun RD (1996) Collaborative optimization: An architecture for large-scale distributed design. PhD Thesis Stanford Univ.
23. Braun RD, Moore AA, Kroo IM (1997) Collaborative approach to launch vehicle design. J Spacecraft and Rockets 34:478–486
24. Burke JV (1991) Calmness and exact penalization. SIAM J Control Optim 29:493–497
25. Burke JV (1991) An exact penalization viewpoint of constrained optimization. SIAM J Control Optim 29:968–998
26. Calamai PH, Vicente LN (1993) Generating linear and linear-quadratic bilevel programming problems. SIAM J Sci Comput 14:770–782
27. Candler W, Townsley R (1982) A linear two-level programming problem. Comput Oper Res 9:59–76
28. Chen Y (1994) Bilevel programming problems: Analysis, algorithms and applications. PhD Thesis Univ. Montreal
29. Chidambaram B, Rao JRJ (1994) A study of constraint activity in bilevel models of optimal design. Techn Report Univ Houston UH-ME-SDL-94-01
30. Clarke FH (ed) (1990) Optimization and nonsmooth analysis. SIAM, Philadelphia
31. Danzig GB, Wolfe P (1960) Decomposition principle for linear programming. Oper Res 8:101–111
32. De Luca A, Di Pillo G (1987) Exact augmented Lagrangian approach to multilevel optimization of large-scale systems. Internat J Syst Sci 18:157–176
33. De Miguel W, Murray W (2006) A Local Convergence Analysis of Bilevel Decomposition Algorithms. Optim Eng 7:99–133
34. De Silva AH, McCormick GP (1992) Implicitly defined optimization problems. Ann Oper Res 34:107–124
35. Dennis JE Jr, Schnabel RB (1983) Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, Englewood Cliffs, NJ
36. Dirickx YMI, Jennergren LP (1979) Systems analysis by multilevel methods. Wiley, New York
37. Edmunds TA, Bard JF (1991) Algorithm for nonlinear bilevel mathematical programs. IEEE Trans Syst, Man Cybern 21:83–89
38. El-Alem MM (1991) A global convergence theory for the {Celis–Dennis–Tapia} trust region algorithm for constrained optimization. SIAM J Numer Anal 28:266–290
39. Falk JE, Liu J (1995) On bilevel programming, Part I: General nonlinear cases. Math Program 70:47–72
40. Fiacco AV (ed) (1983) Introduction to sensitivity and stability analysis in nonlinear programming. Acad. Press, New York
41. Fiacco AV, McCormick GP (eds) (1990) Nonlinear programming, sequential unconstrained minimization techniques. SIAM, Philadelphia
42. Flippo OE (1989) Stability, duality and decomposition in general mathematical programming. PhD Thesis, Erasmus Univ. Rotterdam
43. Flippo OE, Rinnooy Kan AHG (1993) Decomposition in general mathematical programming. Math Program 60:361–382
44. Fortuny-Amat J, McCarl B (1981) A representation of a two-level programming problem. J Oper Res Soc 32:783–792
45. Friesz T, Tobin R, Cho H, Mehta N (1990) Sensitivity analysis based heuristic algorithms for mathematical programs with variational inequality constraints. Math Program 48:265–284
46. Gahutu DWH, Looze DP (1985) Parametric coordination in hierarchical control. Large Scale Systems 8:33–45
47. Gauvin J (1978) The method of parametric decomposition in mathematical programming: the nonconvex case.

In: Lemarechal C, Griffin R (eds) Nonsmooth optimization. Pergamon, Oxford, pp 131–149

48. Gauvin J, Dubeau F (1983) Some examples and counterexamples for stability analysis of nonlinear programming problems. In: Fiacco AV (ed) Math. Program. Stud., vol 21. North-Holland, Amsterdam, pp 69–78
49. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
50. Goulbeck B, Brdys M, Orr CH, Rance JP (1988) A hierarchical approach to optimized control of water distribution systems: Part I, decomposition. Optimal Control Appl Meth 9:51–61
51. Goulbeck B, Brdys M, Orr CH, Rance JP (1988) A hierarchical approach to optimized control of water distribution systems: part II, lower-level algorithm. Optimal Control Appl Meth 9:109–126
52. Haimes YY (1977) Hierarchical analyses of water resources systems. McGraw-Hill, New York
53. Haimes YY, Tarvainen K, Shima T, Thadathil J (1990) Hierarchical multiobjective analysis of large-scale systems. Hemisphere, Washington, DC
54. Hafka RT, Watson LT (2005) Multidisciplinary Design Optimization Problems with Quasiseparable Subsystems. Optim Eng 6:9–20
55. Hafka RT, Watson LT (2006) Decomposition Theory for Multidisciplinary Design Optimization Problems with Mixed Integer Quasiseparable Subsystems. Optim Eng 7:135–149
56. Ishizuka Y, Aiyoshi E (1992) Double penalty method for bilevel optimization problems. Ann Oper Res 34:73–88
57. Kirsch U (1989) Improved optimum structural design by passive control. Engin with Comput 5:13–22
58. Kodiyalam S (1998) Evaluation of methods for multidisciplinary design optimization (MDO), phase I. NASA Contractor Report
59. Kolstad CD, Lasdon LS (1990) Derivative evaluation and computational experience with large bilevel mathematical programs. J Optim Th Appl 65:485–499
60. Lasdon LS (1970) Optimization theory for large systems. MacMillan, New York
61. Loridan P, Morgan J (1988) Quasiconvex lower level problems and applications in two-level optimization. Techn Report Univ Montreal CRM-1578
62. Mahmoudi MS (1977) Multilevel systems control and applications: A survey. IEEE Trans Syst, Man Cybern SMC-7:125–143
63. Marcotte P (1985) A new algorithm for solving variational inequalities with application to the traffic assignment problem. Math Program 33:339–351
64. Marcotte P, Zhu DL (1996) Exact and inexact penalty methods for the generalized bilevel programming problem. Math Program 74:141–157
65. Mesarović MD, Macko D, Takahara Y (1970) Theory of hierarchical, multilevel, systems. Acad. Press, New York
66. Migdalas A, Pardalos PM, Värbrand P (eds) (1998) Multilevel optimization: Algorithms and applications. Kluwer, Dordrecht
67. Moré JJ (1991) Recent developments in software for trust region methods. In: Bachem A, Grötschel M, Korte B (eds) Mathematical Programming: The State of the Art. Springer, Berlin, pp 266–290
68. Muralidhar R, Rao JRJ, Badhrinath K, Kalagatla A (1995) Multilevel formulations and limit analysis and design of structures with bilateral contact constraints. Techn Report Univ Houston UH-ME-SDL-95-02
69. Nachane DM (1984) Optimization methods in multilevel systems: A methodological survey. Europ J Oper Res 21:25–38
70. Nicholls MG (1995) Aluminium production modelling – a non-linear bi-level programming approach. Oper Res 43:208–218
71. Outrata J (1994) On optimization problems with variational inequality constraints. SIAM J Optim 4:340–357
72. Padula SL, Alexandrov NM, Green LL (1996) MDO test suite at NASA Langley Research Center. In: Proc. Sixth AIAA/NASA/ISSMO Symp. Multidisciplinary Analysis and Optimization, AIAA
73. Padula SL, Young KC (1986) Simulator for multilevel optimization research. NASA Techn Memorandum TM-87751
74. Rao JRJ, Badhrinath K (1996) Solution of multilevel structural design problems using a nonsmooth algorithm. In: Proc. Sixth AIAA/NASA/ISSMO Symp. Multidisciplinary Analysis and Optimization, AIAA, AIAA-96-3986-CR.
75. Rao JRJ, Chidambaram B (1993) Parametric deformation and model optimality in concurrent design. Techn Report Univ Houston UH-ME-SDL-93-01
76. Reddy SY, Fertig KW, Smith DE (Aug. 1996) Constraint management methodology for conceptual design tradeoff studies. In: Proc. DETC'96, ASME Paper 96-DETC/DTM-1228
77. Rockafellar RT (1984) Directional differentiability of the optimal value function in a nonlinear programming problem. Math Program Stud 21:312–226
78. Rogers JL (1996) DeMAID/GA – An enhanced design manager's aid for intelligent decomposition. Proc. Sixth AIAA/NASA/ISSMO Symp. Multidisciplinary Analysis and Optimization,), AIAA Paper 96-4157.
79. Rogers JL (1996) DeMAID/GA user's guide – Design manager's aid for intelligent decomposition with a genetic algorithm. NASA Langley Res Center TM-110241
80. Schmitt LA, Chang KJ (1984) A multilevel method for structural synthesis. In: A Collection of Technical Papers: AIAA/ASME/ASCE/AHS 25th Structures, Structural Dynamics and Materials Conf., AIAA
81. Schmitt LA Jr, Mehrinfar M (1982) Multilevel optimum design of structures with fiber-composite stiffened panel components. AiAA J 20(1):138–147

82. Schmitt LA Jr, Ramanathan RK (1978) Multilevel approach to minimum weight design inclusing buckling constraints. AiAA J 16(2):97–104
83. Shimizu K, Aiyoshi E (1981) A new computational method for Stackelberg and min-max problems by use of a penalty method. IEEE Trans Autom Control AC-26:460–466
84. Simaan M, Cruz JB Jr (1973) On the Stackelberg strategy in nonzero-sum games. J Optim Th Appl 11(5):535–555
85. Singh MG, Mahmoud MS, Titli A (1981) A survey of recent developments in hierarchical optimization and control. Proc. IFAC Control Sci. and Techn. 8th Triennial World Congress, Kyoto, Japan, IFAC
86. Sobieski I, Kroo I (1996) Aircraft design using collaborative optimization. In: AIAA paper 96-0715 Presented at the 34th AIAA Aerospace Sci. Meeting, Reno, Nevada, Jan. 15-18, 1996, AIAA
87. Sobieszczanski-Sobieski J (1993) Optimization by decomposition. In: Kamat MP (ed) Structural Optimization: Status and Promise. Progress in Astronautics and Aeronautics, vol 150. AIAA, pp 487–515
88. Sobieszczanski-Sobieski J (1993) Two alternative ways for solving the coordination problem in multilevel optimization. Structural Optim 6:205–215
89. Sobieszczanski-Sobieski J (1996) Multidisciplinary aerospace design optimization: Survey of recent developments. In: Proc. 34-th Aerospace Sci. Meeting and Exhibit, Reno, Nevada, AIAA, AIAA paper 96-0711.
90. Sobieszczanski-Sobieski J, Haftka RT (1997) Multidisciplinary aerospace design optimization: survey of recent developments. Structural Optim 14:1–23
91. Sobieszczanski-Sobieski J, James BB, Dovi AR (1985) Structural optimization by generalized multilevel optimization. AIAA J 23:1775–1782
92. Stackelberg H (ed) (1952) The theory of the market economy. Oxford Univ. Press, Oxford
93. Tappeta RV, Renaud JE (1997) Multiobjective collaborative optimization. J Mechanical Design 119:403–411
94. Vicente LN, Calamai PH (1994) Bilevel and multilevel programming: A bibliographic review. J Global Optim 5:291–306
95. Wagner TC (1993) A general decomposition methodology for optimal system design. PhD Thesis, Univ. Michigan
96. Walsh JL, LaMarsh WJ, Adelman HM (1992) Fully integrated aerodynamic/dynamic/structural optimization of helicopter rotor blades. NASA TM-104226
97. Walsh JL, Young KC, Pritchard JI, Adelman HM, Mantay WR (1995) Integrated aerodynamic/dynamic/structural optimization of helicopter rotor blades using multilevel decomposition. NASA TP-3465
98. Wismer DA (ed) (1971) Optimization methods for large-scale systems. McGraw-Hill, New York
99. Ye JJ, Zhu DL, Zhu QJ (1997) Exact penalization and necessary optimality conditions for generalized bilevel programming problems. SIAM J Optim 7:481–507
100. Zhang R (1994) Problems of hierarchical optimization in finite dimensions. SIAM J Optim 4:521–536

Multilevel Optimization in Mechanics

GEORGIOS E. STAVROULAKIS[1],
EURIPIDIS MISTAKIDIS[2], OLYMPIA PANAGOULI[3]
[1] Carolo Wilhelmina Techn. Universität, Braunschweig, Germany
[2] University Thessaly, Volos, Greece
[3] Aristotle University, Thessaloniki, Greece

MSC2000: 49Q10, 74K99, 74Pxx, 90C90, 91A65

Article Outline

Keywords

Multilevel optimization; Computational mechanics; Parallel computation in mechanics

Multilevel optimization methods have been developed first in the period after 1960. The main scope was to facilitate the optimization of large scale systems in industrial processes and to solve trajectory determination and prediction problems using trajectory decomposition techniques. The reader may refer in this respect to the corresponding articles [3] and [26] and to the references given there but also to the books [27] and [12]. More recent works on this subject have been published in [4,14]. It should be mentioned that certain sources concerning the ideas of multilevel optimization may be found in well-known treatises of calculus of variations and theoretical mechanics, cf. e. g. [5,10]. Indeed, the well-known procedure of variational methods in Mechanics of ‘frozen’ variables or constraints has a great relationship with the ideas of multilevel optimization. Also the well-known iterative methods of H. Cross and G. Kany of linear structural analysis used after 1940 and before the development of computer codes based on the *finite element method* (FEM), for the calculation of framed structures, are nothing than a formulation in the 'language' of structural analysis of a multilevel optimization algorithm for the minimization problem of the complementary energy of the structure, expressed in terms of the bending moments of the beam and column connections.

Among the pioneers in the application of the multilevel optimization methods in mechanics and especially concerning the calculation of structures involving inequality constraints was P.D. Panagiotopoulos [19,20]. The idea was the following: Most mechanical problems can be expressed as the minimum problems of an appropriately formulated energy function. The decomposition of this initial optimization problem into smaller subproblems corresponds to the energetic decomposition of the initial mechanical problem into smaller fictitious subproblems. The mutual interaction of these subproblems yields, after an iterative procedure, the solution of the initial problem. The aforementioned method leads to the following three main applications of the multilevel optimization techniques in the framework of Mechanics and more generally in engineering sciences.

a) Calculation of large structures.
b) Validation of the simplifying assumptions used for the calculation of complex structures. Accuracy testing.
c) Accuracy improvement of simplified models used for the estimation of the behavior of complex structures.

Note that in the above, the term 'structure' can be replaced with the term 'systems', meaning systems whose behavior is characterized by the solution of a minimax problem.

Since most of the multilevel techniques developed in the early sixties for the trajectory determination problems in space science are also applicable to stationarity problems, and since recently it has been proved that in the dynamic problems involving impact phenomena the functional of the action is stationary [22,23] it results that there is also a further application of the multilevel optimization methods:

d) Calculation of the dynamic behavior of structures involving impact effects.

To the aforementioned applications the following, classical one, can be added.

e) Solution of optimal control (minimum of weight or cost, maximum of strength) in dynamic structural analysis problems.

This article deals mainly with static systems. Concerning the application d) and e) the reader is referred to [12,27] in relation with [22,23]. In dynamic problems analogous methods to the static problems can be developed.

The classical *decomposition techniques* which are applied to optimization problems (cf. in this respect also [20, pp. 355ff]) have been extended and they can be applied also to *substationarity problems* [25], i. e. to problems of the type

$$0 \in \overline{\partial} f(x),$$

where f is a nonconvex nonsmooth energy function and $\overline{\partial}$ denotes the generalized gradient of F.H. Clarke [7] as it has been extended by R.T. Rockafellar [25] for nonLipschitzian functionals. In this case the *variational inequalities* of the convex energy problems are replaced by *hemivariational inequalities* (cf. e. g. [8,17,20,21]) and instead of a global minimum of the convex potential or complementary energy functionals, the local minima and maxima are searched and among them the global minimum as well. For the numerical treatment of hemivariational inequalities certain numerical methods have been developed (cf. e. g. [21]) and among them, the two methods described in [15] are extensions of the multilevel optimization methods to substationarity problems.

It should also be noted that most of the domain decomposition methods are special cases of the multilevel optimization algorithms, as it results easily if one considers the energy functionals corresponding to the partial differential equations studied. Then the domain decomposition leads to energy functionals which have to be minimized on the decomposed parts of the domain.

Finally, it should be mentioned that fractal geometries in optimization problems arising in Mechanics are treated by means of appropriate multilevel transformations of the problem as is will be shown further. It is evident that an optimization problem with many variables cannot always directly be decomposed into independent optimization subproblems. The aim of the multilevel optimization is to define with respect to an optimization problem, appropriate mutually independent subproblems. Each of these when solved independently yields the optimum of the overall problem after an iterative procedure which is called second-level controller. The decomposition into subproblems is achieved by choosing some variables, called coordinating variables, which are freely manipulated by the second-level controller in such a way that the subproblems (first-level of the problem) have solutions which in fact yield the optimum of the initial problem, i. e. before its decomposition into subproblems. Here, the ideas of [3] are closely followed.

There are several different methods of transforming a given constrained optimization problem into a multilevel optimization problem. All these methods are basically combination of two methods: the *feasible decomposition method* or *model coordination method* and the *nonfeasible decomposition method* or *goal coordination method*.

Let us consider the problem

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{u}} & \Pi(\mathbf{x},\mathbf{u}) \\ \text{s.t.} & \mathbf{f}(\mathbf{x},\mathbf{u}) = 0 \\ & \mathbf{R}(\mathbf{x},\mathbf{u}) \geq 0, \end{cases} \tag{1}$$

where $\mathbf{x}$ is a vector in E_n, $\mathbf{u}$ is a vector in E_m, $\mathbf{f}$ is an n vector of C^2 functions, Π is a twice continuously differentiable (C^2) function, and $\mathbf{R}$ is an r vector of C^2 functions. To decompose, coordinating variables $\mathbf{s}$ may be substituted not only for a single variable but also, for functions $\mathbf{g}(\mathbf{x},\mathbf{u})$, so that Π is splitted into mutually disjoint parts and the $\mathbf{f}$ and $\mathbf{R}$ equations contain no common $\mathbf{x}$, $\mathbf{u}$, or $\mathbf{s}$ variables between the subproblems. Thus the following problem results:

$$\Pi(\mathbf{x},\mathbf{u},\mathbf{s}) = \sum_{i=1}^{N} \Pi^{(i)}(\mathbf{x}^{(i)},\mathbf{u}^{(i)},\mathbf{s}^{(i)})$$
$$\mathbf{f}^{(i)}(\mathbf{x}^{(i)},\mathbf{u}^{(i)},\mathbf{s}^{(i)}) = 0, \quad i = 1,\ldots,N,$$
$$\mathbf{R}^{(i)}(\mathbf{x}^{(i)},\mathbf{u}^{(i)},\mathbf{s}^{(i)}) \geq 0, \quad i = 1,\ldots,N.$$

The (i) denotes to the ith subproblem or subsystem which must be optimized. For example in a control problem $\mathbf{x}$ denotes the state, $\mathbf{u}$ denotes the control and $\mathbf{x}^{(1)}$ is the state vector for the first subsystem. Also the coupling equations must be added:

$$\mathbf{s}^{(i)} = \mathbf{g}^{(i)}(\mathbf{x}^{(j)},\mathbf{u}^{(j)}) \quad \text{for all } j \neq i.$$

The Lagrangian of the new problem reads

$$\begin{aligned}\widetilde{\Pi}(\mathbf{x},\mathbf{u},\mathbf{s};\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\rho}) &= \sum_{i=1}^{N}\Pi^{(i)} + \sum_{i=1}^{N}\boldsymbol{\lambda}^{(i)\top}\mathbf{f}^{(i)} + \sum_{i=1}^{N}\boldsymbol{\mu}^{(i)\top}(\mathbf{R}^{(i)} - \boldsymbol{\sigma}^{(i)}) \\ &\quad + \sum_{i=1}^{N}\boldsymbol{\rho}^{(i)\top}(\mathbf{g}^{(i)} - \mathbf{s}^{(i)}),\end{aligned} \tag{2}$$

where $\boldsymbol{\sigma}^{(i)} \geq 0$ are additional slack variables such that

$$\mathbf{R}^{(i)} - \boldsymbol{\sigma}^{(i)} = 0.$$

$\widetilde{\Pi}$ is immediately separable into N individual subsystems, except for its last term.

In the method of nonfeasible decomposition it is assumed that $\boldsymbol{\rho}^{(i)}$ has a known value. The term $\boldsymbol{\rho}^{(i)\top}\mathbf{s}^{(i)}$ is put in the ith subsystem and all of the $\boldsymbol{\rho}^{(i)\top}\mathbf{g}^{(i)}(\mathbf{x}^{(j)},\mathbf{u}^{(j)})$ terms associated with the jth variables are put in the jth subsystem. On the other hand, in the feasible decomposition method it is assumed that $\mathbf{s}^{(i)}$ has a known value. Moreover, all of the $\boldsymbol{\rho}^{(i)\top}[\mathbf{g}^{(i)}(\mathbf{x}^{(j)},\mathbf{u}^{(j)}) - \mathbf{s}^{(i)}]$ terms associated with the jth variables are put in the jth subsystem. In both cases, the optimization problem is separable and each subsystem can be optimized independently. Equation (2) is rewritten in more compact form as

$$\begin{aligned}\widetilde{\Pi}(\mathbf{x},\mathbf{v};\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\rho}) &= F(\mathbf{x},\mathbf{v}) + \boldsymbol{\lambda}^{\top}\mathbf{f}(\mathbf{x},\mathbf{v}) + \boldsymbol{\mu}^{\top}[\mathbf{R}(\mathbf{x},\mathbf{v}) - \boldsymbol{\sigma}] \\ &\quad + \boldsymbol{\rho}^{\top}\mathbf{h}(\mathbf{x},\mathbf{v}),\end{aligned} \tag{3}$$

where $\boldsymbol{\sigma} \geq 0$, $\mathbf{v}$ represents $\mathbf{u}$ and $\mathbf{s}$ and $\mathbf{h}(\mathbf{x},\mathbf{v})$ denotes all $\mathbf{g}^{(i)} - \mathbf{s}^{(i)}$, $\boldsymbol{\rho}$ is a Lagrange multiplier vector of the same dimension as $\mathbf{g}$, $\boldsymbol{\mu}$ is an r vector including all Lagrange multipliers, and $\boldsymbol{\lambda}$ is an n vector including all Lagrange multipliers.

The Kuhn–Tucker theory of nonlinear programming [9] implies that if $\Pi(\mathbf{x},\mathbf{v})$ has a critical point at $(\mathbf{x}^0,\mathbf{v}^0)$ such that the constraint equations in (1), are satisfied, and if the rank of

$$\left[\left(\frac{\partial\mathbf{f}}{\partial\mathbf{y}}\right)^{\top}\left(\frac{\partial\mathbf{R}}{\partial\mathbf{y}}\right)^{\top}\left(\frac{\partial\mathbf{h}}{\partial\mathbf{y}}\right)^{\top}\right]$$

is full and equals the rank of

$$\left[\left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)^\top \left(\frac{\partial \mathbf{R}}{\partial \mathbf{y}}\right)^\top \left(\frac{\partial \mathbf{h}}{\partial \mathbf{y}}\right)^\top \left(\frac{\partial \Pi}{\partial \mathbf{y}}\right)^\top\right], \qquad (4)$$

where

$$\mathbf{y} \equiv \begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix}$$

at $(\mathbf{x}^0, \mathbf{v}^0)$, then a set of unique Lagrange multipliers $\boldsymbol{\lambda}^0$, $\boldsymbol{\mu}^0$ and $\boldsymbol{\rho}^0$ exist at the critical point. The necessary conditions for a critical point (local minimum) are

$$\frac{\partial \widetilde{\Pi}}{\partial \mathbf{x}} = \frac{\partial \widetilde{\Pi}}{\partial \mathbf{v}} = 0, \quad \mu_i R_i = 0, \quad \mathbf{R} \geq 0, \quad \boldsymbol{\mu} \leq 0, \quad (5)$$

$$\frac{\partial \widetilde{\Pi}}{\partial \boldsymbol{\lambda}} = \mathbf{f}^\top = 0, \qquad \frac{\widetilde{\Pi}}{\partial \boldsymbol{\rho}} = \mathbf{h}^\top = 0. \qquad (6)$$

If $\Pi(\mathbf{x}, \mathbf{v})$ is convex, if $f_i(\mathbf{x}, \mathbf{v})$ and $h_i(\mathbf{x}, \mathbf{v})$ are convex for λ_i^0 and ρ_i^0 positive, or if $f_i(\mathbf{x}, \mathbf{v})$, $h_i(\mathbf{x}, \mathbf{v})$, $R_i(\mathbf{x}, \mathbf{v})$ are concave for $\lambda_i^0, \rho_i^0, \mu_i^0$ negative, and the above necessary conditions are satisfied, then $\Pi(\mathbf{x}^0, \mathbf{v}^0)$ is the absolute minimum of (1) and $\widetilde{\Pi}$ has a global saddle point at $(\mathbf{x}^0, \mathbf{v}^0)$; that is,

$$\begin{aligned} \widetilde{\Pi}(\mathbf{x}, \mathbf{v}; \boldsymbol{\lambda}^0, \boldsymbol{\mu}^0, \boldsymbol{\rho}^0) &\geq \widetilde{\Pi}(\mathbf{x}^0, \mathbf{v}^0; \boldsymbol{\lambda}^0, \boldsymbol{\mu}^0, \boldsymbol{\rho}^0) \\ &\geq \widetilde{\Pi}(\mathbf{x}^0, \mathbf{v}^0; \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\rho}) \end{aligned}$$

for all $\mathbf{x}$, $\mathbf{v}$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, and $\boldsymbol{\rho}$. These conditions can be relaxed to local convexity and concavity such that only a local minimum and saddle point are assured.

The *nonfeasible gradient controller* of L.C. Lasdon and J.D. Schoeffler [11] has the following form: Given (1), suppose that

a) $\widetilde{\Pi}$ has a global saddle point at $(\mathbf{x}^0, \mathbf{v}^0; \boldsymbol{\lambda}^0, \boldsymbol{\mu}^0, \boldsymbol{\rho}^0)$; and
b) for any given $\boldsymbol{\rho}$, a finite constrained (unique) minimum (constrained by $\mathbf{f}$ and $\mathbf{R}$) exists.

Then the iterative procedure given by

$${}^{i+1}\boldsymbol{\rho} = {}^{i}\boldsymbol{\rho} + \Delta\boldsymbol{\rho},$$

where

$$\Delta\boldsymbol{\rho} = +\kappa \mathbf{h}(\mathbf{x}^*, \mathbf{v}^*), \quad \text{with } \kappa > 0,$$

will converge to $\boldsymbol{\rho}^0$ and the absolute minimum of (1). Note that a local saddle point can replace a), then the initial guess on $\boldsymbol{\rho}$ must be within this saddle region. However, then the algorithm leads only to a local minimum. This Lasdon gradient controller can be considered as a variant of the modified *Arrow–Hurwicz gradient method* of K. Arrow, L. Hurwicz and H. Uzawa [1].

The *feasible gradient controller* of C.B. Brosilow et al. [6] has the following form: Given (1), suppose that

a) a finite minimum exists at $(\mathbf{x}^0, \mathbf{v}^0)$; and
b) all the conditions of (5) and (6) are fulfilled except for $\partial\widetilde{\Pi}/(\partial \mathbf{s}) = 0$, (where $\mathbf{v}$ denotes all $\mathbf{s}$ and $\mathbf{u}$).

Then the iterative procedure given by

$${}^{i+1}\mathbf{s} = {}^{i}\mathbf{s} + \Delta\mathbf{s},$$

where

$$\Delta\mathbf{s} = -\kappa \left(\frac{\partial \widetilde{\Pi}}{\partial \mathbf{s}}\right), \quad \text{with } \kappa > 0,$$

will converge to $\mathbf{s}^0 = \mathbf{x}^0$ and the minimum of (1).

The good choice of κ is important for the gradient calculations. Then at the second level of the feasible method, we may write ([3, p. 142]) that

$$\mathrm{d}\boldsymbol{\Pi}^* = \frac{\partial \widetilde{\Pi}}{\partial \mathbf{s}} \mathrm{d}\mathbf{s} = -\kappa \frac{\partial \widetilde{\Pi}}{\partial \mathbf{s}} \left(\frac{\partial \widetilde{\Pi}}{\partial \mathbf{s}}\right)^\top, \quad \kappa > 0.$$

An estimate of the expected improvement is written as $-\alpha\overline{\boldsymbol{\Pi}}^*$, $\alpha > 0$, where α is usually 10% or so. Then

$$\kappa = \frac{\alpha \widetilde{\Pi}^*}{(\partial\widetilde{\Pi}/\partial \mathbf{s})\,(\partial\widetilde{\Pi}/\partial \mathbf{s})^\top}. \qquad (7)$$

In the case of nonfeasible decomposition a similar equation may be obtained [3]:

$$\kappa = \frac{\alpha \widetilde{\Pi}^*}{\mathbf{g}^\top \mathbf{g}}. \qquad (8)$$

Note that $\Delta\mathbf{s}$ and $\Delta\boldsymbol{\rho}$ become singular at the optimum if (7) and (8) are used, respectively, and therefore these values of $\Delta\mathbf{s}$ and $\Delta\boldsymbol{\rho}$ are not appropriate to obtain exact solutions.

There is also the possibility to apply a Newton–Raphson controller both for the feasible and for the nonfeasible method in the second level (cf. in this context [3, p. 173]).

For instance examining (5) and (6), it is obvious that the only necessary condition not satisfied by the subsystems is $\mathbf{g} = 0$ in the nonfeasible decomposition method. Thus the Newton–Raphson method has as task to solve $\mathbf{g} = 0$ by an iterative method at the second level.

Note that the main characteristic of the aforementioned methods, i. e. the decomposition into subsystems and the separable optimization applies also to nonsmooth convex or nonconvex optimization problems.

Large Cable Structures

Here a possibility offered in structural analysis by the multilevel optimization algorithms is presented. Certain subproblems do not contain inequalities, i. e. are bilateral, and thus they can be treated by the available classical (i. e. based only on inequalities) FEM programs.

In the majority of *cable structures* the number of cables and nodes is large, and so an optimization problem with a large number of unknowns and constraints must be solved. Here, a multilevel optimization technique suitable for the solution of this kind of optimization problem is proposed. The initial optimization problem is decomposed into a number of subproblems. In the 'first level' of the calculation, each subproblem is optimized separately, and in the 'second level' the solutions of these subproblems are combined to yield the overall optimum.

It is interesting to note that some of these subproblems constitute minimization problems without inequality constraints (corresponding to classical bilateral structures), and the algorithms for their numerical treatment are much faster. The initial problem is decomposed into two subproblems: the first involves only the displacement terms and corresponds to a structure resulting from the given one by considering that all the cables act as bars (capable of having compressive forces), and the second, including only the slackness terms, corresponds to a hypothetical slack structure. In order to perform the decomposition, the potential energy of the structure is written in the form

$$\Pi(\mathbf{u},\mathbf{v}) = \Pi'(\mathbf{u}) + \Pi''(\mathbf{v}) + \mathbf{u}^\top \mathbf{G}\mathbf{K}_0\mathbf{v}, \tag{9}$$

where

$$\Pi'(\mathbf{u}) = \frac{1}{2}\mathbf{u}^\top \mathbf{K}\mathbf{u} - \mathbf{u}^\top(\mathbf{G}\mathbf{K}_0\mathbf{e}_0 + \mathbf{p}) \tag{10}$$

and

$$\Pi''(\mathbf{v}) = \frac{1}{2}\mathbf{v}^\top \mathbf{K}_0\mathbf{v}^\top + \mathbf{v}^\top(\mathbf{a} - \mathbf{K}_0\mathbf{e}_0). \tag{11}$$

In the above equations $\mathbf{u}$, $\mathbf{v}$, $\mathbf{p}$, $\mathbf{e}_0$ are the displacements, slackness, loading and initial strain vectors respectively, $\mathbf{K}_0$ is the natural stiffness matrix, $\mathbf{K}$ is the stiffness matrix of the assembled structure and $\mathbf{G}$ is the equilibrium matrix. Introducing the variable $\mathbf{w}$ the minimization problem (9) takes the form

$$\min \Pi(\mathbf{u},\mathbf{v},\mathbf{w}) = \Pi'(\mathbf{u}) + \Pi''(\mathbf{v}) + \mathbf{u}^\top \mathbf{G}\mathbf{K}_0\mathbf{w}.$$

The Lagrangian of this problem is

$$\Pi_1(\mathbf{u},\mathbf{v},\mathbf{w}) = \Pi(\mathbf{u},\mathbf{v},\mathbf{w}) + \boldsymbol{\rho}^\top(\mathbf{v} - \mathbf{w}),$$

where $\boldsymbol{\rho}$ is the vector of the Lagrange multipliers. The decomposition can be performed by means of two methods: the nonfeasible gradient controller method of Lasdon and Schoeffler and the feasible gradient controller method of Brosilow, Lasdon and Pearson [11]. In the nonfeasible gradient controller method the value of $\boldsymbol{\rho}$ is supposed to be constant in the first level, say $\boldsymbol{\rho}_1$, and the minimization problem decomposes into the two subproblems

$$\min_{\mathbf{u},\mathbf{w}}\{\Pi'(\mathbf{u}) + \mathbf{u}^\top \mathbf{G}\mathbf{K}_0\mathbf{w} - \boldsymbol{\rho}_1^\top \mathbf{w}\}$$

and

$$\min_{\mathbf{v}}\left\{\Pi''(\mathbf{v}) + \boldsymbol{\rho}_1^\top \mathbf{v} \colon\ \mathbf{v} + \mathbf{p} \geq 0\right\}.$$

After performing the optimization, the values of $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$, e. g. $\mathbf{u}_1$, $\mathbf{v}_1$ and $\mathbf{w}_1$, result. It is obvious that $\mathbf{v}_1 \neq \mathbf{w}_1$. The task of the second level is to estimate a new value of $\boldsymbol{\rho}$, e. g. $\boldsymbol{\rho}_2$ by means of the equation

$$\boldsymbol{\rho}_2 = \boldsymbol{\rho}_1 + \kappa(\mathbf{v}_1 - \mathbf{w}_1), \quad \kappa > 0,$$

where κ is a properly chosen constant (see, e. g., [11]), and to transmit this value to the first level. The optimization is performed again, new values $\mathbf{u}_2$, $\mathbf{v}_2$ and $\mathbf{w}_2$ result, etc., until the differences $\mathbf{v}_i - \mathbf{w}_i$ are made negligible. The algorithm converges in a finite number of steps, provided that the minima exist [11].

In the feasible gradient controller method, the value of $\mathbf{w}$ is taken as constant in the first level, e. g. $\mathbf{w}_1$, and thus the initial problem decomposes into the two subproblems

$$\min_{\mathbf{u}}\{\Pi'(\mathbf{u}) + \mathbf{u}^\top \mathbf{G}\mathbf{K}_0\mathbf{w}_1\}$$

and

$$\min_{\mathbf{v},\boldsymbol{\rho}}\left\{\Pi_1''(\mathbf{v}) + \boldsymbol{\rho}^\top(\mathbf{v}-\mathbf{w}_1''):\ \mathbf{v}+b \geq 0\right\}.$$

As a result of the optimization, the values of $\mathbf{u}$, $\mathbf{v}$ and $\boldsymbol{\rho}$, e. g. $\mathbf{u}_1$, $\mathbf{v}_1$ and $\boldsymbol{\rho}_1$ are calculated. By means of the second level a new value of $\mathbf{w}$, e. g. $\mathbf{w}_2$, is estimated and transmitted to the first level. This value is given by the equation

$$\mathbf{w}_2 = \mathbf{w}_1 - \kappa\left(\frac{\partial \Pi_1(\mathbf{u},\mathbf{v},\mathbf{w})}{\partial \mathbf{w}}\right)_{\mathbf{w}=\mathbf{w}_1}, \quad \kappa > 0,$$

where κ is a properly chosen constant (see, e. g., [11]). The optimization yields a new set of values $\mathbf{u}_2$, $\mathbf{v}_2$ and $\boldsymbol{\rho}_2$ and the procedure is continued until the difference between the consecutive values of vector $\mathbf{w}$ becomes sufficiently small.

For numerical applications the reader is referred to [20].

Large Elastoplastic Structures

We consider here the holonomic *plasticity* model [13], (extension to nonholonomic plasticity problems is straightforward) described by the following equations:

$$\begin{aligned}
&\mathbf{e} = \mathbf{F}_0\mathbf{s},\\
&\mathbf{e} = \mathbf{e}_0 + \mathbf{e}_E + \mathbf{e}_P,\\
&\mathbf{e}_P = \mathbf{N}\boldsymbol{\lambda},\\
&\boldsymbol{\phi} = \mathbf{N}^\top\mathbf{s} - \mathbf{k},\\
&\boldsymbol{\lambda} \geq 0, \quad \boldsymbol{\phi} \leq 0, \quad \boldsymbol{\phi}^\top\boldsymbol{\lambda} = 0,
\end{aligned}$$

where $\mathbf{F}_0$ is the natural flexibility matrix of the structure, $\mathbf{e}$ the respective strain vector consisting of three parts, the initial strain $\mathbf{e}_0$, the elastic strain $\mathbf{e}_E$ and the plastic strain $\mathbf{e}_P$, $\boldsymbol{\lambda}$ are the plastic multipliers vector, $\boldsymbol{\phi}$ the yield functions, $\mathbf{N}$ is the matrix of the gradients of the yield functions with respect to the stresses and $\mathbf{k}$ is a vector of positive constants. The potential energy of the structure is written in the form

$$\Pi(\mathbf{u},\boldsymbol{\lambda}) = \Pi'(\mathbf{u}) + \Pi''(\boldsymbol{\lambda}) - \mathbf{u}^\top\mathbf{G}\mathbf{K}_0\mathbf{N}\boldsymbol{\lambda}$$

where

$$\begin{aligned}
&\Pi'(\mathbf{u}) = \tfrac{1}{2}\mathbf{u}^\top\mathbf{K}\mathbf{u} - \mathbf{e}_0^\top\mathbf{K}_0\mathbf{G}^\top\mathbf{u} - \mathbf{p}^\top\mathbf{u},\\
&\Pi''(\boldsymbol{\lambda}) = \tfrac{1}{2}\boldsymbol{\lambda}^\top\mathbf{N}^\top\mathbf{K}_0\mathbf{N}\boldsymbol{\lambda} + \mathbf{e}_0^\top\mathbf{K}_0\mathbf{N}\boldsymbol{\lambda} - \mathbf{k}\boldsymbol{\lambda}.
\end{aligned}$$

Again, $\mathbf{K}$ is the stiffness matrix of the structure and $\mathbf{K}_0$ is the inverse of $\mathbf{F}_0$.

The solution of the problem can be obtained by minimizing the potential energy of the structure:

$$\min\{\Pi(\mathbf{u},\boldsymbol{\lambda}):\ \boldsymbol{\lambda} \geq 0\}. \tag{12}$$

By introducing a new variable $\mathbf{w}$, (12) takes the form

$$\begin{aligned}
\min\Big\{\Pi(\mathbf{u},\boldsymbol{\lambda},\mathbf{w}) = {}&\Pi'(\mathbf{u}) + \Pi''(\boldsymbol{\lambda})\\
&-\mathbf{u}^\top\mathbf{G}\mathbf{K}_0\mathbf{N}\mathbf{w}:\ \mathbf{w} = \boldsymbol{\lambda},\ \boldsymbol{\lambda} \geq 0\Big\}.
\end{aligned} \tag{13}$$

As in the previous section, the decomposition can be performed by the two methods of the feasible and the nonfeasible gradient controller respectively. For the sake of brevity only the nonfeasible gradient method will be shown here. The Lagrangian of (13) is first considered

$$\Pi(\mathbf{u},\boldsymbol{\lambda},\mathbf{w}) = \Pi(\mathbf{u},\boldsymbol{\lambda},\mathbf{w}) + \boldsymbol{\rho}^\top(\boldsymbol{\lambda}-\mathbf{w})$$

and the minimization problem is decomposed in the following two subproblems

$$\min_{\mathbf{u},\mathbf{w}}\left\{\Pi'(\mathbf{u}) - \mathbf{u}^\top\mathbf{G}\mathbf{K}_0\mathbf{N}\mathbf{w} - \boldsymbol{\rho}^\top\mathbf{w}\right\} \tag{14}$$

and

$$\min_{\boldsymbol{\lambda}}\left\{\Pi_1''(\boldsymbol{\lambda}) + \boldsymbol{\rho}^\top\boldsymbol{\lambda}:\ \boldsymbol{\lambda} \geq 0\right\}. \tag{15}$$

In the first step it is supposed that the value of $\boldsymbol{\rho}$ is constant (say $\boldsymbol{\rho}_1$) and we take as a result from (14) and (15) the values $\mathbf{u}_1$, $\boldsymbol{\lambda}_1$ and $\mathbf{w}_1$. Obviously $\boldsymbol{\lambda}_1 \neq \mathbf{w}_1$. Then the second level controller estimates the new value of $\boldsymbol{\rho}$ from the equation

$$\boldsymbol{\rho}_2 = \boldsymbol{\rho}_1 + \kappa(\boldsymbol{\lambda}_1 - \mathbf{w}_1), \quad \kappa > 0,$$

and transmits it to the first level, and the procedure is continued until the differences $\boldsymbol{\lambda}_i - \mathbf{w}_i$ become appropriately small.

The same procedure can be applied also to holonomic models including hardening and to nonholonomic plasticity models [13].

Validation and Improvements of Simplified Models

In mechanics and engineering sciences as well as in economy, simplified models are often considered for the treatment of complicated problems, e. g. concerning the calculation of stresses in complex structures. In these models it is assumed that certain quantities do not influence considerably the solution of the problem. By means of the multilevel decomposition, a method which permits the *validation* of these models and the improvement of their accuracy can be developed. This idea is explained in the sequel.

A. Consider a large structure involving also some cables and assume that due to the pretension of the cables the structure is calculated as if the cables are rods, i. e. by ignoring the fact that a cable may become slack and then it has zero stresses. Then in the equations (9)–(11) $\mathbf{v} = 0$ and the solution of the minimum problem is obtained by solving an unconstrained minimization problem, i. e. by a linear system solver. In order to check whether the solution of the simplified model is close to the solution of the initial problem, in which some cables, say r, may become *slack*, i. e. $v_i > 0$, $i = 1, \ldots, r$, it is enough to verify whether the second level controller which gives a value of the slackness of the cables causes a significant change in the solution of the first level problem which corresponds to the simplified structure. Also the algorithm offers an improvement of the solution of the simplified model.

B. Here, the investigation of the mutual influence of two subsystems is presented. Consider two substructures connected together, for instance a cylindrical shell with a hemispherical shell covering the one end of the cylinder. The solution of the whole linear elastic structural compound minimizes, for a given external loading, the potential (or the complementary) energy of the whole structure. Let $\mathbf{x}_1$ (respectively, $\mathbf{x}_2$) be the variables of the cylindrical (respectively, the hemispherical) shell and let $\mathbf{z}$ be the common variables at the contact line which are common in both structures. In order to decompose the potential energy into two minimum problems, one containing the unknowns of the cylindrical shell and the other of the hemispherical shell, the common variables for the cylindrical (respectively, hemispherical) shell are denoted by $\mathbf{z}_1$ (respectively, $\mathbf{z}_2$) and thus the initial problem

$$\min_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{z}} \{\Pi(\mathbf{x}_1,\mathbf{x}_2,\mathbf{z}) = \Pi_1(\mathbf{x}_1,\mathbf{z}) + \Pi_2(\mathbf{x}_2,\mathbf{z})\}$$

is written as

$$\min_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{z}_1,\mathbf{z}_2} \{\Pi_1(\mathbf{x}_1,\mathbf{z}_1) + \Pi_2(\mathbf{x}_2,\mathbf{z}_2)\colon\ \mathbf{z}_1 - \mathbf{z}_2 = 0\}\,.$$

Here Π_1 (respectively, Π_2) denotes the potential or the complementary energy of the cylindrical (respectively, the hemispherical) shell. Thus it can be tested by the nonfeasible controller method how the difference $\mathbf{z}_1 - \mathbf{z}_2$ influences the solution of the problem. The procedure is similar in the case of elastoplastic structures with the difference that the minimum is constrained by inequalities.

The above procedure may find applications in estimating the influence of saddles on pipelines of rigidity rings on long tubes etc.

C. Note that in all the above cases the Lagrange multipliers have a precise meaning: they correspond in the sense of energy to the chosen coordinating variables, i. e., if the coordinating variables are stresses (respectively, strains) or forces (respectively, displacements) then the coordinating Lagrange multipliers are strains (respectively, stresses) or displacements (respectively, forces). Thus the feasible and the nonfeasible decomposition method have a precise mechanical meaning. In the first case the Lagrange multipliers, i. e. the strains (respectively, the stress) are controlled while in the second one the coordinating variables, i. e. the stress (respectively, the strain) of the links between the two substructures are controlled, in order to achieve the position of equilibrium of the whole structure.

D. Some of the resulting substructures may have a known analytical solution. Then this fact facilitates the calculation and may be applied as a test for the accuracy of the resulting solution via a numerical technique, e. g. by the FEM model. The procedure is described in [24].

E. The multilevel decomposition method can be used also as estimator of the sensitivity of the final solution to small changes of the system to be optimized [24]. This method may be used for example in estimating how a partial change in a structure influences the stress and strain field of the structure without solving twice the structure.

Decomposition Algorithms for Nonconvex Minimization Problems

In unilateral contact problems with friction, Panagiotopoulos proposed in 1975 an algorithm [18] called later PANA-algorithm for the decomposition of the *quasivariational inequality* problems into two classical variational inequality problems which are equivalent to two minimization problems. Analogous decomposition methods of complicated problems using an analogous to [18] fixed point procedure can be applied to the treatment of much more complicated problems today involving nonconvex energy functions. This section is devoted to the study of multilevel decomposition algorithms for problems belonging to the general framework of the substationarity problems.

It is known that the equilibrium of an elastic body Ω in adhesive contact with a support Γ is governed by the following problem [17,21]: Find $u \in V$ such as to satisfy the hemivariational inequality

$$\alpha(u, v-u) + \int_{\Gamma} j_N^0(u_N, v_N - u_N)\mathrm{d}\Gamma + \int_{\Gamma} j_T^0(u_T, v_T - u_T)\mathrm{d}\Gamma \geq (f, v-u), \quad \forall v \in V. \tag{16}$$

Here u, v are the displacement fields, f are all the applied forces, (f, v) – usually a L^2 internal product – is the work of the applied forces, $\alpha(u, v)$ is the elastic strain energy which is usually a coercive form, j_N (respectively, j_T) denote the nonconvex, locally Lipschitz generally nonsmooth energy density functions of the adhesive forces in the normal (respectively, the tangential) direction to the interface Γ. It is assumed that the normal adhesive action is independent of the tangential adhesive action. Moreover, j_N^0, j_T^0 denote the directional derivative in the sense of Clarke [7], and u_N, v_N (respectively, u_T, v_T) denote the normal (respectively, tangential) component of the displacement with respect to Γ. The solution of the above problem can be obtained in most cases of practical interest (cf. [21]) under certain mild hypotheses which guarantee this equivalence, by solving the substationarity problem

$$0 \in \bar{\partial} I(u) = \bar{\partial}\left\{\frac{1}{2}\alpha(u,u) + \int_{\Gamma} j_N(u_N)\mathrm{d}\Gamma + \int_{\Gamma} j_T(u_T)\mathrm{d}\Gamma - (f,u)\right\},$$

where $\bar{\partial}$ denotes the generalized gradient of Clarke.

In engineering problems the nonconvex superpotentials (cf. e.g. [16]) j_N and j_T are not independent but they depend j_N (respectively, j_T) on the vectors S_T (respectively, S_N), where S_T, S_N are the reactions corresponding to u_T, u_N respectively. In this case a hemivariational inequality cannot be formulated. In order to solve this problem numerically one may apply the following procedure: In the first step it is assumed that S_N is given, say, $S_N^{(0)}$ and the problem ($S_N^{(0)}$ enters with its work into $(f_1^{(0)}, u)$)

$$0 \in \bar{\partial}\left\{\frac{1}{2}\alpha(u,u) + \int_{\Gamma} j_T(S_N^{(0)}, u_T)\mathrm{d}\Gamma - (f_1^{(0)}, u)\right\} \tag{17}$$

is solved. The above problem yields a value of S_T, say $S_T^{(1)}$. Then the problem

$$0 \in \bar{\partial}\left\{\frac{1}{2}\alpha(u,u) + \int_{\Gamma} j_N(S_T^{(1)}, u_N)\mathrm{d}\Gamma - (f_2^{(1)}, u)\right\} \tag{18}$$

is solved ($S_T^{(1)}$ enters with its work into $(f_2^{(1)}, u)$) yielding a new value of S_N, say $S_N^{(1)}$, and so on until the differences $\| S_N^{(i)} - S_N^{(i+1)} \|$ and $\| S_T^{(i)} - S_T^{(i+1)} \|$ at each point of the discretized interface Γ become appropriately small. Here $\| \cdot \|$ denotes the $\mathbf{R}^3$-norm because the values are checked pointwise. The first (respectively, second) problem with $j_N = 0$ (respectively, with $j_T = 0$) corresponds to the first level (respectively, to the second level). Applications of the above procedure can be found in [15,20,21].

Structures with Fractal Interfaces

In this section the attention is focused on the fractal geometry of interfaces where their behavior is modeled by means of an appropriate nonmonotone contact and friction mechanism. The interfaces of fractal geometry are analyzed here as a sequence of classical interface subproblems. These classical subproblems result from the consideration of the *fractal interface* as the unique 'fixed point' of a given *iterative function system* (IFS), which consists of N contractive mappings w_i: $\mathbf{R}^2 \to \mathbf{R}^2$ with contractivity factors $0 \leq s_i < 1$, $i = 1, \ldots, N$ [2]. According to this procedure, a fractal set A is the 'fixed point' of a transformation W i. e.

$$A = W(A) = \bigcup_{i=1}^{N} W_i(A),$$

where W_i is defined

$$W_i(B) = \{w_i(x)\colon\ x \in B\}\,, \quad \forall B \in H(\mathbb{R}^2).$$

Generally a *fractal set* A is given by the relation:

$$A = \lim_{n \to \infty} W^{(n)}(B), \quad \forall B \in H(\mathbb{R}^2),$$

where $H(\mathbf{R}^2)$ is the space of all compact subsets of $\mathbf{R}^2$. Thus each level corresponds to a classical geometry approximating the fractal geometry. Within each level a new optimization problem is solved with the new data. Thus the multilevel character of the optimization problem results from the necessity to take into account the fractal geometry.

In the sequel a linear elastic structure occupying a subset Ω of $\mathbf{R}^3$ is considered. In its undeformed state the structure has a boundary Γ which is decomposed into two mutually disjoint parts Γ_U and Γ_F. It is assumed that on Γ_U(respectively, Γ_F) the displacements (respectively, the tractions) are given. In the structure Ω some cracks with interfaces Φ of fractal type are formed. These cracks in brittle materials frequently propagate along one or more irregular ways. In this case the fracture system may be considered to be a cluster of branches propagating in such a way that new branches in the $n + 1$ step are successively created from a former branch at the n step. In other words the fracture system can be modeled by an IFS procedure. Regarding now the boundary conditions on Φ, it is assumed that nonmonotone, possibly multivalued laws describe the behavior of each interface in the normal and tangential directions. More specifically, it is assumed that the following boundary conditions hold:

$$-S_N \in \overline{\partial} j_N(u_N, x),$$
$$-S_T \in \overline{\partial} j_T(u_T, x).$$

Then according to the previous section, an equilibrium position of Ω is characterized by the hemivariational inequality (16).

In this case, where the fractured body Ω with fractal interfaces Φ is studied, it is necessary to substitute in (16) the domain Γ with Φ. As it has been mentioned above, Φ is the fixed point of a given transformation denoted by W, i. e.

$$\Phi = W\Phi,$$
$$\Phi^{(n+1)} = W\Phi^{(n)},$$
$$\Phi^{(n)}_{n\to\infty} \to \Phi.$$

Thus, for each approximation $\Phi^{(n)}$ of the fractal interface Φ a structure $\Omega^{(n)}$ must be solved. Since $\Phi^{(n)}$ is an interface set with classical geometry the solutions $u^{(n)}$ and $\sigma^{(n)}$ (where $u^{(n)}$ and $\sigma^{(n)}$ are the corresponding displacement and stress fields) are obtained using numerical procedures for the solution of (17) and (18). This procedure is repeated several times by increasing n; at the limit $n \to \infty$, $u^{(n)}$ and $\sigma^{(n)}$ give the solution of the fractal interface problem.

See also

- ▶ Bilevel Fractional Programming
- ▶ Bilevel Linear Programming
- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Bilevel Programming
- ▶ Bilevel Programming: Applications
- ▶ Bilevel Programming: Applications in Engineering
- ▶ Bilevel Programming: Implicit Function Approach
- ▶ Bilevel Programming: Introduction, History and Overview
- ▶ Bilevel Programming in Management
- ▶ Bilevel Programming: Optimality Conditions and Duality
- ▶ Multilevel Methods for Optimal Design
- ▶ Stochastic Bilevel Programs

References

1. Arrow KJ, Hurwicz L, Uzawa H (1985) Studies in linear and nonlinear programming. Stanford Univ Press, Palo Alto
2. Barnsley M (1988) Fractals everywhere. Acad Press, New York

3. Bauman EJ (1971) Trajectory decomposition. In: Leondes CT (ed) Optimization methods for large scale systems with applications. McGraw-Hill, New York
4. Bertsekas DP, Tsitsiklis JN (1989) Parallel and distributed computation: numerical methods. Prentice-Hall, Englewood Cliffs, last edition: Athena Sci, Belmont Mass, 1997.
5. Bolza O (1904) Lectures on the calculation of variations. Chicago, Chicago
6. Brosilow CB, Lasdon LS, Pearson JD (1965) A multi-level technique for optimization. In: Proc Joint Autom Control Conf. Rensselaer Polytech Inst, Troy
7. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, New York
8. Demyanov VF, Stavroulakis GE, Polyakova LN, Panagiotopoulos PD (1996) Quasidifferentiability and nonsmooth modelling in mechanics, engineering and economics. Kluwer, Dordrecht
9. Hadley G (1964) Non-linear and dynamic programming. Addison-Wesley, Reading
10. Hamel G (1967) Theoretische Mechanik. Springer, Berlin
11. Lasdon LC, Schoeffler JD (1965) A multi-level technique for optimization. In: Proc Joint Autom Control Conf. Rensselaer Polytech Inst, Troy
12. Leondes CT (ed) (1968) Advances in control systems. Theory and applications. Acad Press, New York
13. Maier G (1968) A quadratic programming approach for certain classes of non linear structural problems. Meccanica 2:121–130
14. Migdalas A, Pardalos PM (1996) Special issue on hierarchical and bilevel programming. J Global Optim 8(3)
15. Mistakidis ES, Stavroulakis GE (1998) Nonconvex optimization in mechanics. Algorithms, heuristics and engineering applications by the FEM. Kluwer, Dordrecht
16. Moreau JJ, Panagiotopoulos PD, Strang G (eds) (1988) Topics in nonsmooth mechanics. Birkhäuser, Basel
17. Naniewicz Z, Panagiotopoulos PD (1995) Mathematical theory of hemivariational inequalities and applications. M. Dekker, New York
18. Panagiotopoulos PD (1975) A nonlinear programming approach to the unilateral contact – and friction – boundary value problem in the theory of elasticity. Ingen Archiv 44:421–432
19. Panagiotopoulos PD (1976) A variational inequality approach to the inelastic stress-unilateral analysis of cable structures. Comput Structures 6:133–139
20. Panagiotopoulos PD (1985) Inequality problems in mechanics and applications. Convex and nonconvex energy Functions. Birkhäuser, Basel, Russian Translation: MIR, Moscow 1989
21. Panagiotopoulos PD (1993) Hemivariational inequalities. Applications in mechanics and engineering. Springer, Berlin
22. Panagiotopoulos PD (1995) Modelling of nonconvex nonsmooth energy problems. Dynamic hemivariational inequalities with impact effects. J Comput Appl Math 63:123–138
23. Panagiotopoulos PD (1995) Variational principles for contact problems including impact phenomena. In: Raous M (ed) Contact Mechanics. Plenum, New York
24. Panagiotopoulos PD, Mistakidis ES, Stavroulakis GE, Panagouli OK (1998) Multilevel optimization methods in mechanics. In: Multilevel Optimization: Algorithms, Complexity and Applications. Kluwer, Dordrecht, pp 51–90
25. Rockafellar RT (1979) La théorie des sous-gradients et ses applications à l'optimization. Fonctions convexes et nonconvexes. Les Presses de l'Univ Montréal, Montréal
26. Schoeffler JD (1971) Static multilevel systems. In: Leondes CT (ed) Optimization methods for large scale systems with applications. McGraw-Hill, New York
27. Wismer DA (ed) (1971) Optimization methods for large scale systems with applications. McGraw-Hill, New York

Multi-objective Combinatorial Optimization

MOCO

Jacques Teghem
Lab. Math. & Operational Research Fac.,
Polytechn. Mons, Mons, Belgium

MSC2000: 90C29, 90C27

Article Outline

Keywords

Multi-objective programming; Combinatorial optimization

It is well known that, on the one hand, *combinatorial optimization* (CO) provides a powerful tool to formulate and model many optimization problems, on the other hand, a multi-objective (MO) approach is often a realistic and efficient way to treat many real world applications. Nevertheless, until recently, Multi-objective combinatorial optimization (MOCO) did not receive much attention in spite of its potential applications. One of the reason is probably due to specific difficulties of MOCO models. We can distinguish three main difficulties. The first two are the same as those existing for multi-objective integer linear programming (MOILP) problem (cf. ► Multi-objective Integer Linear Programming), i. e.

- the number of *efficient solutions* may be very large;
- the nonconvex character of the feasible set requires to device specific techniques to generate the so-called 'nonsupported' efficient solutions (cf. ► Multi-objective Integer Linear Programming).

A particular single CO problem is characterized by some specificities of the problem, generally a special form of the constraints; the existing methods for such problem use these specificities to define efficient ways to obtain an optimal solution. For MOCO problem, it appears interesting to do the same to obtain the set of efficient solutions. Consequently, and contrary to what is often done in MOLP and MOILP methods, a third difficulty is to elaborate methods avoiding to introduce additional constraints so that we preserve during all the procedure the particular form of the constraints.

The general form of a MOCO problem is

$$(\mathrm{P})\quad \begin{cases} '\min_{X\in S}' & z_k(X) = c_k X, \\ & \quad k = 1,\dots,K, \\ \text{where} & S = D \cap B^n \\ \text{with} & X(n\times 1), \\ & B = \{0,1\} \end{cases}$$

and D is a specific polytope characterizing the CO problem: assignment problem, knapsack problem, traveling salesman problem, etc.

There exists several surveys on MOCO; some are devoted to specific problems (i. e., the particular form of D): the shortest path problem [8], transportation networks [2], and the scheduling problem [6,7]; the survey [9] is more general examining successively the literature on MO assignment problems, knapsack problems, network flow problems, traveling salesman problems, location problems, set covering problems.

In the present article we put our attention on the existing methodologies for MOCO. First we examine how to determine the set $E(\mathrm{P})$ of all the efficient solutions and we distinguish three approaches: direct methods, two-phase methods and heuristic methods. Subsequently we analyse interactive approaches to generate a 'good compromise' satisfying the decision maker.

Generation of *E*(P)

Direct Methods

The first idea is to use intensively classical methods for single objective problem (P) existing in the literature to determine $E(\mathrm{P})$. Of course, each time a feasible solution is obtained the k values $z_k(X)$ are calculated and compared with the list $\widehat{E(\mathrm{P})}$ containing all the feasible solutions already obtained and non dominated by another generated feasible solution. Clearly, $\widehat{E(\mathrm{P})}$, called the *set of potential efficient solutions*, plays the role of the so-called 'incumbent solution' in single objective methods. At each step, $\widehat{E(\mathrm{P})}$ is updated and at the end of the procedure $E(\mathrm{P}) = \widehat{E(\mathrm{P})}$. Such extension of single objective method is specially designed for enumerative procedure based on a *branch and bound* approach. Unfortunately, in a MO framework, a node of the branch and bound tree is less often fathomed than in the single objective case, so that logically such MO procedure is less efficient.

We describe below an example of such direct method, extending the well known Martello–Toth procedure, for the multi-objective *knapsack problem* formulated as

$$\begin{cases} '\max' \quad z_k(X) = \sum_{j=1}^{n} c_j^{(k)} x_j, \quad k = 1,\dots,K, \\ \sum_{j=1}^{n} w_j x_j \le W \quad x_j = (0,1). \end{cases}$$

The following typical definitions are used ($k = 1,\dots,K$):

- O_k: variables order according to decreasing values of c_j^k / w_j.
- $r_j^{(k)}$: the rank of variable j in order O_k.
- Θ: variables order according to increasing values of $\sum_{k=1}^{K} r_j^{(k)} / K$.

We assume that variables are indexed according to ordinal preference Θ.

At any node of the branch and bound tree, variables are set to 0 or 1; let B_0 and B_1 denote the index sets of variables assigned to the values 0 and 1, respectively. Let F be the index set of free variables which always follow, in the order Θ, those belonging to $B_1 \cup B_0$. If $i-1$ is the last index of fixed variables, we have $B_1 \cup B_0 = \{1, \dots, i-1\}$; $F = \{i, \dots, n\}$.
Initially, $i = 1$. Let

- $\overline{W} = W - \sum_{j\in B_1} w_j \geq 0$ be the leftover capacity of the knapsack.
- $\underline{Z} = \left(\underline{z}_k = \sum_{j\in B_1} c_j^{(k)}\right)_{k=1,\dots,K}$ be the criteria values vector obtained with already fixed variables. $\widehat{E(\mathrm{P})}$ contains nondominated feasible values $\underline{Z}$ and is updated at each new step.
  Initially, $\underline{z}_k = 0, \forall\, k$, and $\widehat{E(\mathrm{P})} = \emptyset$.
- $\overline{Z} = (\overline{z}_k)$ be the vector whose components are upper bounds of feasible values respectively for each objective at considered node. These upper bounds are evaluated separately, for instance as in the Martello–Toth method.
  Initially, $\overline{z}_k = \infty, \forall\, k$.

A node is fathomed in the following two situations:
i) if $\left\{j \in F\colon\ w_j < \overline{W}\right\} = \emptyset$; or
ii) $\overline{z}$ is dominated by $z^* \in \widehat{E(\mathrm{P})}$.
When the node is fathomed, the backtracking procedure is performed: a new node is build up by setting to zero the variable corresponding to the last index in B_1. Let t be this index:

$$B_1 \leftarrow B_1 \backslash \{t\},$$
$$B_0 \leftarrow (B_0 \cap \{1, \dots, t-1\}) \cup \{t\},$$
$$F \leftarrow \{t+1, \dots, n\}.$$

When the node is nonfathomed, a new node of the branch and bound tree is build up for next iteration, as follows:

- Define s to be the index variable such that
$$\max\left\{l \in F\colon\ \sum_{j=i}^{l} w_j < \overline{W}\right\}.$$
  If $w_i > \overline{W}$, set $s = i - 1$.
- If $s \geq i$:
$$B_1 \leftarrow B_1 \cup \{i, \dots, s\},$$
$$B_0 \leftarrow B_0,$$
$$F \leftarrow F\backslash\{i, \dots, s\}.$$

  If $s = i - 1$,
$$B_1 \leftarrow B_1 \cup \{r\},$$
$$B_0 \leftarrow B_0 \cup \{i, \dots, r-1\},$$
$$F \leftarrow F\backslash\{i, \dots, r\},$$
  with $r = \min\left\{j \in F\colon\ w_j < \overline{W}\right\}$.

The procedure stops when the initial node is fathomed and then $E(\mathrm{P}) = \widehat{E(\mathrm{P})}$. An illustration is given in [10].

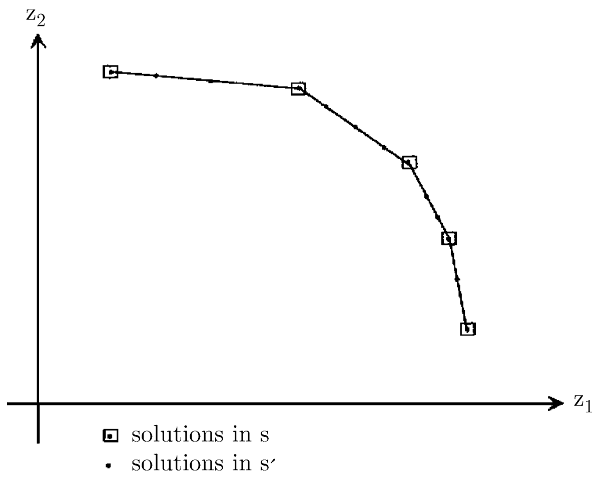
Two-Phase Method

Such an approach is particularly well designed for bi-objective MOCO problems. The first phase consists to determine the set $SE(\mathrm{P})$ of supported efficient solutions (see ► Multi-objective Integer Linear Programming). Let $S \cup S'$ be the list of supported efficient solutions already generated; S is initialized with the two efficient optimal solutions respectively of objectives z_1 and z_2. Solutions of S are ordered by increasing value of criterion 1; let X_r and X_s be two consecutive solutions in S, thus with $z_{1r} < z_{1s}$ and $z_{2r} > z_{2s}$, where $z_{kl} = z_k(X_l)$. The following single-criterion problem is considered:

$$(\mathrm{P}_\lambda)\quad \begin{cases} \min & z_\lambda(X) = \lambda_1 z_1(X) + \lambda_2 z_2(X) \\ & X \in S = D \cap B^{(n)} \\ & \lambda_1 \geq 0, \quad \lambda_2 \geq 0. \end{cases}$$

This problem is optimized with a classical single objective CO algorithm for the values $\lambda_1 = z_{2r} - z_{2s}$ and $\lambda_2 = z_{1s} - z_{1r}$; with these values the search direction $z_\lambda(X)$ corresponds in the objective space to the line defined by Z_r and Z_s. Let $\{X^t\colon t = 1, \dots, T\}$ be the set of optimal solutions obtained in this manner and $\{Z_t\colon t = 1, \dots, T\}$ their images in the objective space. There are two possible cases:

- $\{Z_r, Z_s\} \cap \{Z_t\colon t = 1, \dots, T\} = \emptyset$: Solutions X^t are new supported efficient solutions. X^1 and X^T, provided $T > 1$, are put in S and, if $T > 2$, $X^2, \dots, X^{T-1}$ are put in S'. It will be necessary at further steps to consider the pairs (X^r, X^1) and (X^T, X^s)
- $\{Z_r, Z_s\} \subset \{Z_t\colon t = 1, \dots, T\}$: Solutions $\{X^t\colon t = 1, \dots, T\} \setminus \{X^r, X^s\}$ are new supported efficient solutions giving the same optimal value as X^r and X^s for $z_\lambda(X)$; we put them in list S'.



Multi-objective Combinatorial Optimization, Figure 1
***SE*(P) = *S* ∪ *S*′**

This first phase is continued until all pairs (X^r, X^s) of S have been examined without extension of S.

Finally, we obtain $SE(\mathrm{P}) = S \cup S'$ as illustrated in Fig. 1.

The purpose of the second phase is to generate the set $NSE(\mathrm{P}) = E(\mathrm{P}) \setminus SE(P)$ of nonsupported efficient solutions. Each nonsupported efficient solution has its image inside the triangle $\Delta\ Z_r Z_s$ determined by two successive solutions X^r and X^s of $SE(\mathrm{P})$ (see Fig. 1). So each of the $|SE(\mathrm{P})| - 1$ triangles $\Delta\ Z_r Z_s$ are successively analysed. This phase is more difficult to manage and is dependent of the particular MOCO problem analysed; in general, this second phase is achieved using partly a classical single objective CO method. An example of such second phase is given in ► Bi-objective Assignment Problem and in [14] for the bi-objective knapsack problem.

Heuristic Methods

As pointed out in [9,10,14], it is unrealistic to extend the exact methods describe above to MOCO problems with more than two criteria or more than a few hundred variables; the reason is that these methods are too consuming time. Because a *metaheuristic*, simulating annealing (SA), tabu search (TS), genetic algorithms (GA), etc., provide, for the single objective problem, excellent solutions in a reasonable time, it appeared logical to try to adapt these metaheuristics to a multi-objective framework.

The seminal work in this direction is the 1993 Ph.D. thesis of E.L. Ulungu, which gave rise to the so-called *MOSA method* to approximate $E(\mathrm{P})$ (see, in particular, [11]). After this pioneer study, this direction has been tackled by other research teams: P. Czyzak and A. Jaszkiewicz ([3]) proposed another way to adapt simulating annealing to a MOCO problem; independently, [4,5] and [1] did the same with tabu search, the later combining also tabu search and genetic algorithms; genetic algorithms are also used in [13].

The principle idea of MOSA method can be resumed in short terms. One begins with an initial iterate X_0 and initializes the set of potentially efficient points PE to just contain X_0. One then samples a point Y in the neighborhood of the current iterate. But instead of accepting Y if it is better than the current iterate on an objective: we now accept it if it is *not dominated* by any of the points currently in the set PE. If it is not dominated, we make Y the current iterate, add it to PE, and throw out any point in PE that are dominated by Y. On the other hand, if Y is dominated, we still make it the current iterate with some probability. In this way, as we move the iterate through the space, we simultaneously build up a set PE of potentially efficient points. The only complicated aspect of this scheme is the method for computing the acceptance probability for Y when it is dominated by a point in PE. The MOSA method is described in details in [11] and in ► Bi-objective Assignment Problem.

Interactive Determination of a Good Compromise

The general idea of *interactive methods* is described in ► Multi-objective Integer Linear Programming. Two types of methods can be distinguished, which we treat in the following subsections.

Goal Programming

As pointed out in [9], this methodology is often used by American researchers to treat several case studies. The general idea of *goal programming* method is to introduce for each objective k deviation variables d^+ and d^-, respectively by excess and by default, with respect to a certain a priori goal g_k, so that goal constraints are defined. If some priorities expressed by some weights p_k are given, this results in a single-objective problem (P_g)

defined by the global weighted deviation function:

$$(\mathrm{P}_g)\quad \begin{cases} \min & \sum_{k=1}^{K} p_k d_k^- \\ \text{s.t.} & z_k(X) + d_k^+ - d_k^- = g_k , \quad \forall k , \\ & X \in S = D \cap B^n . \end{cases}$$

When a solution is obtained, the decision maker can possibly modify the values of the goals g_k before a new iteration is performed. One drawback is that the additional goal constraints induce the loss of the particular structure of the initial CO problem, so that a general ILP software must be used to solve problem (P_g).

Interactive Two-Phase Methods and MOSA Method

The two-phase methodology described above can easily be adapted to build interactively a good compromise. At each step of the first phase, the decision maker can indicate which pair (X_r, X_s) he prefers so that only a small subset of $SE(\mathrm{P})$ is generated in the direction given by the decision maker; at the second phase, only one (or a few number of) triangles $\Delta Z_r Z_s$ is (are) analysed to verify if there exists in it a more satisfying non-supported efficient solution. In the same spirit, an interactive MOSA method can be designed (see also [12]): the decision maker gives some goals g_k and only the solutions satisfying $z_k(X) \leq g_k$ are putting in the list of potential efficient solutions. When this list contains a certain a priori fixed number of solutions, the decision maker indicates which one is preferred, modifies the goals g_k in a more restrictive sense before to continue the search with MOSA.

An example of such interactive procedure is given in [12] for a real case study.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Combinatorial Matrix Analysis
- ▶ Combinatorial Optimization Algorithms in Resource Allocation Problems
- ▶ Combinatorial Optimization Games
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Evolutionary Algorithms in Combinatorial Optimization
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fractional Combinatorial Optimization
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Neural Networks for Combinatorial Optimization
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling
- ▶ Replicator Dynamics in Combinatorial Optimization

References

1. Ben Abdelaziz F, Chaouachi J, Krichen S (1997) A hybrid heuristic for multiobjective knapsack problems. Techn Report Inst Sup Gestion, Tunisie, submitted
2. Current JR, Min H (1986) Multiobjective design of transportation networks: taxonomy and annotation. Europ J Oper Res 26(2):187–201
3. Czyzak P, Jaszkiewicz A (1998) Pareto simulated annealing – A metaheuristic technique for multiple objective combinatorial optimization. J Multi-Criteria Decision Anal 7:34–47
4. Gandibleux X, Mezdaoui N, Fréville A (1997) A tabu search procedure to solve multiobjective combinatorial optimisation problems. In: Caballero R, Steuer R (eds) Proc. Volume of MOPGP'96. Springer, Berlin
5. Hansen MP (1996) Tabu search for multiobjective optimization: MOTS. Techn Report Inst Math Modelling, Techn Univ Denmark
6. Hoogeveen H (1992) Single machine bicriteria scheduling. PhD Diss, Univ Eindhoven
7. Köksalan M, Köksalan–Konda CkiS (1997) Multiple criteria scheduling on single machine: A review and a general approach. In: Karwan M, et al. (eds) Essays in Decision Making. Springer, Berlin

8. Ulungu EL, Teghem J (1991) Multi-objective shortest problem path: A survey. In: Cerny M, Glackaufova D, Loula D (eds) Proc Internat Workshop on MCDM, Liblice, pp 176–188
9. Ulungu EL, Teghem J (1994) Multi-objective combinatorial optimization problems: A survey. J Multi-Criteria Decision Anal 3:83–104
10. Ulungu EL, Teghem J (1997) Solving multi-objective knapsack problem by a branch and bound procedure. In: Climaco J (ed) Multicriteria Analysis. Springer, Berlin, pp 269–278
11. Ulungu EL, Teghem J, Fortemps Ph, Tuyttens D (1999) MOSA method: A tool for solving MOCO problems I. Multi-Criteria Decision Anal 8:221–236
12. Ulungu EL, Teghem J, Ost Ch (1998) Efficiency of interactive multi-objective simulated annealing through a case study. J Oper Res Soc 49:1044–1050
13. Viennet R, Fontex M (1996) Multi-objective combinatorial optimization using a genetic algorithm for determining a Pareto set. Internat J Syst Sci 27(2):255–260
14. Visée M, Teghem J, Pirlot M, Ulungu EL (1998) Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. J Global Optim 12:139–155

Multi-objective Fractional Programming Problems

ZHIAN LIANG
Department of Applied Mathematics,
Shanghai University of Finance and Economics,
Shanghai, P.R. China

MSC2000: 90C29

Article Outline

Keywords and Phrases

Multiobjective fractional programming problem; (F, α, ρ, d)-convex functions; Efficient solution; Efficiency condition; Duality

Introduction

A number of optimization problems are actually multiobjective optimization problems (MOPs), where the objectives are conflicting. As a result, there is usually no single solution which optimizes all objectives simultaneously. A number of techniques have been developed to find a compromise solution to MOPs. The reader is referred to the recent book by Miettinen [16] about the theory and algorithms for MOPs. Fractional programming problems(FPPs) arise from many applied areas such as portfolio selection, stock cutting, game theory, and numerous decision problems in management science. Many approaches for FPPs have been exploited in considerable details. See, for example, Avriel et al. [3], Craven [5], Schaible [24,25], Schaible and Ibaraki [26] and Stancu-Minasian [27,28].

In this paper, we consider the following multiobjective fractional programming problem:

$$\text{(MFP)} \quad \min \quad \frac{f(x)}{g(x)} \triangleq \left(\frac{f_1(x)}{g_1(x)}, \frac{f_2(x)}{g_2(x)}, \dots, \frac{f_p(x)}{g_p(x)}\right)^{\mathrm{T}},$$
$$\text{s.t.} \quad h(x) \leq 0, \quad x \in X,$$

where $X \subset R^n$ is an open set, f_i, g_i $(i = 1, 2, \dots, p)$ are real-valued functions defined on X, and h is an m-dimensional vector-valued function defined on X. Suppose that $f_i(x) \geq 0$ and $g_i(x) > 0$ for $x \in X$ and $i = 1, 2, \dots, p$. Moreover, let f_i, g_i $(i = 1, 2, \dots, p)$ and h_j $(j = 1, 2, \dots, m)$ be continuously differentiable over X and denote the gradients of f_i, g_i and h_j at x by $\nabla f_i(x)$, $\nabla g_i(x)$ and $\nabla h_j(x)$, respectively.

If the parameter p in the problem (MFP) is equal to 1, then (MFP) corresponds to the following single-objective fractional programming problem:

$$\text{(FP)} \quad \min \frac{f(x)}{g(x)},$$
$$\text{s.t.} \quad h(x) \leq 0, \quad x \in X,$$

where $X \subset R^n$ is an open set, f, g are real-valued functions defined on X, and h is an m-dimensional vector-valued function defined on X, $f(x) \geq 0$ and $g(x) > 0$ for all $x \in X$. Moreover, assume that $f(x)$, $g(x)$ and $h_j(x)$ $(j = 1, 2, \dots, m)$ are continuously differentiable over X.

Khan and Hanson [10], and Reddy and Mukherjee [21] considered the optimality conditions and duality for (FP) with respect to the following generalized concepts of convexity, respectively.

Definition 1 [6] Let f be a real function defined on an open set $X \subseteq R^n$ and differentiable at x_0. Given a mapping $\eta : X \times X \to R^n$, the function f is said to be invex at x_0 with respect to η if, $\forall x \in X$, the following inequality holds:

$$f(x) - f(x_0) \geq \nabla f(x_0)^{\mathrm{T}} \eta(x, x_0) .$$

Definition 2 [7] Let f be a real function defined on an open set $X \subseteq R^n$ and differentiable at x_0. Given a real number ρ, a mapping $\eta : X \times X \to R^n$ and a scalar function $d : X \times X \to R$, the function f is said to be ρ-invex at x_0 with respect to η and d if, $\forall x \in X$, the following inequality holds:

$$f(x) - f(x_0) \geq \nabla f(x_0)^{\mathrm{T}} \eta(x, x_0) + \rho d^2(x, x_0) .$$

The authors of references [10,21] imposed the corresponding generalized convexity on the numerator and denominator individually for the objective function in the problem (FP), and then derived some optimality conditions and duality results. How to extend these methods to the multiobjective case is still an open problem [21].

As far as the multiobjective fractional problem (MFP) is concerned, Jeyakumar and Mond [8] introduced a concept of v-invexity as follows.

Definition 3 Let $f : X \to R^p$ be a real vector function defined on an open set $X \subseteq R^n$ and each component of f be differentiable at x_0. The function f is said to be v-invex at $x_0 \in X$ if there exist a mapping $\eta : X \times X \to R^n$ and a function $\alpha_i : X \times X \to R_+ \setminus \{0\}$ $(i = 1, 2, \ldots, p)$ such that, $\forall x \in X$,

$$f_i(x) - f_i(x_0) \geq \alpha_i(x, x_0) \nabla f_i(x_0)^{\mathrm{T}} \eta(x, x_0) .$$

Jeyakumar and Mond [8] obtained some weak efficiency conditions and duality results for a nonconvex multiobjective fractional programming problem via the concept of v-invexity, v-pseudoinvexity and v-quasiinvexity.

Motivated by various concepts of generalized convexity, Liang et al. [12] introduced a unified formulation of the generalized convexity, which was called (F, α, ρ, d)-convexity, and obtained some corresponding optimality conditions and duality results for the single-objective fractional problem (FP). In this paper, we will extend the methods adopted for the single-objective problem (FP) in [12] to the multiobjective problem (MFP).

Definition 4 A function $F : R^n \to R$ is said to be sublinear if for any $\alpha_1, \alpha_2 \in R^n$,

$$F(\alpha_1 + \alpha_2) \leq F(\alpha_1) + F(\alpha_2) , \tag{1}$$

and for any $r \in R_+$, $\alpha \in R^n$,

$$F(r\alpha) = rF(\alpha) . \tag{2}$$

Note that the concept of the sublinear function was given in Preda [20]. Now, a sublinear function is defined simply as a function that is subadditive and positively homogeneous, which is free of extraneous symbols in Preda [20]. It follows from (2) that $F(0) = 0$.

Based upon the concept of the sublinear function, we recall the unified formulation about generalized convexity, i. e., (F, α, ρ, d)-convexity, which was introduced in [12] as follows.

Definition 5 Given an open set $X \subset \Re^n$, a number $\rho \in R$, and two functions $\alpha : X \times X \to R_+ \setminus \{0\}$ and $d : X \times X \to R$, a differentiable function f over X is said to be (F, α, ρ, d)-convex at $x_0 \in X$ if for any $x \in X$, $F(x, x_0; \cdot) : \Re^n \to \Re$ is sublinear, and $f(x)$ satisfies the following condition:

$$\begin{aligned} f(x) - f(x_0) \geq & F(x, x_0; \alpha(x, x_0) \nabla f(x_0)) \\ & + \rho d^2(x, x_0) . \end{aligned} \tag{3}$$

The function f is said to be (F, α, ρ, d)-convex over X if, $\forall x_0 \in X$, it is (F, α, ρ, d)-convex at x_0; f is said to be strongly $(F, \alpha, \rho, d) - convex$ or $(F, \alpha) - convex$ if $\rho > 0$ or $\rho = 0$, respectively.

From Definition 5, there are the following special cases:

(i) If $\alpha(x, x_0) = 1$ for all $x, x_0 \in X$, then (F, α, ρ, d)-convexity is (F, ρ)-convexity [20].

(ii) If $F(x, x_0; \alpha(x, x_0) \nabla f(x_0)) = \nabla f(x_0)^{\mathrm{T}} \eta(x, x_0)$ for a certain mapping $\eta : X \times X \to R^n$, then (F, α, ρ, d)-convexity is ρ-invexity defined in [7].

(iii) If $\rho = 0$ or $d(x, x_0) \equiv 0$ for all $x, x_0 \in X$ and $F(x, x_0; \alpha(x, x_0) \nabla f(x_0)) = \nabla f(x_0)^{\mathrm{T}} \eta(x, x_0)$

for a certain mapping $\eta : X \times X \to R^n$, then (F, α, ρ, d)-convexity reduces to invexity [6].

In the following, ρ, α and d are referred to as parameters of (F, α, ρ, d)-convexity. Furthermore, we will adopt the following conventions.

Let R^n_+ denote the nonnegative orthant of R^n and x^{T} denote the transpose of the vector $x \in R^n$. For any two vectors $x = (x_1, x_2, \dots, x_n)^{\mathrm{T}}, y = (y_1, y_2, \dots, y_n)^{\mathrm{T}} \in R^n$, we denote:

$$\begin{array}{lll} x = y \text{ implying} & x_i = y_i, & i = 1, 2, \dots, n; \\ x \prec y \text{ implying} & x_i \le y_i, & i = 1, 2, \dots, n, \\ & & \text{but } x \ne y; \\ x < y \text{ implying} & x_i < y_i, & i = 1, 2, \dots, n; \\ x \not\prec y \text{ implying} & y_i < x_i & \text{for at leastone } i\,. \end{array}$$

A solution of the problem (MFP) is referred to as an efficient (Pareto optimal) solution, which is defined as follows.

Definition 6 A feasible solution $x_0 \in X$ of (MFP) is called an efficient solution of (MFP) if there exists no other feasible solution $x \in X$ such that

$$\frac{f(x)}{g(x)} \prec \frac{f(x_0)}{g(x_0)}\,.$$

In [14], Maeda gave a kind of constraint qualification, which was called generalized Guignard constraint qualification (GGCQ), under which he derived the following Kuhn–Tucker type necessary conditions for a feasible solution x_0 to be an efficient solution to the problem (MFP):

If x_0 is an efficient solution of (MFP) and (GGCQ) holds at x_0 [14], then there exist $\tau = (\tau_1, \tau_2, \dots, \tau_p)^{\mathrm{T}} \in R^p_+, \tau > 0, \sum_{i=1}^p \tau_i = 1$ and $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_m)^{\mathrm{T}} \in R^m_+$ such that

$$\sum_{i=1}^p \tau_i \nabla \frac{f_i(x_0)}{g_i(x_0)} + \sum_{j=1}^m \lambda_j \nabla h_j(x_0) = 0,$$
$$\lambda_j h_j(x_0) = 0, \qquad j = 1, 2, \dots, m\,.$$

This paper is organized as follows. In Sect. "Efficiency Conditions", efficiency conditions for the multiobjective fractional problem (MFP) involving (F, α, ρ, d)-convexity are presented. The duality properties of the problem (MFP) are studied in Sect. "Duality", including several duals for (MFP) and some weak and strong duality theorems. Concluding remarks are given in the last section.

Efficiency Conditions

First, we present a lemma which indicates that (F, α, ρ, d)-convexity can be preserved after taking division.

Lemma 1 *Let $X \subset R^n$ be an open set. Assume that p, q are real-valued differentiable functions defined on X and $p(x) \ge 0, q(x) > 0$ for all $x \in X$. If p and $-q$ are (F, α, ρ, d)-convex at $x_0 \in X$, then p/q is $(F, \overline{\alpha}, \overline{\rho}, \overline{d})$-convex at x_0, where $\overline{\alpha}(x, x_0) = \frac{\alpha(x,x_0)q(x_0)}{q(x)}, \overline{\rho} = \rho\left(1 + \frac{p(x_0)}{q(x_0)}\right)$ and $\overline{d}(x, x_0) = \frac{d(x,x_0)}{q^{\frac{1}{2}}(x)}$.*

In the following, we present some sufficient efficiency conditions for (MFP) under appropriate (F, α, ρ, d)-convexity assumptions.

Theorem 1 *Let x_0 be a feasible solution of (MFP). Suppose that there exist $\tau = (\tau_1, \tau_2, \dots, \tau_p)^T \in R^p_+$, $\tau > 0$, $\sum_{i=1}^p \tau_i = 1$ and $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_m)^T \in R^m_+$ such that*

$$\sum_{i=1}^p \tau_i \nabla \frac{f_i(x_0)}{g_i(x_0)} + \sum_{j=1}^m \lambda_j \nabla h_j(x_0) = 0\,, \tag{4}$$

$$\lambda_j h_j(x_0) = 0, \qquad j = 1, 2, \dots, m\,. \tag{5}$$

If f_i and $-g_i$ $(i = 1, 2, \dots, p)$ are $(F, \alpha_i, \rho_i, d_i)$-convex at x_0, h_j $(j = 1, 2, \dots, m)$ is $(F, \beta_j, \zeta_j, c_j)$-convex at x_0, and

$$\sum_{i=1}^p \tau_i \overline{\rho}_i \frac{\overline{d}_i^2(x, x_0)}{\overline{\alpha}_i(x, x_0)} + \sum_{j=1}^m \lambda_j \zeta_j \frac{c_j^2(x, x_0)}{\beta_j(x, x_0)} \ge 0\,, \tag{6}$$

where $\overline{\alpha}_i(x, x_0) = \frac{\alpha_i(x,x_0)g_i(x_0)}{g_i(x)}, \overline{\rho}_i = \rho_i\left(1 + \frac{f_i(x_0)}{g_i(x_0)}\right)$, and $\overline{d}_i(x, x_0) = \frac{d_i(x,x_0)}{g_i^{\frac{1}{2}}(x)}$, then x_0 is a global efficient solution for (MFP).

Corollary 1 *Let x_0 be a feasible solution of (MFP). Suppose that there exist $\tau = (\tau_1, \tau_2, \dots, \tau_p)^T \in R^p_+$, $\tau > 0, \sum_{i=1}^p \tau_i = 1$, and $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_m)^T \in R^m_+$ such that*

$$\sum_{i=1}^p \tau_i \nabla \frac{f_i(x_0)}{g_i(x_0)} + \sum_{j=1}^m \lambda_j \nabla h_j(x_0) = 0,$$
$$\lambda_j h_j(x_0) = 0, \qquad j = 1, 2, \dots, m\,.$$

If f_i and $-g_i$ $(i = 1,2,\dots,p)$ are strongly (F,α_i,ρ_i,d_i)-convex (or (F,α_i)-convex) at x_0, h_j $(j = 1,2,\dots,m)$ is strongly (F,β_j,ζ_j,c_j)-convex (or (F,β_j)-convex) at x_0, then x_0 is a global efficient solution for (MFP).

For $i = 1,2,\dots,p$, if $g_i(x) = 1$ for all $x \in X$, $f_i(x)$ need not be nonnegative, and the functions involved are assumed to be invex, ρ-invex with respect to $\eta\colon X\times X \to R^n$, $d\colon X\times X \to R$, (F,ρ)-convex, or generalized (F,ρ)-convex, respectively, then we can obtain the corresponding results presented in [1,2,9].

Next, we consider a special case of (MFP), in which the fractional objective functions have the same denominator. For $i = 1,2,\dots,p$, let $g_i(x) = g(x)$ in (MFP). The property about the efficient solution of this special (MFP) can be obtained similarly as that in Theorem 1, so we state the following theorem:

Theorem 2 *Let x_0 be a feasible solution of (MFP). Suppose that there exist $\tau = (\tau_1,\tau_2,\dots,\tau_p)^T \in R^p_+$, $\tau > 0$, $\sum_{i=1}^p \tau_i = 1$, and $\lambda = (\lambda_1,\lambda_2,\dots,\lambda_m)^T \in R^m_+$ such that*

$$\sum_{i=1}^p \tau_i \nabla \frac{f_i(x_0)}{g(x_0)} + \sum_{j=1}^m \lambda_j \nabla h_j(x_0) = 0,$$
$$\lambda_j h_j(x_0) = 0, \qquad j = 1,2,\dots,m\,.$$

If $-g$ is (F,α,ρ,d)-convex at x_0,f_i $(i = 1,2,\dots,p)$ is (F,α,ρ_i,d)-convex at x_0,h_j $(j = 1,2,\dots,m)$ is $(F,\overline{\alpha},\zeta_j,\overline{d})$-convex at x_0, and $\sum_{i=1}^p \tau_i\overline{\rho}_i + \sum_{j=1}^m \lambda_j\zeta_j \ge 0$, where $\overline{\alpha}(x,x_0) = (\alpha(x,x_0)g(x_0))/g(x)$, $\overline{\rho}_i = \rho_i + \rho(f_i(x_0))/g(x_0)$ and $\overline{d}(x,x_0) = (d(x,x_0))/(g^{\frac{1}{2}}(x))$, then x_0 is a global efficient solution for (MFP).

Finally, we present an equivalent formulation of the problem (MFP). Let $G(x) = \prod_{i=1}^p g_i(x)$, $G_i(x) = \frac{G(x)}{g_i(x)}$ $(i = 1,2,\dots,p)$. Then (MFP) can be written in the following form:

$$(\overline{\text{MFP}})$$
$$\min\left(\frac{G_1(x)f_1(x)}{G(x)}, \frac{G_2(x)f_2(x)}{G(x)}, \dots, \frac{G_p(x)f_p(x)}{G(x)}\right)^{\mathrm{T}},$$
$$\text{s.t.}\quad h(x) \le 0,\ x \in X.$$

By Theorem 2, we have the following corollary:

Corollary 2 *Let x_0 be a feasible solution of $(\overline{\text{MFP}})$. Suppose that there exist $\tau = (\tau_1,\tau_2,\dots,\tau_p)^T \in R^p_+$, $\tau > 0$, $\sum_{i=1}^p \tau_i = 1$, and $\lambda = (\lambda_1,\lambda_2,\dots,\lambda_m)^T \in R^m_+$ such that*

$$\sum_{i=1}^p \tau_i \nabla \frac{f_i(x_0)}{g_i(x_0)} + \sum_{j=1}^m \lambda_j \nabla h_j(x_0) = 0,$$
$$\lambda_j h_j(x_0) = 0, \quad j = 1,2,\dots,m.$$

If $-G$ is (F,α,ρ,d)-convex at x_0, $G_i f_i (i = 1,2,\dots,p)$ is (F,α,ρ_i,d)-convex at x_0, $h_j(j = 1,2,\dots,m)$ is $(F,\overline{\alpha},\zeta_j,\overline{d})$-convex at x_0, and $\sum_{i=1}^p \tau_i\overline{\rho}_i + \sum_{j=1}^m \lambda_j\zeta_j \ge 0$, where $\overline{\rho}_i = \rho_i + \rho(f_{i(x_0)})/(g_i(x_0))$, $\overline{\alpha}(x,x_0) = (\alpha(x,x_0)G(x_0))/G(x)$, and $\overline{d}(x,x_0) = (d(x,x_0))/(G^{1/2}(x))$, then x_0 is a global efficient solution for (MFP).

Under the assumptions of Theorem 2 or Corollary 2, if $\rho \ge \max\limits_{1\le i\le p} \rho_i$, $\overline{\rho}_i = \rho_i(1 + f_i(x_0)/g(x_0))$, or $\overline{\rho}_i = \rho_i(1 + f_i(x_0)/g_i(x_0))$, respectively, then the corresponding results still hold.

Duality

Many types of duals for a given mathematical programming problem. Two well-known duals are the Wolfe type dual [29] and the Mond-Weir type dual [17]. Recently, the mixed (or general type) dual has been considered for various optimization problems [1,2,11,13,18,19,20,30,31,32]. The mixed dual includes the Wolfe type dual and the Mond-Weir type dual as special cases. In the sequel, the generalized Mond-Weir dual are discussed first, and then three other types of duals are presented, which are based on (F,α,ρ,d)-convexity for the problem (MFP).

Let $M = \{1,2,\dots,m\}$ and $M_0, M_1,\dots,M_q$ be a partition of M, i. e., $\bigcup_{k=1}^q M_k = M$, $M_k \bigcap M_l = \emptyset$ for $k \ne l$. The generalized Mond-Weir dual of (MFP) is as follows:

$$\max \frac{f(u)}{g(u)} + \lambda^{\mathrm{T}}_{M_0} h_{M_0}(u)\, e \overset{\Delta}{=} \left(\frac{f_1(u)}{g_1(u)} + \lambda^{\mathrm{T}}_{M_0} h_{M_0}(u), \dots, \frac{f_p(u)}{g_p(u)} + \lambda^{\mathrm{T}}_{M_0} h_{M_0}(u)\right)^{\mathrm{T}},$$

$$\text{s.t.} \sum_{i=1}^p \tau_i \nabla \frac{f_i(u)}{g_i(u)} + \sum_{j=1}^m \lambda_j \nabla h_j(u) = 0,$$

$$\begin{aligned}
&\lambda_{M_k}^{\mathrm{T}} h_{M_k}(u) \geq 0, k = 1,2,\dots,q,\\
&\tau = (\tau_1, \tau_2, \dots, \tau_p)^{\mathrm{T}} \in R_+^p, \tau > 0,\\
&\sum_{i=1}^{p} \tau_i = 1,\\
&\lambda_{M_k} \in R_+^{|M_k|}, k = 0,1,2,\dots,q,\\
&u \in X,
\end{aligned}$$

where $e = (1,1,\dots,1)^{\mathrm{T}}$ and λ_{M_k} denotes the column vector whose subscripts of components belong to M_k. In particular, if $M_0 = M, M_k = \emptyset, k = 1,2,\dots,q$, then the above dual becomes the Wolfe type dual; if $M_0 = \emptyset$ and $q = 1, M_1 = M$, the Mond-Weir type dual is obtained. Since the Wolfe type dual is unsuitable for single objective fractional programming problems [15,22,23], the duals with $M_0 \neq \emptyset$ are certainly unsuitable for (MFP). For the generalized Mond-Weir type dual, we only consider the case $M_0 = \emptyset, M_1 = M$, i. e., the Mond-Weir dual.

Mond-Weir Dual

The Mond-Weir dual of the problem (MFP) has the following form:

(MFD1)

$$\begin{aligned}
&\max \frac{f(u)}{g(u)} = \left(\frac{f_1(u)}{g_1(u)}, \frac{f_2(u)}{g_2(u)}, \dots, \frac{f_p(u)}{g_p(u)}\right)^{\mathrm{T}}\\
&\text{s.t.} \sum_{i=1}^{p} \tau_i \nabla \frac{f_i(u)}{g_i(u)} + \sum_{j=1}^{m} \lambda_j \nabla h_j(u) = 0,\\
&\quad \lambda^{\mathrm{T}} h(u) \geq 0,\\
&\quad \tau = (\tau_1, \tau_2, \dots, \tau_p)^{\mathrm{T}} \in R^p, \tau > 0, \sum_{i=1}^{p} \tau_i = 1,\\
&\quad \lambda = (\lambda_1, \lambda_2, \dots, \lambda_m)^{\mathrm{T}} \in R_+^m, u \in X.
\end{aligned}$$

Theorem 3 *(Weak Duality) Assume that $\overline{x}$ is a feasible solution of (MFP) and $(\overline{u}, \overline{\tau}, \overline{\lambda})$ is a feasible solution of (MFD1). If f_i and $-g_i$ $(i = 1,2,\dots,p)$ are $(F, \alpha_i, \rho_i, d_i)$-convex at $\overline{u}$, h_j $(j = 1,2,\dots,m)$ is (F, β, ζ_j, c_j)-convex at $\overline{u}$, and the inequality*

$$\sum_{i=1}^{p} \overline{\tau}_i \overline{\rho}_i \frac{\overline{d}_i^2(\overline{x}, \overline{u})}{\overline{\alpha}_i(\overline{x}, \overline{u})} + \sum_{j=1}^{m} \overline{\lambda}_j \zeta_j \frac{c_j^2(\overline{x}, \overline{u})}{\beta(\overline{x}, \overline{u})} \geq 0 \qquad (7)$$

holds, where $\overline{\alpha}_i(\overline{x}, \overline{u}) = \alpha_i(\overline{x}, \overline{u})(g(\overline{u}))/(g(\overline{x}))$, $\overline{\rho}_i = \rho_i(1 + (f_i(\overline{u}))/(g_i(\overline{u})))$, and $\overline{d}_i(\overline{x}, \overline{u}) = (d_i(\overline{x}, \overline{u}))/(g_i^{\frac{1}{2}}(\overline{x}))$, then we have

$$\frac{f(\overline{x})}{g(\overline{x})} \not< \frac{f(\overline{u})}{g(\overline{u})}.$$

Corollary 3 *(Weak Duality) Assume that $\overline{x}$ is a feasible solution of (MFP), and $(\overline{u}, \overline{\tau}, \overline{\lambda})$ is a feasible solution of (MFD1). If f_i and $-g_i$ $(i = 1,2,\dots,p)$ are strongly $(F, \alpha_i, \rho_i, d_i)$-convex (or (F, α_i)-convex) at $\overline{u}$, and h_j $(j = 1,2,\dots,m)$ is strongly (F, β, ζ_j, c_j)-convex (or (F, β)-convex) at $\overline{u}$, then*

$$\frac{f(\overline{x})}{g(\overline{x})} \not< \frac{f(\overline{u})}{g(\overline{u})}.$$

Theorem 4 *(Strong Duality) Assume that $\overline{x}$ is an efficient solution of (MFP) and the constraint qualification (GGCQ) holds at $\overline{x}$ [14]. Then there exists $(\overline{\tau}, \overline{\lambda}) \in R_+^p \times R_+^m$ such that $(\overline{x}, \overline{\tau}, \overline{\lambda})$ is a feasible solution of (MFD1), and the objective function values of (MFP) and (MFD1) at the corresponding points are equal. If the assumptions about the generalized convexity and the inequality (7) in Theorem 3 are also satisfied, then $(\overline{x}, \overline{\tau}, \overline{\lambda})$ is an efficient solution of (MFD1).*

Schaible Dual

In this subsection, we shall consider the following extended form of the Schaible dual for (MFP) [22,23]:

(MFD2)

$$\begin{aligned}
&\max \lambda = (\lambda_1, \lambda_2, \dots, \lambda_p)^{\mathrm{T}}\\
&\quad \text{s.t.} \sum_{i=1}^{p} \tau_i \nabla_u (f_i(u) - \lambda_i g_i(u)) + \sum_{j=1}^{m} \nu_j \nabla h_j(u)\\
&\qquad = 0,\\
&\qquad f_i(u) - \lambda_i g_i(u) \geq 0, \; i = 1,2,\dots,p\\
&\qquad \nu^{\mathrm{T}} h(u) \geq 0,\\
&\qquad \tau > 0, \quad \sum_{i=1}^{p} \tau_i = 1,\\
&\qquad \lambda \in R_+^p, \quad \tau \in R_+^p, \quad \nu \in R_+^m, \quad u \in X.
\end{aligned}$$

Theorem 5 *(Weak Duality). Assume that $\overline{x}$ is a feasible solution of (MFP) and $(\overline{u}, \overline{\tau}, \overline{\lambda}, \overline{\nu})$ is a feasible solution of (MFD2). If one of the following holds:*

- *(I) f_i and $-g_i$ $(i = 1,2,\dots,p)$ are (F,α_i,ρ_i,d_i)-convex at $\overline{u}$, h_j $(j = 1,2,\dots,m)$ is (F,β,ζ_j,c_j)-convex at $\overline{u}$, and*

$$\sum_{i=1}^{p}\overline{\tau}_i\rho_i(1+\overline{\lambda}_i)\frac{d_i^2(\overline{x},\overline{u})}{\alpha_i(\overline{x},\overline{u})}+\sum_{j=1}^{m}\overline{\nu}_j\zeta_j\frac{c_j^2(\overline{x},\overline{u})}{\beta(\overline{x},\overline{u})}\ge 0; \tag{8}$$

- *(II) f_i and $-g_i$ $(i = 1,2,\dots,p)$ are (F,α,ρ_i,d)-convex at $\overline{u}$, h_j $(j = 1,2,\dots,m)$ is (F,α,ζ_j,d)-convex at $\overline{u}$, and the vectors $\overline{\tau},\overline{\lambda},\overline{\nu}$ satisfy:*

$$\sum_{i=1}^{p}\overline{\tau}_i\rho_i(1+\overline{\lambda}_i)+\sum_{j=1}^{m}\overline{\nu}_j\zeta_j\ge 0\,, \tag{9}$$

then

$$\frac{f(\overline{x})}{g(\overline{x})}\not<\overline{\lambda}.$$

Theorem 6 *(Strong Duality). Assume $\overline{x}$ is an efficient solution of (MFP), and the constraint qualification (GGCQ) holds at $\overline{x}$ [14]. Then there exist $\overline{\tau}\in R_+^p$, $\overline{\lambda}\in R_+^p$, $\overline{\nu}\in R_+^m$ such that $(\overline{x},\overline{\tau},\overline{\lambda},\overline{\nu})$ is a feasible solution of (MFD2) and $\overline{\lambda}=\frac{f(\overline{x})}{g(\overline{x})}$. Furthermore, if all assumptions in Theorem 5 are satisfied, then the corresponding $(\overline{x},\overline{\tau},\overline{\lambda},\overline{\nu})$ is an efficient solution of (MFD2).*

Extended Bector Type Dual

For a single-objective fractional programming problem in [4], Bector used the positivity of the denominator to transform the inequality constraints and add them to the objective by Lagrangian mulitipliers for establishing a kind of dual. Since the denominators in (MFP) need not be the same, we use the equivalent form $(\overline{MFP})$ of (MFP) to establish the following dual, which is called the extended Bector type dual of (MFP):

(MFD3)

$$\max\begin{pmatrix}\frac{G_1(u)f_1(u)+\nu_{M_0}^{\mathrm{T}}h_{M_0}(u)}{G(u)}\\ \vdots\\ \frac{G_p(u)f_p(u)+\nu_{M_0}^{\mathrm{T}}h_{M_0}(u)}{G(u)}\end{pmatrix}^{\mathrm{T}}$$

$$\begin{aligned}\text{s.t.}\ &\sum_{i=1}^{p}\tau_i\nabla_u\frac{G_i(u)f_i(u)+\nu_{M_0}^{\mathrm{T}}h_{M_0}(u)}{G(u)}\\ &\quad+\sum_{k=1}^{q}\nabla_u\nu_{M_k}^{\mathrm{T}}h_{M_k}(u)=0,\\ &\nu_{M_k}^{\mathrm{T}}h_{M_k}(u)\ge 0,\quad k=1,2,\dots,q,\\ &G_i(u)f_i(u)+\nu_{M_0}^{\mathrm{T}}h_{M_0}(u)\ge 0,\\ &i=1,2,\dots,p,\\ &\sum_{i=1}^{p}\tau_i=1,\tau=(\tau_1,\tau_2,\dots,\tau_p)^{\mathrm{T}}\in R_+^p,\\ &\tau>0,\\ &u\in X,\quad \nu_{M_k}\in R_+^{|M_k|},\quad k=0,1,2,\dots,q\,.\end{aligned}$$

Theorem 7 *(Weak Duality) Let x be a feasible solution of (MFP) and (u,τ,ν) be a feasible solution of (MFD3). Assume that $-G$ is (F,α,ρ,d)-convex at u, G_if_i $(i = 1,\dots,p)$ is (F,α,ρ_i,d)-convex at u and h_j $(j = 1,\dots,m)$ is (F,α,ζ_j,d)-convex at u. If $\rho\ge\max_{1\le i\le p}\rho_i$ and the following inequality holds:*

$$\sum_{i=1}^{p}\tau_i\rho_i\left(1+\frac{G_i(u)f_i(u)+\nu_{M_0}^{T}h_{M_0}(u)}{G(u)}\right)+\sum_{j\in M_0}\nu_j\zeta_j+G(u)\sum_{k=1}^{q}\sum_{j\in M_k}\nu_j\zeta_j\ge 0\,, \tag{10}$$

then we have

$$\frac{f(x)}{g(x)}\not<\frac{\overline{G}(u)f(u)+\nu_{M_0}^{T}h_{M_0}(u)\,e}{G(u)}\,,$$

where $\overline{G}(u)=diag\{G_1(u),\dots,G_p(u)\}$ and each component in $e\in R^p$ is equal to 1.

Theorem 8 *(Strong Duality) Assume that $\overline{x}$ is an efficient solution of (MFP) and the constraint qualification (GGCQ) holds at $\overline{x}$ [14]. Then there exists $(\overline{\tau},\overline{\nu})$ such that $(\overline{x},\overline{\tau},\overline{\nu})$ is a feasible solution of (MFD3), and the objective function values of (MFP) and (MFD3) at $\overline{x}$ and $(\overline{x},\overline{\tau},\overline{\nu})$, respectively, are equal. If the assumptions and conditions in Theorem 7 are also satisfied, then $(\overline{x},\overline{\tau},\overline{\nu})$ is an efficient solution of (MFD3).*

Concluding Remarks

In this paper, a unified formulation of the generalized convexity defined in [12] is adopted, which includes many other generalized convexity concepts in optimization theory as special cases. Our concept of generalized convexity is suitable to analyze the efficiency conditions and duality of multiobjective fractional programming problems. Efficiency conditions and duality for a class of multiobjective fractional programming problems are presented. We extend the methods, which were adopted for single-objective fractional programming problems in [10,12,21], to the case with multiple fractional objectives. We also present the extended Bector type dual by using an equivalent formulation of the primal problem. Note that we only consider (MFP) from a viewpoint of the efficient solution in this paper. The methods used here can be extended to the study of (MFP) from a viewpoint of the weak efficient solution.

References

1. Aghezzaf B, Hachimi M (2000) Generalized convexity and duality in multiobjective programming problems. J Glob Optim 18:91–101
2. Aghezzaf B, Hachimi M (2001) Sufficiency and duality in multiobjective programming involving generalized (F, ρ)-convexity. J Math Anal Appl 258:617–628
3. Avriel M, Diewert WE, Schaible S, Zang I (1988) Generalized Concaveity. Plenum Press, New York
4. Bector CR (1973) Duality in nonlinear fractional programming. Z Oper Res 17:183–193
5. Craven BD (1988) Fractional Programming. Heldermann, Berlin
6. Hanson MA (1981) On sufficiency of the Kuhn–Tucker conditions. J Math Anal Appl 80:544–550
7. Jeyakumar V (1985) Strong and weak invexity in mathematical programming. Methods Oper Res 55:109–125
8. Jeyakumar V, Mond B (1992) On generalized convex mathematical programming. J Aust Math Soc Series B 34:43–53
9. Kaul RN, Suneja SK, Srivastava MK (1994) Optimality criteria and duality in multiple-objective optimization involving generalized invexity. J Optim Theor Appl 80(3):465–482
10. Khan Z, Hanson MA (1997) On ratio invexity in mathematical programming. J Math Anal Appl 205:330–336
11. Li Z (1993) Duality theorems for a class of generalized convex multiobjective programming problems. Acta Scientiarum Naturalium Universitatis NeiMongol 24(2):113–118
12. Liang ZA, Huang HX, Pardalos PM (2001) Optimality conditions and duality for a class of nonlinear fractional programming problems. J Optim Theor Appl 110(3):611–619
13. Liang Z, Ye Q (2001) Duality for a class of multiobjective control problems with generalized invexity. J Math Anal Appl 256:446–461
14. Maeda T (1994) Constraint qualifications in multiobjective optimization problems: differentiable case. J Optim Theor Appl 80(3):483–500
15. Mangasarian OL (1969) Nonlinear Programming. McGraw-Hill, New York
16. Miettinen KM (1999) Nonlinear Multiobjective Optimization. Kluwer, Dordrecht
17. Mond B, Weir T (1981) Generalized concavity and daulity. In: Schaible S, Ziemba WT (eds) Generalized Convexity in Optimization and Economics. Academic Press, New York, pp 263–280
18. Mond B, Weir T (1982) Duality for fractional programming with generalized convexity conditions. J Inf Optim Sci 3(2):105–124
19. Mukherjee RN, Rao CP (2000) Mixed type duality for multiobjective variational problems. J Math Anal Appl 252:571–586
20. Preda V (1992) On efficiency and duality for multiobjective programs. J Math Anal Appl 166:365–377
21. Reddy LV, Mukherjee RN (1999) Some results on mathematical programming with generalized ratio invexity. J Math Anal Appl 240:299–310
22. Schaible S (1976) Duality in fractional proramming: a unified approach. Oper Res 24:452–461
23. Schaible S (1976) Fractional programming, I. Duality Manag Sci 22:858–867
24. Schaible S (1981) Fractional programming: applications and algorithms. Eur J Oper Res 7:111–120
25. Schaible S (1995) Fractional Programming. In: Horst R, Pardalos PM (eds) Handbook of Global Optimization. Kluwer, Dordrecht, pp 495–608
26. Schaible S, Ibaraki T (1983) Fractional programming. Eur J Oper Res 12:325–338
27. Stancu-Minasian IM (1997) Fractional programming: theory, methods and applications. Kluwer, Dordrecht
28. Stancu-Minasian IM (1999) A fifth bibliograpy of fractional programming. Optimization 45:343–367
29. Wolfe P (1961) A duality theorem for nonlinear programming. Q Appl Math 19:239–244
30. Xu Z (1996) Mixed type duality in multiobjective programming problems. J Math Anal Appl 198:621–635
31. Yang XM, Teo KL, Yang XQ (2000) Duality for a class of nondifferentiable multiobjective programming problems. J Math Anal Appl 252:999–1005
32. Zhang Z, Mond B (1997) Duality for a nondifferentiable programming problem. Bull Aust Math Soc 55:29–44

Multi-objective Integer Linear Programming

MOILP

JACQUES TEGHEM
Lab. Math. & Operational Research Fac.,
Polytechn. Mons, Mons, Belgium

MSC2000: 90C29, 90C10

Article Outline

Keywords

Multi-objective programming; Integer; Linear programming

From the 1970s onwards, *multi-objective linear programming* (MOLP) methods with continuous solutions have been developed [8]. However, it is well known that discrete variables are unavoidable in the linear programming modeling of many applications, for instance, to represent an investment choice, a production level, etc.

The mathematical structure is then *integer linear programming* (ILP), associated with MOLP giving a MOILP problem. Unfortunately, MOILP cannot be solved by simply combining ILP and MOLP methods, because it has got its own specific difficulties.

The problem (P) considered is defined as

$$
(\mathrm{P})\quad \begin{cases} \underset{X\in D}{{}'\max{}'} & z_k(X) = \sum_{j=1}^{n} c_j^{(k)} x_j, \\ & k = 1,\dots,K, \\ \text{where} & D = \left\{ X \in \mathbb{R}^n : \begin{array}{c} TX \le d, \\ X \ge 0, \\ x_j \text{ integer}, \\ j \in J \end{array} \right\} \\ \text{with} & T(m\times n), \\ & d(m\times 1), \\ & X(n\times 1), \\ & J \subset \{1,\dots,n\}. \end{cases}
$$

If we denote $LD = \{X\colon TX \le d, X \ge 0\}$, problem (LP) is the linear relaxation of problem (P):

$$
(\mathrm{LP})\quad \begin{cases} {}'\max{}' & z_k(X), \quad k = 1,\dots,K, \\ & X \in LD \end{cases}
$$

A solution $X^\star$ in D (or LD) is said to be *efficient* for problem (P) (or (LP)) if there does not exist any other solution in D (or LD) such that $z_k(X) \ge z_k(X^\star)$, $k = 1, \dots, K$, with at least one strict inequality.

Let $E(\cdot)$ denote the set of all efficient solutions of problem $(\cdot)$. It is well known (see [8]) that (LP) may be characterized by the optimal solutions of the single objective and parametrized problem:

$$
(\mathrm{LP}_\lambda)\quad \begin{cases} \max & \sum_{k=1}^{K} \lambda_k z_k(X) \\ & X \in LD \\ \text{with} & \lambda_k > 0, \quad \forall k, \\ & \sum_{k=1}^{K} \lambda_k = 1 \end{cases}
$$

This fundamental principle – often called *Geoffrion's theorem* – is no longer valid in presence of discrete variables because the set D is not convex. The set of optimal solutions of problem (P_λ), defined as problem (LP_λ) in which LD is replaced by D, is only a subset $SE(\mathrm{P})$ of $E(\mathrm{P})$; the solutions in $SE(\mathrm{P})$ are called *supported efficient solutions*, while the solutions belonging to $NSE(\mathrm{P}) = E(\mathrm{P}) \setminus SE(\mathrm{P})$ are called *nonsupported efficient solutions*.

The breakdown of Geoffrion's theorem for problem (P) can be illustrated by the following obvious example:

$$
\begin{aligned}
&K = 2, \\
&z_1(X) = 6x_1 + 3x_2 + x_3, \\
&z_2(X) = x_1 + 3x_2 + 6x_3, \\
&D = \{X\colon\ x_1 + x_2 + x_3 \le 1,\ x_i \in \{0,1\}\}.
\end{aligned}
$$

For this problem,

$$
E(\mathrm{P}) = \{(1,0,0); (0,1,0); (0,0,1)\}
$$

while $NSE(\mathrm{P}) = \{(0, 1, 0)\}$.

Nevertheless, V.J. Bowman [1] has given a theoretical characterization of E(P): Setting

$$M_k = \max_{X\in D} z_k(X),$$
$$\overline{z}_k = M_k + \varepsilon_k, \quad \text{with } \varepsilon_k > 0,$$
$$\rho > 0,$$

then E(P) is characterized by the optimal solutions of the problem(P^T_λ):

$$\min_{X\in D}\max_k \left(\lambda_k\left(\overline{z}_k - z_k(X)\right) + \rho\left(\sum_{k=1}^{K}\left(\overline{z}_k - z_k(X)\right)\right)\right),$$

consisting of minimizing the augmented weighted Tchebychev distance between $z_k(X)$ and $\overline{z}_k$.

Let us note that another characterization of E(P) is given in [2] for the particular case of binary variables.

Two types of problems can be analysed:

- Generate E(P) explicitly. Several methods have been proposed; they are reviewed in [10]. below we will present two of them, which appear general, characteristic and efficient.
- To determine interactively with the decision maker a 'best compromise' in E(P) according to the preferences of the decision maker. Some of the existing approaches are reviewed in [11]; below we will describe three of these interactive methods.

Generation of *E*(P)

Klein–Hannan Method

See [5]. This is an iterative procedure for sequentially generating the complete set of efficient solutions for problem (P) (we suppose that the coefficients $c_j^{(k)}$ are integers); it consists in solving a sequence of progressively more constrained single objective ILP problems and can be implemented through use of any ILP algorithm.

- (Initialization: step 0) An objective function $l \in \{1, \dots, K\}$ is chosen arbitrarily and the following single objective ILP problem is considered:

  $$(\mathrm{P}_0) \quad \max_{X\in D} z_l(X).$$

  Let $E(\mathrm{P}_0)$ be the set of all optimal solutions of (P_0) and let E_0(P) be the set of solutions defined as $E_0(\mathrm{P}) = E(\mathrm{P}_0) \cap E(\mathrm{P})$. Thus, E_0(P) is the subset of nondominated solutions in $E(\mathrm{P}_0)$.
- (Step j, ($j \geq 1$)) The efficient solutions generated at the previous steps are denoted by X_r^*, $r = 1, \dots, R$, i. e. $\cup_{i=1}^{j-1} E_i(P) = \{X_r^* : r = 1, \dots, R\}$. In this jth step, the following problem is solved

  $$(\mathrm{P}_j) \quad \begin{cases} \max\limits_{X\in D} \; z_l(X) \\ \bigcap\limits_{r=1}^{R} \left(\bigcup\limits_{\substack{k=1\\k\neq l}}^{K} z_k(X) \geq z_k(X_r^*) + 1 \right). \end{cases}$$

  The new set of constraints represents the requirement that a solution to (P_j) be better on some objective $k \neq l$ for each efficient solution X_r^* generated during the previous steps; an example of implementation of theseconstraints is given in [5]. The set of solutions E_j(P) is then defined as $E_j(\mathrm{P}) = E(\mathrm{P}_j) \cap E(\mathrm{P})$, where $E(\mathrm{P}_j)$ is the set of all optimal solutions of (P_j).

The procedure continues until, at some iteration J, the problem (P_J) becomes infeasible; at this time $E(\mathrm{P}) = \cup_{j=0}^{J-1} E_j(\mathrm{P})$.

Kiziltan–Yucaoglu Method

See [4]. This is a direct adaptation to a multi-objective framework of the well-known *Balas algorithm* for the ILP problem with binary variables.

At node S^r of the *branch and bound* scheme, the following problem is considered:

$$\begin{cases} '\max' & \sum\limits_{j\in F^r} c_j x_j + \sum\limits_{j\in B^r} c_j \\ \text{s.t.} & \sum\limits_{j\in F} t_j x_j \leq d^r \\ & x_j = (0,1) \\ \text{where} & B^r \text{ is the index set of variables} \\ & \quad \text{assigned the value one} \\ & F^r \text{ is the index of free variables} \\ & d^r = d - \sum\limits_{j\in B^r} t_j \\ & t_j \text{ is the } j\text{th column of } T \\ & c_j \text{ is the vector of components } c_j^{(k)}. \end{cases}$$

The node S^r is called *feasible* when $d^r \geq 0$ and *infeasible* otherwise. The three *basic rules of the branch and bound* algorithm are:

- (bounding rule) A lower and upper bound vector, $\underline{Z}^r$ and $\overline{Z}^r$, respectively, are defined as

$$\underline{Z}^r = \sum_{j \in B^r} c_j ,$$
$$\overline{Z}^r = \underline{Z}^r + Y^r ,$$

  where $Y^r_k = \sum_{j \in F^r} \max\{0, c^k_j\}$. The vector $\underline{Z}^r$ is added to a list $\widehat{E}$ of existing lower bounds if $\underline{Z}^r$ is not dominated by any of the existing vectors of $\widehat{E}$. At the same time, any vector of $\widehat{E}$ dominated by $\underline{Z}^r$ is discarded.
- (fathoming rules) In the multi-objective case, the feasibility of a node is no longer a sufficient condition for fathoming it. The three general fathoming conditions are:
  - $\overline{Z}^r$ is dominated by some vector of $\widehat{E}$;
  - the node S^r is feasible and $\underline{Z}^r = \overline{Z}^r$;
  - the node S^r is unfeasible and $\sum_{j \in F^r} \min(0, t_{ij}) > d^r_i$ for some $i = 1, \ldots, m$.

  The usual backtracking rules are applied.
- (branching rule) A variable $x_l \in F^r$ is selected to be the branching variable.
  - If the node S^r is feasible, $l \in \{j \in F^r : c_j \not\leq 0\}$.
  - Otherwise, index l is selected by the *minimum unfeasibility criterion*:

$$\min_{j \in F^r} \sum_{i=1}^{m} \max\left(0, -d^r_i + t_{ij}\right) .$$

When the explicit enumeration is complete, $E(\text{P}) = \widehat{E}$.

Interactive Methods

Such methods are particularly important to solve multi-objective applications. The general idea is to determine progressively a good compromise solution integrating the preferences of the decision maker.

The dialog with the decision maker consist of a succession of 'calculation phase' managed by the model and 'information phase' managed by the decision maker.

At each calculation phase, one or several new efficient solutions are determined taking into account the information given by the decision maker at the preceding information phase. At each information phase, a few number of easy questions are asked to the decision maker to collect information about its preferences in regard to the new solutions.

Gonzalez–Reeves–Franz Algorithm

See [3]. In this method a set $\widetilde{E}$ of K efficient solutions is selected and updated in each algorithm step according to the decision maker's preferences. At the end of the procedure, $\widetilde{E}$ will contain the most preferred solutions. The method is divided in two stages: in the first one, the supported efficient solutions are considered, while the second one deals with nonsupported efficient solutions.

- (Stage 1): Determination of the best supported efficient solutions. $\widetilde{E}$ is initialized with K optimal solutions of the K single objective ILP problems. Let us denote by $\widetilde{Z}$ the K corresponding points in the objective space of the solution of $\widetilde{E}$. At each iteration, a linear direction of search $G(X)$ is build:$G(X)$ is the inverse mapping of the hyperplane defined by the points of $\widetilde{Z}$ in the objective space into the decision space. A new supported efficient solution X^* is determined by solving the single objective ILP problem $\max_{X \in D} G(X)$ and Z^* is the corresponding point in the objective space. Then:
  - if $Z^* \notin \widetilde{Z}$ and the decision maker prefers solution X^* to at least one solution of $\widetilde{E}$: the least preferred solution is replaced in $\widetilde{E}$ by X^* and a new iteration is performed;
  - if $Z^* \notin \widetilde{Z}$ and X^* is not preferred to any solution in $\widetilde{E}$: $\widetilde{E}$ is not modified and the second stage is initiated;
  - if $Z * \widetilde{Z}$: $\widetilde{Z}$ defines a face of the efficient surface and the second stage is initiated.
- (Stage 2): Introduction of the best non supported solutions. We will not give details about this second stage (see [3] or [10]); letus just say that it is performed in the same spirit but considering the single objective problem

$$\begin{cases} \max & G(X) \\ & X \in D \\ & G(X) \leq \widetilde{G} - \varepsilon \quad \text{with } \varepsilon > 0 \end{cases}$$

  where $\widetilde{G}$ is the optimal value obtained for the last function $G(X)$ considered.

Steuer–Choo Method

See [9]. Several interactive approaches of MOLP problems can also be applied to MOILP; among them, we mention only the Steuer–Choo method, which is a very general procedure based on problem (P_λ^T) defined in the introduction.

The first iteration uses a widely dispersed group of λ weighting vectors to sample the set of efficient solutions. The sample is obtained by solving problem (P_λ^T) for each of the λ values in the set. Then the decision maker is asked to identify the most preferred solution $X^{(1)}$ among the sample. At iteration j, a more refined grid of weighting vectors λ is used to sample the set of efficient solution in the neighborhood of the point $z_k(X^{(j)})$ $(k = 1, \dots, K)$ in the objective space. Again the sample is obtained by solving several problems (P_λ^T) and the most preferred solution $X^{(j+1)}$ is selected. The procedure continues using increasingly finer sampling until the solution is deemed to be acceptable.

The MOMIX Method

(See [6].) The main characteristic of this method is the use of an interactive branch and bound concept – initially introduced in [7] – to design the interactive phase.

- (First compromise): The following minimax optimization, with $m = 1$, is performed to determined the compromise $\widetilde{X}^{(1)}$:

$$(P^m) \quad \begin{cases} \min & \delta \\ \forall k & \Pi_k^{(m)}(M_k^{(m)} - z_k(X)) \leq \delta, \\ & X \in D^{(m)} \end{cases}$$

  where
  - $D^{(1)} \equiv D$;
  - $[m_k^{(1)}, M_k^{(1)}]$ are the variation intervals of the criteria k, provided by the pay-off table (see [8]);
  - $\Pi_k^{(1)}$ are certain normalizing weights taking into account these variation intervals (see [8]).

Remark 1 If the optimal solution is not unique, an augmented weighted Tchebychev distance is required in order to obtain an efficient first solution.

- (Interactive phases): There are integrated in an interactive branch and bound tree; a first step (a *depth-first* progression in the tree) leads to the determination of a first good compromise; the second step (a *backtracking* procedure) confirms the degree of satisfaction achieved by the decision maker or it finds a better compromise if necessary.
  - (Depth first progression): For $m \geq 1$, let at the mth iteration
    1) $\widetilde{X}^{(m)}$ be the mth compromise;
    2) $z_k^{(m)}$ be the corresponding values of the criteria;
    3) $[m_k^{(m)}, M_k^{(m)}]$ be the variation intervals of the criteria; and
    4) $\Pi_k^{(m)}$ be the weight of the criteria.

    The decision maker has to choose, at this mth iteration, the criterion $l_m(1) \in \{k : k = 1, \dots, K\}$ he is willing to improve in priority. Then a new constraint is introduced so that the feasible set becomes $D^{(m+1)} \equiv D^{(m)} \cap \{z_{lm}(1)(X) > z_{lm}(1)^{(m)}\}$ Further, the variation intervals $[m_k^{(m+1)}, M_k^{(m+1)}]$ and the weights $\Pi_k^{(m+1)}$ are updated on the new feasible set $D^{(m+1)}$. The new compromise $\widetilde{X}^{(m+1)}$ is obtained by solving the problem (P^{m+1}).

    Different tests allow to terminate this first step. The node $(m+1)$ is fathomed if one of the following conditions is verified:
    a) $D^{(m+1)} = \emptyset$;
    b) $M_k^{(m+1)} - m_k^{(m+1)} \leq \epsilon_k \ \forall k$;
    c) the vector $\widehat{Z}$ of the *incumbent values* (values of the criteria for the best compromise already determined) is preferred to the new ideal point (of component $M_k^{(m+1)}$).

    The first step of the procedure is stopped if either more than q successive iterations do not bring an improvement of the incumbent point $\widehat{Z}$ or more than Q iterations have been performed.

    Note that the parameters ϵ_k, q and Q are fixed in the agreement with the decision maker.

  c) (Backtracking procedure): It can be hoped that the appropriate choice of the criterion $z_{lm}(1)$, at each level m of the depth-first progression, has been made so that at the end of the first step, a good compromise has been found.

  Nevertheless, it is worth examining some other parts of the tree to confirm the satisfaction of the decision maker. The complete tree is generated in the following manner: at each level, K subnodes are introduced by successively adding the constraints:

$$\begin{array}{llllll} z_{l_m(1)}(X) > z^{(m)}_{l_m(1)}, & & & & & \\ z_{l_m(2)}(X) > z^{(m)}_{l_m(2)}; & z_{l_m(1)}(X) \leq z^{(m)}_{l_m(1)}, & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ z_{l_m(K)}(X) > z^{(m)}_{l_m(K)}; & z_{l_m(k)}(X) \leq z^{(m)}_{l_m(k)}, & & & & \end{array}$$

for all $k = 1, \dots, K - 1$, where $l_m(k) \in \{k\colon k = 1, \dots, K\}$ is the kth objective that the decision maker wants to improve at the mth level of the branch and bound tree.

At each level m, the criteria are thus ordered according to the priorities of the decision maker in regard with the compromise $\widetilde{X}^{(m)}$.

The usual backtracking procedure is applied; yet it seems unnecessary to explore the whole tree. Indeed, the subnode $k > \overline{K}$ of each branching correspond to a simultaneous relaxation of those criteria $l_m(k)$, $k \leq \overline{K}$, the decision maker wants to improve in priority!

Therefore, the subnodes $k > \overline{K} = 2$ or 3, for instance, do almost certainly not bring any improved solutions.

The fathoming tests and the stopping tests are again applied in this second step.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Branch and Price: Integer Programming with Column Generation
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Mixed Integer Classification Problems
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-dependent Traveling Salesman Problem

References

1. Bowman VJ Jr (1976) On the relationship of the Tchebycheff norm of the efficient frontier of multi-criteria objectives. In: Thiriez H, Zionts S (eds) Multiple Criteria Decision Making. Springer, Berlin, pp 76–85
2. Burkard RE (1981) A relationship between optimality and efficiency in multiple criteria 0–1 programming problems. Comput Oper Res 8:241–247
3. Gonzalez JJ, Reeves GR, Franz LS (1985) An interactive procedure for solving multiple objective integer Linear programming problems. In: Haimes Y, Chankong V (eds) Decision Making with Multiple Objectives. Springer, Berlin, pp 250–260
4. Kiziltan G, Yucaoglu E (1983) An algorithm for multiobjective zero-one linear programming. Managem Sci 29(12):1444–1453
5. Klein D, Hannan E (1982) An algorithm for the multiple objective integer linear programming. EJOR 9(4):378–385
6. L'Hoir H, Teghem J (1995) Portfolio selection by MOLP using an interactive branch and bound. Found Computing and Decision Sci 20(3):175–185
7. Marcotte O, Soland R (1986) An interactive branch and bound algorithm for multiple criteria optimization. Managem Sci 32(1):61–75

8. Steuer RE (1986) Multiple criteria optimization theory, computation and applications. Wiley, New York
9. Steuer RE, Choo E-U (1983) An interactive method weighted Tchebycheff procedure for multiple objective programming. Math Program 26:326–344
10. Teghem J, Kunsch P (1986) Interactive method for multi-objective integer linear programming. In: Fandel G, et al. (eds) Large Scale Modelling and Interactive decision analysis. Springer, Berlin 75–87
11. Teghem J, Kunsch P (1986) A survey of techniques for finding efficient solutions to multi-objective integer linear programming. Asia–Pacific J Oper Res 3:1195–106

Multi-objective Mixed Integer Programming

Maria João Alves, João Clímaco
Fac. Economics, University Coimbra and INESC, Coimbra, Portugal

MSC2000: 90C29, 90C11

Article Outline

Keywords

Multi-objective mathematical programming; Multicriteria analysis; Interactive method

A *multi-objective (multicriteria) mixed integer programming*(*MOMIP*) problem is a mathematical programming problem that considers more than one objective function and some but not all the variables are constrained to be integer valued. The integer variables can either be binary or take on general integer values. The problem may be stated as follows:

$$\begin{cases} \max & z_1 = f_1(x) \\ & \vdots \\ \max & z_k = f_k(x) \\ \text{s.t.} & x \in X \end{cases}$$

where $X \subset \mathbf{R}^n$ denotes the nonconvex set of feasible solutions defined by a set of functional constraints, $x \geq 0$ and x_j integer $j \in J \subset \{1, \dots, n\}$. It is assumed that X is compact (closed and bounded) and nonempty.

Although a MOMIP problem may be nonlinear, models with linear constraints and linear objective functions have been more often considered. In a multi-objective mixed integer linear programming (*MOMILP*) problem, the functional constraints can be defined as $Ax \leq b$, and the objective functions $f_i(x) = c_i x$, $i = 1, \dots, k$, where A is a $m \times n$ matrix, b is a m-dimensional column vector and c_i, $i = 1, \dots, k$, are n-dimensional row vectors.

Multi-objective mixed integer programming is very useful for many areas of application such as communication, transportation and location, among others. Integer variables are required in a real-world model whenever it is sought to incorporate discrete phenomena; for instance, investment choices, production levels, fixed charges, logical conditions or disjunctive constraints. However, research on MOMIP has been rather limited. Concerning multi-objective mathematical programming, most research efforts have been so far devoted to linear programming with continuous variables (MOLP). The introduction of discrete phenomena into multi-objective models leads to all-integer or mixed integer problems that are more difficult to tackle. They can not be handled by most MOLP approaches because the feasible set is no longer convex. Also, there are multi-objective approaches designed for all-integer problems that do not apply to the mixed integer case. Therefore, even for the linear case, techniques for dealing with multi-objective mixed integer programming involve more than the combination of MOLP with multi-objective integer programming techniques.

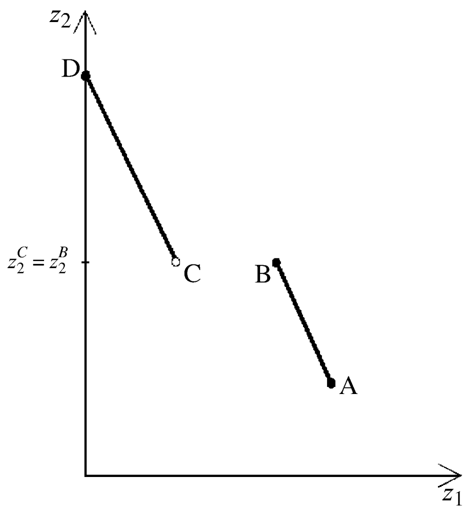
Efficiency and Nondominance

The concept of *efficiency* (or *nondominance*) in MOMIP is defined as usually for multi-objective mathematical programming: A solution $\overline{x} \in X$ is *efficient* if and only if it does not exist another $x \in X$ such that $f_i(x) \geq f_i(\overline{x})$ for all $i \in \{1, \ldots, k\}$ and $f_i(x) > f_i(\overline{x})$ for at least one i. A solution $\overline{x} \in X$ is *weakly efficient* if and only if it does not exist another $x \in X$ such that $f_i(x) > f_i(\overline{x})$ for all $i \in \{1, \ldots, k\}$.

Let $Z \subset \mathbf{R}^k$ be the image of the feasible region X in the criterion (objective function) space. A criterion point $\overline{z} \in Z$ corresponding to a (weakly) efficient solution $\overline{x} \in X$ is called (weakly) *nondominated*. The designations 'efficient', 'nondominated' and 'Pareto optimal' are often used as synonyms.

Supported and Unsupported Nondominated Solutions

Since the feasible region is nonconvex, unsupported nondominated points/solutions may exist in a MOMIP problem. A nondominated point $\overline{z} \in Z$ is *unsupported* if it is dominated by a convex combination (which does not belong to Z) of other nondominated criterion points (belonging to Z). In Fig. 1 the line segment from A to B plus D is the set of supported nondominated criterion points. The line segment from C to D excluding C and D is the set of unsupported nondominated criterion points. Note that convex combinations of B and D dominate the line segment from C to D, excluding D. C is a weakly nondominated solution.



Multi-objective Mixed Integer Programming, Figure 1
Nondominated criterion points of a MOMILP problem

Characterization of the Nondominated Set

Unlike MOLP, the nondominated (or efficient) set of MOMIP problems can not be fully determined by parameterizing on λ the weighted-sums program:

$$(\mathrm{P}_\lambda) \quad \begin{cases} \max & \left\{ \sum_{i=1}^{k} \lambda_i f_i(x) \colon \ x \in X \right\} \\ \text{where} & \lambda \in \Lambda \,. \end{cases}$$

Here,

$$\Lambda = \left\{ \lambda \in \mathbb{R}^k \colon \ \begin{array}{l} \lambda_i > 0 \quad \forall i, \\ \sum_{i=1}^{k} \lambda_i = 1 \end{array} \right\} .$$

The unsupported nondominated solutions cannot be reached even if the complete parameterization on λ is attempted.

Researchers on multi-objective mathematical programming early recognized this fact and stated other characterizations for the nondominated set that fit MOMIP and, in particular, MOMILP problems. Basically, two main characterizations are defined. One consists of introducing additional constraints into the *weighted-sums program*. Generally, these constraints impose bounds on the objective function values. This form of characterization may be regarded as a particularization of the general characterization provided by R.M. Soland [13]. The other is based on the Tchebycheff theory whose theoretical foundation originated from V.J. Bowman [3]. More details about these characterizations and on how they provide the computation of nondominated solutions will be given later. Although providing very important theoretical results, the characterizations of the nondominated set do not offer an explicit means to provide decision support for MOMIP problems. However, some authors have developed decision support methods for these problems.

Interactive Versus Noninteractive Methods

Methods may be either *noninteractive* (in general, generating methods designed to find the whole or a subset of the nondominated solutions) or *inter-*

active (characterized by phases of human intervention alternated with phases of computation). Generating methods for MOMIP problems usually require an excessive amount of computational resources, both in processing time and storage capacity. Even specialized generating algorithms developed just for bi-objective problems, which profit from graphical representations on the criterion space, tend to be inadequate to deal with large problems. Nevertheless, the distinction between interactive and generating methods is not always clear. Some approaches attempt to find a representative subset of the nondominated set (generating methods according to the above definition) and would be easily embodied in an interactive framework. The bi-objective method of R. Solanki [14] may be regarded as an example of such an approach.

Taking into account the difficulties mentioned above, and the large number of nondominated solutions in many problems, special attention to interactive methods will be paid. First of all, a short remark is made about the major paradigms followed by the authors of interactive methods. Some authors admit that the decision maker's (DM) preferences can be represented by an *implicit utility function*. The interactive process consists in building a protocol of interaction aiming to discover the optimum (or an approximation of it) of that implicit utility function. The convergence to this optimum requires no contradictions in the DM's responses given throughout the interactive process.

In contrast with implicit utility function approaches, the *open communication* approaches are based on a progressive and selective learning of the nondominated set. The terminology of open communication is inspired on the concept of open exchange, defined by P. Feyerbend [6]. Such multi-objective approaches are not intended to converge to any 'best' compromise solution but to help the DM to avoid the search for nondominated solutions he/she is not at all interested in. There are no irrevocable decisions during the whole process and the DM is always allowed to go 'backwards' at a later interaction. So, at each interaction, the DM is only asked to give some indications on what direction the search for nondominated solutions must follow, or occasionally to introduce additional constraints. The process only finishes when the DM considers to have gained sufficient insight into the nondominated solution set. Using the terminology of B. Roy [12], 'convergence' must give place to 'creation'. The interactive process is a constructive process, not the search for something 'pre-existent'.

Although we personally prefer the open communication methods, we will include in the next section a tentative classification of both, drawing out some differences and similarities between them. We adopt this perspective because this question is not specific to mixed integer programming and arguments pro or against each approach, besides being subjective, are the same as in other multi-objective programming fields. Furthermore, since MOMIP is still in its early steps, no behavioral studies exist addressing the use of procedures within this context.

As we have mentioned before, research on MOMIP has been rather scarce in comparison to other fields of the multi-objective mathematical programming, namely in MOLP. We will mention herein some well-known methods specially designed for MOMIP or far more generally applicable.

Computing Processes and Their Use in Interactive Methods

Weighted-Sums Programs with Additional Constraints

The introduction of bounds on the objective function values into the *weighted-sums program* (P_λ) enables this program to also compute unsupported nondominated solutions:

$$(\mathrm{P}_{\lambda,g}) \quad \max\left\{\sum_{j=1}^{k}\lambda_i f_i(x)\colon\ x\in X,\ f(x)\geq g\right\},$$

where $f(x)=(f_1(x),\dots,f_k(x))$, $\lambda\in\Lambda$ and g is a vector of objective bounds. Besides the fact that every solution obtained by $(\mathrm{P}_{\lambda,g})$ is nondominated, there always exists a $g\in\mathbf{R}^k$ such that $(\mathrm{P}_{\lambda,g})$ yields a particular nondominated solution. Other types of additional constraints can also be used.

A scalarizing program which consists of the weighted-sums program combined with additional constraints is used for computing nondominated solutions in the interactive branch and bound method of B. Villarreal et al. [18]. The additional constraints are bounds imposed on integer variables by the branching process. This method, which is devoted to MOMILP problems, received later improvements in [8] and [11]. Starting by applying the well-known (MOLP) *Zionts–Wallenius procedure* to the linear relaxation of the MOMILP problem, the method then employs a branch and bound phase until an integer solution that satisfies the DM is achieved. An implicit utility function is assumed and the DM's preferences are assessed using pairwise evaluations of decision alternatives and trade-off analysis. In light of the DM's underlying utility function, decisions on whether to apply again the Zionts–Wallenius procedure to the linear relaxation of a candidate multi-objective subproblem, or to continue to branch by appending a constraint on a variable, are successively made.

Another method that uses particular forms of $(\mathrm{P}_{\lambda,g})$ to compute nondominated solutions is due to Y. Aksoy [1]. This is an interactive method for bicriterion mixed integer programs that employs a branch and bound scheme to divide the subset of nondominated solutions considered at each node into two disjoint subsets. The branching process seeks to bisect the range of nondominated values for z_2 at the node under consideration, checking whether a nondominated point exists whose value for z_2 is in the middle of the range. If no such solution exists, that subset is divided using two nondominated points whose values for z_2 are the closest (one up and the other down) to the middle value. These nondominated solutions are obtained by solving $(\mathrm{P}_{\lambda,g})$ optimizing one objective function and bounding the other. The interactive process requires the DM to make pairwise comparisons in order to determine the branching node and to adjust the incumbent solution to the preferred nondominated solution. It is assumed that the DM's preferences are consistent, transitive and invariant over the process aiming to optimize the DM's implicit utility function.

C. Ferreira et al. [5] proposed a decision support system for bicriterion mixed integer programs. The interactive process follows an open communication protocol asking the DM to specify bounds for the objective function values. These bounds are input into $(\mathrm{P}_{\lambda,g})$ defining subregions to carry on the search for nondominated solutions. Some objective space regions are progressively eliminated either by dominance or infeasibility.

Tchebycheff and Achievement Scalarizing Programs

Bowman [3] proved that the parameterization on w of $\min_{x\in X} \|\overline{f} - f(x)\|_w$ generates the nondominated set, where $w_i \geq 0$ for all i, $\sum_{i=1}^{k} w_i = 1$, $\overline{f}$ is a criterion point such that $\overline{f} > f(x)$ for all $x \in X$ and $\|\overline{f} - f(x)\|_w$ denotes the *w-weighted Tchebycheff metric*, that is, $\max_{1\leq i\leq k}\{w_i|\overline{f}_i - f_i(x)|\}$. This scalarizing program is equivalent to

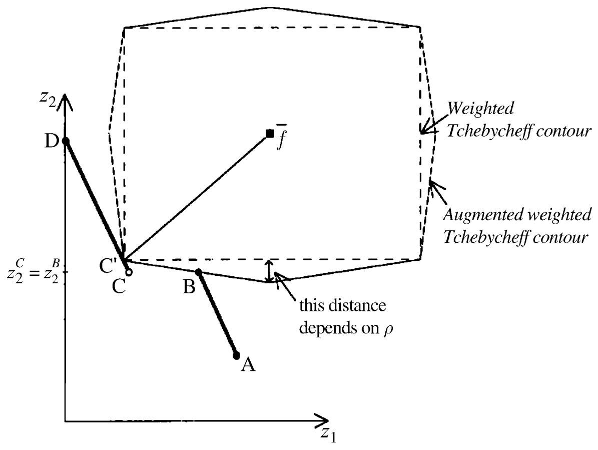
$$(\mathrm{T}_w)\quad \begin{cases} \min & \alpha \\ \text{s.t.} & w_i\left(\overline{f}_i - f_i(x)\right) \leq \alpha, \quad 1 \leq i \leq k, \\ & x \in X, \quad \alpha \geq 0. \end{cases}$$

(T_w) may yield weakly nondominated solutions (for instance, point C in Fig. 1). Replacing the objective function in (T_w) by $\alpha - \rho \sum_{i=1}^{k} f_i(x)$ with ρ a small positive value, all the solutions returned by this augmented weighted Tchebycheff program are nondominated. R.E. Steuer and E.-U. Choo [16] proved that there are always ρ small enough that enable to reach all the nondominated set for the finite-discrete and polyhedral feasible region cases.

Concerning the MOMIP case, although there may be portions of the nondominated set that the program is unable to compute, even considering ρ very small (for example, the line segment from C to C' in Fig. 2, for a given ρ), this characterization is still possible in practice. Note that ρ can be set so small that the DM is unable to discriminate between those solutions and a nearby weakly nondominated solution (this corresponds to C' getting closer to C in Fig. 2).

In [16] and [15] a lexicographic weighted Tchebycheff program is proposed for the nonlinear and infinite-discrete feasible region cases to overcome this drawback of the augmented weighted Tchebycheff program. The lexicographic approach can also be applied to the mixed integer (linear) case. However, it is more difficult to implement since two stages of optimization are employed. At the first stage only α is minimized. When the first stage results in alternative optima, a second stage is required. It consists of minimizing $-\sum_{i=1}^{k} f_i(x)$ over the solutions that minimize α in order to eliminate the weakly nondominated solutions.

Besides (T_w) (either the augmented or the lexicographic forms), there are other similar approaches that also allow to characterize the nondominated set of



z_2
D
$\bar{f}$
Weighted Tchebycheff contour
Augmented weighted Tchebycheff contour
$z_2^C = z_2^B$
C'
C
B
this distance depends on ρ
A
z_1

Multi-objective Mixed Integer Programming, Figure 2
Illustration of the augmented weighted Tchebycheff metric

multi-objective mixed integer programs. An approach of this type consists in discarding the w-vector or fixing it and varying $\overline{f}$, the criterion reference point that represents the DM's aspiration levels. This scalarizing program can be denoted by $(\mathrm{T}_{\overline{f}})$. There always exist reference points satisfying $\overline{f} > f(x)$ for all $x \in X$, such that $(\mathrm{T}_{\overline{f}})$ produces a particular nondominated solution $\overline{z} = f(\overline{x})$. The variation of $\overline{f}$ can be done according to a vector direction θ, leading to $(\mathrm{T}_{\overline{f}+\theta})$. The reference points are thus projected onto the nondominated set. Reference points that do not satisfy the condition $\overline{f} > f(x)$ for all $x \in X$ may also be considered provided that the α variable is defined without sign restriction. This corresponds to the minimization of a distance from Z to the reference point if the latter is not attainable and to the maximization of such a distance if the reference point is attainable. If reference or aspiration levels are used as controlling parameters, the (weighted) Tchebycheff metric changes its form of dependence on controlling parameters and should be interpreted as an *achievement function* [9].

Like (T_w), the simplest form of $(\mathrm{T}_{\overline{f}})$ may produce weakly nondominated solutions. The augmented form is a good substitute in practice and the lexicographic approach guarantees that all nondominated solutions can be reached. In what follows, let (T.) denote either the simplest, the augmented or the lexicographic form.

Scalarizing programs (T_w), $(\mathrm{T}_{\overline{f}})$ and their extensions or slight different formulations are used to generate nondominated solutions in several (interactive) methods proposed in literature, namely in the following ones.

Steuer and Choo [16] proposed a general purpose multi-objective programming interactive method that assumes an implicit DM's utility function without any special restriction on shape. The strategy of the interactive procedure is to sample series of progressively smaller subsets of nondominated solutions. At each interaction, the DM selects his/her preferred solution from a sample of nondominated solutions obtained from (T_w) with several w-vectors and the ideal criterion point in the role of $\overline{f}$. The solution preferred by the DM provides information to tighten the set of w-vectors for the next interaction. The procedure terminates when a nondominated criterion point sufficiently close to the optimal criterion point of the underlying utility function is found.

Solanki's method [14], which is designed for bi-objective mixed integer linear programs, is an adaptation of the noninferior set estimation (NISE) method developed by J.L. Cohon for bi-objective linear programs. It seeks to generate a representative subset of nondominated solutions by combining the NISE's key features with weighted Tchebycheff scalarizing programs. At each iteration, a new nondominated solution, say z^3, is computed by solving (T_w) for specific w and $\overline{f}$, assuring that z^3 belongs to the region between a pair of nondominated criterion points previously determined, say (z^1, z^2). This pair is then replaced by (z^1, z^3) and (z^3, z^2). The approximation of the nondominated surface is progressively improved, thus decreasing the 'errors' associated with the approximate representation of the pairs. This 'error' is measured by the largest range of the two objectives for the points forming the pair. The algorithm finishes when the maximum 'error' is lower than a predefined maximum allowable 'error'.

Another interactive method capable of solving MOMIP problems was developed by A. Durso [4]. This method employs a branching scheme considering progressively smaller portions of the nondominated set by imposing lower bounds on the criterion values. At each interaction, the k nondominated solutions that define the (quasi)ideal criterion point for each new node are calculated. The DM is then asked to select the node for branching by choosing the preferred ideal point. The branching process begins by solving an equally weighted augmented Tchebycheff program to determine a 'centralized' nondominated point for the subset of the node under exploration. Once the DM chooses the most preferred of the $k + 1$ nondominated points already known for this node, say $\widehat{z}$, up to k new nodes (children) are created. Each child inherits its parent's bounding constraints and uses $\widehat{z}$ to further restrict one of them. Thus, the ith child restricts the ith criterion by imposing $f_i(x) \geq \widehat{z}_i + \delta$ with δ small positive. This approach may be regarded as an open communication procedure that terminates when the DM is satisfied with the incumbent solution (the preferred nondominated solution obtained so far).

M.J. Alves and J. Clímaco [2] proposed a MOMILP open communication interactive approach. It combines the Tchebycheff theory with the traditional branch and bound technique for solving single-objective mixed integer programs. At each interaction, the DM specifies either a reference point $\overline{f}$, which is input in $(\mathrm{T}_{\overline{f}})$ to compute a nondominated solution via branch and bound, or just selects an objective function, say f_j, he/she wants to improve with respect to the previous nondominated solution. In the latter case, the reference point is automatically adjusted by increasing the jth component of $\overline{f}$ keeping the others equal, in order to produce new nondominated solutions (directional search) more suited to the DM's preferences. This involves an iterative process of sensitivity analysis and operations to update the branch and bound tree. The sensitivity analysis takes advantage of the special behavior of the parametric scalarizing program $(\mathrm{T}_{\overline{f}+\theta})$. It returns a value $\overline{\theta}_j > 0$ such that the structure of the previous branch and bound tree remains unchanged for variations in $\overline{f}_j$ up to $\overline{f}_j + \overline{\theta}_j$. Therefore, reference points $\overline{f} + \theta = (\overline{f}_1, \ldots, \overline{f}_j + \theta_j, \ldots, \overline{f}_k)$ with $\theta_j \leq \overline{\theta}_j$ lead to nondominated solutions that may be obtained in a straightforward way. If the DM wishes to continue the search in the same direction, a slight increase over $\overline{\theta}_j$, say $\overline{\theta}_j + \epsilon$, is first considered. In this case, the previous sensitivity analysis also returns the best candidate node, i. e., an ancestor of the node that will produce the next nondominated solution. The previous branch and bound tree is thus used to proceed to the next computations. Since further branching is usually required, an attempt is made to simplify the tree before enlarging it. The underlying idea is to avoid an evergrowing tree. This simplification means cutting off parts of the tree linked by branching constraints no longer active. In sum, this approach brings together sensitivity analysis phases meant to adjust the reference point and simplification/branching operations of the search tree to compute nondominated solutions. This process is repeated as long as the DM wishes to continue the directional search or if the reference point has not been adjusted enough to yield a nondominated solution different from the previous one (a situation that occurs more often in all-integer programs than in mixed integer models). Computational experiments have shown that this multi-objective approach succeeds in performing directional searches. The times of computing phases using simplification/branching operations have been significantly reduced by this strategy.

Some researchers have developed other methods for multi-objective integer programming that are also applicable to the mixed integer case. Good examples of such approaches are those in [10,17] and [7]. In our opinion, they all are open communication procedures that share some key features, namely the concept of projecting a reference direction onto the nondominated surface (although this procedure is used in different ways) and the type of information required about the DM's preferences. This information lies fundamentally in the specification of aspiration levels for the objective function values (reference points). Some of these approaches are continuous/integer ([7,10]) working almost all the time with nondominated continuous solutions of the linear relaxation of the problem. Whenever the DM finds a satisfactory continuous solution, an integer nondominated solution close to it is then computed.

Conclusions and Future Developments

Most methods developed so far for MOMIP problems require an excessive amount of computational effort, or require too much cognitive load from the DM, or only address bi-objective problems. In addition, computational experience with real-world applications is lacking. Although interesting or promising approaches have been developed, further research efforts must be made in order to build effective interactive methods able to handle real-sized problems.

See also

► Branch and Price: Integer Programming with Column Generation

- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Graph Coloring
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-dependent Traveling Salesman Problem

References

1. Aksoy Y (1990) An interactive branch-and-bound algorithm for bicriterion nonconvex/mixed integer programming. Naval Res Logist 37:403–417
2. Alves MJ, Clímaco J (2000) An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound. Europ J Oper Res 124(3):478–494
3. Bowman VJ (1976) On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives. In: Thiriez H, Zionts S (eds) Multiple Criteria Decision Making. Lecture notes Economics and Math Systems. Springer, Berlin, pp 76–86
4. Durso A (1992) An interactive combined branch-and-bound/Tchebycheff algorithm for multiple criteria optimization. In: Goicoechea A, Duckstein L, Zionts S (eds) Multiple Criteria Decision Making, Proc 9th Internat Conf. Springer, Berlin, pp 107–122
5. Ferreira C, Santos BS, Captivo ME, Clímaco J, Silva CC (1996) Multiobjective location of unwelcome or central facilities involving environmental aspects: A prototype of a decision support system. Belgian J Oper Res Statist Comput Sci 36(2–3):159–172
6. Feyerabend P (1975) Against method. Verso, London
7. Karaivanova J, Korhonen P, Narula S, Wallenius J, Vassilev V (1995) A reference direction approach to multiple objective integer linear programming. Europ J Oper Res 81:176–187
8. Karwan MH, Zionts S, Villarreal B, Ramesh R (1985) An improved interactive multicriteria integer programming algorithm. In: Haimes Y, Chankong V (eds) Decision Making with Multiple Objectives. Lecture notes Economics and Math Systems. Springer, Berlin, pp 261–271
9. Lewandowski A, Wierzbicki A (1988) Aspiration based decision analysis and support. Part I: Theoretical and methodological backgrounds. WP-88-03, Internat Inst Appl Systems Anal (IIASA), Austria
10. Narula SC, Vassilev V (1994) An interactive algorithm for solving multiple objective integer linear programming problems. Europ J Oper Res 79:443–450
11. Ramesh R, Zionts S, Karwan MH (1986) A class of practical interactive branch and bound algorithms for multicriteria integer programming. Europ J Oper Res 26:161–172
12. Roy B (1987) Meaning and validity of interactive procedures as tools for decision making. Europ J Oper Res 31:297–303
13. Soland RM (1979) Multicriteria optimization: A general characterization of efficient solutions. Decision Sci 10:26–38
14. Solanki R (1991) Generating the noninferior set in mixed integer biobjective linear programs: An application to a location problem. Comput Oper Res 18(1):1–15
15. Steuer R (1986) Multiple criteria optimization: Theory, computation and application. Wiley, New York
16. Steuer R, Choo E-U (1983) An interactive weighted Tchebycheff procedure for multiple objective programming. Math Program 26:326–344
17. Vassilev V, Narula SC (1993) A reference direction algorithm for solving multiple objective integer linear programming problems. J Oper Res Soc 44(12):1201–1209
18. Villarreal B, Karwan H M, Zionts S (1980) An interactive branch and bound procedure for multicriterion integer linear programming. In: Fandel G, Gal T (eds) Multiple Criteria Decision Making: Theory and Application. Lecture notes Economics and Math Systems. Springer, Berlin, pp 448–467

Multi-objective Optimization and Decision Support Systems

SERPIL SAYIN
Koç University, İstanbul, Turkey

MSC2000: 90B50, 90C29, 65K05, 90C05, 91B06

Article Outline

Keywords

Multiple criteria decision making; Vector optimization; Efficient solution; Decision support

Multiple criteria decision making (*MCDM*) refers to the explicit incorporation of more than one evaluation criteria into a decision problem. MCDM has been a very active field of research roughly since the 1970s. Although boundaries might be fuzzy and overlapping, multicriteria decision analysis (studying the problem of identifying the 'most-preferred' among a finite discrete set of alternatives), multi-attribute utility theory (using utility functions explicitly to model a decision maker's preferences) and multi-objective optimization (modeling the decision problem within a mathematical programming framework) have emerged as major fields of interest under MCDM. For more information on the general field of MCDM, see [21].

Multi-objective mathematical programming provides a flexible modeling framework that allows for simultaneous optimization of more than one objective function over a feasible set. Mathematically, the multi-objective optimization problem can be expressed as:

$$\text{(MOO)} \quad \begin{cases} \max & f(x), \\ \text{s.t.} & x \in X, \end{cases}$$

where $X \subseteq \mathbf{R}^n$ is the set of feasible alternatives and $f = (f_1, \dots, f_p)\colon \mathbf{R}^n \to \mathbf{R}^p$, $p \geq 2$, is a vector-valued function. Note that X can be any set, continuous or discrete, expressed through constraints, and the objective function f can be of any form.

The increased flexibility provided by (MOO) also raises the question of what constitutes a solution to it. The definition of optimality is no longer valid, as each objective function would possibly yield a different optimal solution. Therefore solving the (MOO) problem is about studying the inherent trade-offs among conflicting objectives. Efficient solutions are the ones that possess the relevant trade-off information. An $x^o \in \mathbf{R}^n$ is called an *efficient solution* for the (MOO) problem if $x^o \in X$ and there exists no $x \in X$ such that $f(x) \geq f(x^o)$ with strict inequality holding for at least one component. The set of all efficient solutions of the (MOO) problem is usually denoted by X_E. As per the above definition, the most-preferred solution of the decision maker should belong to X_E, as solutions that are not efficient, the dominated ones, can be improved upon in at least one objective without worsening the others.

Since X_E is usually a big set, confining the most-preferred solution to X_E does not help identify the most-preferred solution immediately. In particular, the difficulty of defining and obtaining the most-preferred solution, the one that the decision maker would identify as the solution to the decision-making problem, and the need for the inevitable involvement of the decision maker in the solution procedure has resulted in very different solution approaches to the (MOO) problem.

Traditional Classification

The timing of the involvement of the decision maker in the solution procedure has been a crucial factor that distinguishes among various approaches to the (MOO) problem [13]. *A priori methods*, methods that use prior articulation of preferences, ask the decision maker to specify preference information prior to the application of an optimization routine. The elicitation of preference information can be directed towards deriving a utility function that describes the decision maker's preferences [14], or as in goal programming [7] and compromise programming [23], a standard model can be imposed upon the decision maker. As these methods reduce the (MOO) problem to a single-objective optimization problem and they aspire to find a single solution to it, they have received considerable recognition although their assumptions are usually restrictive.

The *interactive methods* require the interaction of the decision maker with the computer while solving a particular (MOO) problem. Usually, the idea is to construct a model that proposes solutions to the

(MOO) problem based on some initial input. The decision maker is then invited to reply to the solution by providing additional preference information. The interaction between the computer program and the decision maker continues until a satisfying solution is obtained.

Interactive methods are important in more than one way. First, they have introduced the means for practically solving a (MOO) problem [12]. Second, they help a decision maker learn about the inherent trade-offs of a problem during the solution process [5]. Third, the idea underlying the interactive methods constitutes the major motivation behind the contemporary *decision support systems*. Although interactive algorithms have encountered a certain level of acceptance from practitioners [1,20], they are not without disadvantages. They usually rely too much on the information provided by the decision maker, are not able to provide a global look at X_E, and thus at the trade-offs inherent in a problem, and they focus on finding a single solution whereas a number of solutions may be compatible with the decision maker's preferences. Moreover, their information requests may be overwhelming for the decision maker. It has been discussed that interactive methods need to address behavioral aspects of decision making [16] and concentrate on interfacing the decision maker[15] as well as broadening their model base [10]. Although they do not encompass all the raised issues, some of the interactive (MOO) algorithms have already evolved into decision support systems that provide a friendly environment for modeling as well as problem solving [17]. It can be expected that more decision support systems to solve problem (MOO) will appear in the near future.

Perhaps the most straight-forward way of approaching the (MOO) problem is as in *vector optimization* methods. Also referred to as *posterior methods*, these methods are based on the sole assumption that the decision maker prefers more to less in each objective function in (MOO) hence they propose identifying all of the efficient solutions of (MOO) and presenting them to the decision maker for the identification of the most-preferred solution. Along with theoretical findings [2,11], some vector optimization methods have been proposed; however, the methods have not gained practical recognition in general. The failure in the implementation of the proposed methods can be explained by the heavy computational requirements of these methods. Perhaps a more important factor is the difficulty of presenting the efficient set in a 'legible' way to the decision maker. Furthermore, as the efficient set is usually continuous when the feasible region is, the task of identifying the most-preferred solution is a monstrous one attributed to the decision maker.

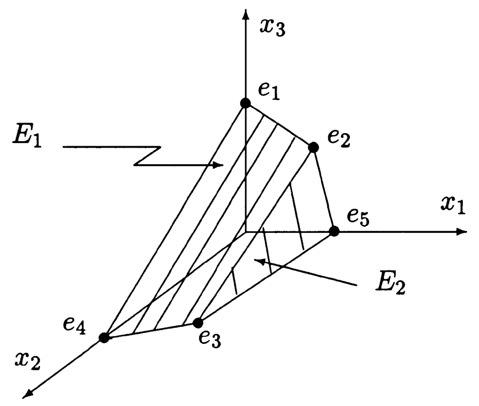
Multi-Objective Linear Programming

When (MOO) has linear objective functions and a polyhedral feasible set, the resulting problem is called a *multiple objective linear programming* (MOLP) problem. The MOLP problem has mathematical features that make it easier to characterize and obtain the efficient set compared to the more general case. More specifically, it has been shown that the efficient set of the MOLP problem consists of a collection of efficient faces of the feasible region. As faces of a polyhedron can be characterized in a number of ways, for instance as the convex hull of its extreme points if its compact, as the optimal solution set to a particular optimization problem, or as a polyhedron itself, it becomes possible to obtain and present the efficient set [9,18,22].

Yet the computational effort increases with problem size, and the (MOO) problem cannot be considered truly solved at this stage without some mechanism that helps the decision maker identify the most-preferred solution in this huge and hard-to-explore set. Most of the vector optimization methods have concentrated on finding the set of efficient extreme points of the multiple objective linear programming problem. These are usually methods that rely on simplex-like procedures or parametric searches that incorporate book-keeping mechanisms based on the fact that the set of efficient extreme points is connected. A well-known procedure that solves (MOLP) for all of its extreme points is ADBASE which was developed by R.E. Steuer [19].

Example 1 Consider the MOLP problem [18]:

$$\begin{cases} \max & x_1, x_2, x_3, \\ \text{s.t.} & 2x_1 + 3x_2 + 4x_3 \leq 12 \\ & 4x_1 + x_2 + x_3 \leq 8 \\ & x_1, x_2, x_3 \geq 0. \end{cases} \tag{1}$$



The efficient set is the union of the two shaded efficient faces E_1 and E_2. There are 5 efficient extreme points: $e_1 = (0, 0, 3)$, $e_2 = (10/7, 0, 16/7)$, $e_3 = (12/10, 32/10, 0)$, $e_4 = (0, 4, 0)$, $e_5 = (2, 0, 0)$. If X denotes the feasible region, The face marked E_1 can be characterized as the polyhedron that forces the first constraint in (1) to equality in the definition of X. It can also be defined as the convex hull of its four extreme points e_1, e_2, e_3, e_4. Finally, it is the optimal solution set to the optimization problem

$$\begin{cases} \max & \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 \\ \text{s.t.} & x \in X \end{cases}$$

for $(\lambda_1, \lambda_2, \lambda_3) = (2, 3, 4)$, and its positive multiples.

In large problems, the set of efficient extreme points may still contain too many points to be studied by the decision maker. Moreover, extreme efficient points may not carry the trade-off information well since some portions of the efficient set may end up being overemphasized whereas some regions are highly missed. Indeed, there is no reason for a decision maker to be solely interested in extreme point efficient solutions. The attractiveness of efficient extreme points mostly lies in their mathematical properties. With this motivation, a method that applies to a general set of (MOO) problems has been suggested to find globally-representative subsets of the efficient set [6].

Working in the Outcome Space

The outcome set $Y = \{ y \in \mathbf{R}^p \colon y = f(x) \; \exists \; x \in X \}$ helps redefine an equivalent problem to (MOO) in p-dimensional *outcome space*:

$$\text{(MOOO)} \quad \begin{cases} \max & y \\ \text{s.t.} & y \in Y . \end{cases}$$

As the number of objectives p is usually much less than the number of variables n, the structure of Y is simpler than that of X [4,8]. The ability to work directly with (MOOO) thus has the potential of providing significant computational benefits that vector optimization algorithms have tried to realize [3].

Reflections on Optimization Trends

As a field within the general field of optimization, multi-objective optimization is naturally affected by the trends that become dominant in optimization. Consequently, interior point methods, genetic algorithms, neural networks have been applied to the (MOO) problem in various ways. As there are difficult problems under (MOO) that cannot be yet practically solved, new developments in the general field of optimization constitute a potential to solve these problems.

Nonlinear and Integer Problems

Most of the algorithms proposed to solve problem (MOO) concentrate on the fully linear case. In general, when nonlinearities are introduced, the efficient solutions and the efficient set become difficult to characterize. There are some algorithms that allow for nonlinearities in the objective functions, and in the constraints that define the feasible region, but usually in a conservative way so as to retain some computational tractability. Similarly, the multiple objective integer programming problem is a very difficult one to solve due to the additional complications related to integrality.

Applications

Along with what one can call ‘case studies’, certain applications that are more generic than a case study but more specific than problem (MOO) itself have appeared. Typical examples include, but are not limited to, bicriteria network optimization problems, bicriteria knapsack problems, and multicriteria scheduling problems. Since usually these are problems that naturally involve multiple criteria, the methods developed for these problems have practical implications. Most of the methods developed can be categorized under a priori methods. A typical approach is to form a weighted combination of the objective functions. Recently, interactive and vector optimization approaches that deal with similar problems have also appeared.

A Related Optimization Problem

A related problem is the problem of optimizing a function g: $\mathbf{R}^n \rightarrow \mathbf{R}^p$ over the efficient set X_E. This can be a difficult *global optimization* problem depending on the properties of the objective function g. The problem is motivated in different ways. Sometimes, in certain settings, a function that is to serve as a pseudo utility function is available. Then optimizing this pseudo utility function over the efficient set in a sense corresponds to solving problem (MOO) itself. In addition, when g becomes one of the objective functions, then solving this problem provides the range of values the objective function takes over the efficient set. This information is valuable for a decision maker who is trying to make assessments to solve a problem and is used in some of the interactive algorithms. The difficulty of the problem has also resulted in heuristic solution approaches.

Trends

The advances in information technology affect the field of multiple criteria decision making heavily. Faster computers and parallel processing opportunities make it timewise feasible to solve optimization problems that would be deemed impractical in the past. Improved graphical capabilities make it feasible to accommodate sophisticated user interfaces to invite the decision maker in the problem solving process more actively and reliably. The developments in the World Wide Web present many opportunities to explore for individual and group decision support. At this point in time, there is still a need to solve the MOO problem in a rigorous, user-friendly and creative way. The decision support systems that enable the involvement of the decision maker in modeling and problem solving practically seem to be the way of solving (MOO) problems. The vector optimization approaches can also benefit from a decision support framework in their effort to help the decision maker identify a most-preferred solution.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Benayoun R, De Montgolfier J, Tergny J, Larichev O (1971) Linear programming with multiple objective functions: Step method (STEM). Math Program 1:366–375
2. Benson HP (1978) Existence of efficient solutions for vector maximization problems. J Optim Th Appl 26:569–580
3. Benson HP (1998) A hybrid approach for solving multiple objective linear programs in outcome space. J Optim Th Appl 98:17–35
4. Benson HP, Lee D (1996) Outcome-based algorithm for optimizing over the efficient set of a bicriteria linear programming problem. J Optim Th Appl 88(1):77–105
5. Benson HP, Lee D, McClure JP (1997) A multiple-objective linear programming model for the citrus rootstock selection problem in Florida. J Multi-Criteria Decision Anal 6:1–13
6. Benson HP, Sayin S (1997) Towards finding global representations of the efficient set in multiple objective mathematical programming. Naval Res Logist 44:47–67
7. Charnes A, Cooper WW (1977) Goal programming and multiple objective optimization-Part 1. Europ J Oper Res 1:39
8. Dauer JP, Liu Y-H (1990) Solving multiple objective linear programs in objective space. Europ J Oper Res 46:350–357
9. Ecker JG, Hegner NS, Kouada IA (1980) Generating all maximal efficient faces for multiple objective linear programs. J Optim Th Appl 30:353–381
10. Gardiner LR, Steuer RE (1994) Unified interactive multiple-objective programming - An open-architecture for accommodating new procedures. J Oper Res Soc 45(12):1456–1466
11. Geoffrion AM (1968) Proper efficiency and the theory of vector maximization. J Math Anal Appl 22:618–630

12. Geoffrion AM, Dyer JS, Feinberg A (1972) An interactive approach for multi-criterion optimization with an application to the operations of an academic department. Managem Sci 19:357–368
13. Hwang CL, Masud ASM (1979) Multiple objective decision making-methods and applications, A state of the art survey. Lecture Notes Economics and Math Systems. Springer, Berlin
14. Keeney RL, Raiffa H (1976) Decisions with multiple objectives: Preferences and value tradeoffs. Wiley, New York
15. Korhonen P, Laakso J (1986) A visual interactive method for solving the multiple criteria problem. Europ J Oper Res 24:277–287
16. Korhonen P, Moskowitz H, Wallenius J (1990) Choice behavior in interactive multiple criteria decision making. Ann Oper Res 23:161–179
17. Korhonen P, Wallenius J (1988) A Pareto race. Naval Res Logist 35:615–623
18. Sayin S (1996) An algorithm based on facial decomposition for finding the efficient set in multiple objective linear programming. Oper Res Lett 19:87–94
19. Steuer RE (1983) Operating manual for the ADBASE multiple objective linear programming package. Techn Report College Business Admin Univ Georgia, Athens
20. Steuer RE, Choo E-U (1983) An interactive weighted Tchebycheff procedure for multiple objective programming. Math Program 26:326–344
21. Steuer RE, Gardiner LR, Gray J (1996) A bibliographic survey of the activities and the international nature of multiple criteria decision making. J Multi-Criteria Decision Anal 5:195–217
22. Yu PL, Zeleny M (1975) The set of all nondominated solutions in linear cases and a multicriteria simplex method. J Math Anal Appl 49:430–468
23. Zeleny M (1973) Compromise programming. In: Cochrane JL, Zeleny M (eds) Multiple Criteria Decision Making. Univ South Carolina Press, Columbia

Multi-objective Optimization: Interaction of Design and Control

Carl A. Schweiger, Christodoulos A. Floudas
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90C29, 90C11, 90C90

Article Outline

Keywords

Interaction of design and control; Multi-objective optimization; Mixed integer nonlinear optimization; Pareto optimal solution

Traditionally, *process design* and *process control* are treated sequentially. Dynamics are not considered during the design phase, and flowsheet changes can not be made during the control phase. The problem with this approach is that the two are inherently connected as the design of the process affects its controllability. Thus, the steady state design and the dynamic operability issues should be treated simultaneously. Analyzing the *interaction of design and control* addresses the issue of quantitatively determining the *trade-offs* between the steady state economics and the dynamic controllability.

The interaction of design and control problem is to determine the process flowsheet which is both the economically optimal and controllable. There are different methods for addressing this problem. One common approach is to use overdesign where, once the economic steady state design is determined, surge tanks are added or equipment sizes are increased in order to handle any dynamic problems which may arise. This overdesign is usually based on heuristic rules and will likely move the design away from its economic optimum. There is no guarantee that the measures taken will even improve the controllability of the process. Other methods may examine the dynamic operation of several designs to determine which has the best controllability aspects.

There are very few methods which address the interaction of design and control in a quantitative manner.

The interaction of design and control can be addressed through a *process synthesis* approach involving optimization. This approach involves the representation of design alternatives through a process superstructure, the mathematical modeling of the superstructure, and the development of an algorithm to extract the optimal flowsheet from the superstructure. The simultaneous optimization of the design and control of the process is handled through multiple objectives representing the steady state economics and dynamic controllability. This naturally leads to a multi-objective framework.

Multi-objective Optimization

In any decision making process, the goal is to reach the best compromise solution among a number of competing objectives. Many examples of competing objectives exist in the field of engineering. For example, in the design of a process, one may have to consider safety and operational issues as well as economic issues. A decision making process is necessary when the most economic design is not the safest or most operable.

The best compromise solution depends on the relative importance of the conflicting objectives. This relative importance is not easily determined and is usually a subjective decision. The one responsible for making this decision is the decision maker (DM) whose choice can be based on a number of factors. Since subjective measures and decisions do not translate well into mathematics, a quantitative way of determining the tradeoffs and relative importance among the the objectives is necessary for a multi-objective optimization framework.

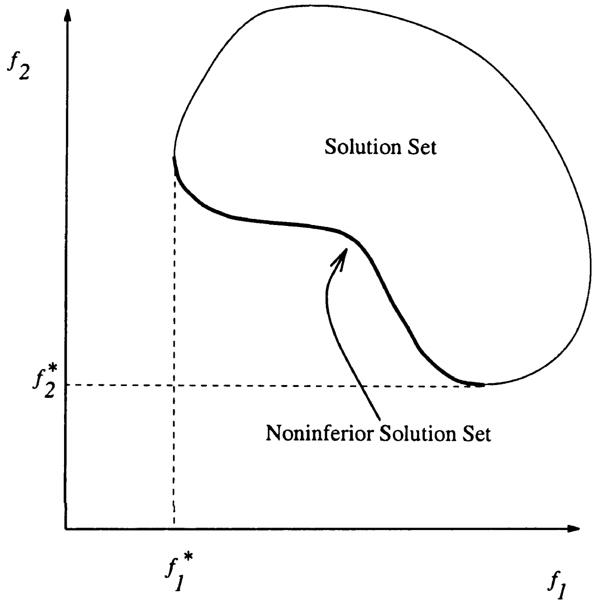
Multi-objective Framework for the Interaction of Design and Control

In analyzing the interaction of design and control, the objectives that are considered measure the steady state economics and the dynamic controllability of the process. The optimization approach in process synthesis serves as the basis for the multi-objective framework for the interaction of design and control. The procedure involves four steps:

1) Process representation;
2) Mathematical modeling;
3) Generation of noninferior solution set (determine trade-offs);
4) Best-compromise examination.

The first step is the representation of all the possible design alternatives through a process superstructure. In this step, all the units and possible connections of interest are incorporated into the superstructure such that all designs of interest are included as a subset of the superstructure.

Next, a mathematical model of the superstructure is developed for the superstructure as well as for for objective functions. The mathematical formulation is determined by the structure of the process flowsheet and must include all information needed to evaluate the objective functions. The objective functions must measure the economics of the process as well as the controllability of the process. Since the objective related to the economic performance is determined by steady state operation and the objective for the controllability is determined by its dynamic operation, the mathematical model most contain both steady state and dynamic information. The mathematical formulation involves both continuous and discrete variables where discrete variables are used to indicate the existence of units and connections within the flowsheet.



Multi-objective Optimization: Interaction of Design and Control, Figure 1
Noninferior solution set for a problem with two objectives

Once the model has been formulated, an algorithm is developed and used to determine the quantitative trade-offs among the competing objectives. Individually, each objective can be optimized, but together, they will be in conflict. This means that there is a set of solutions where one objective can be improved only at the expense of the other objectives. This set of solutions is called the *noninferior solution set* which is visually depicted for a two objective problem in Fig. 1. This solution set is also referred to as *nondominated* and *Pareto optimal* and the surface of noninferior solutions implicitly defines a function $G(\mathbf{J})$.

Using the information about the trade-offs among the competing objectives, a strategy for determining the best compromise solution is developed. This strategy is based on information from the DM and depends on the relative weights given to the objectives. These weights are varied systematically to locate the solution which the DM prefers the most. How to determine these weights is one of the more interesting aspects of the problem.

Note that the multi-objective problem can be reduced if some of the objectives (presumably those with very low weights) need not be optimized but simply brought to a satisfactory level. In this case, these objectives can be incorporated into the problem as constraints.

General Mathematical Formulation

The mathematical model is a multi-objective *mixed integer nonlinear programming* problem which has the following form:

$$\begin{cases} \text{OPTIMIZE} & \mathbf{J}(\mathbf{x},\mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q . \end{cases} \tag{1}$$

In this formulation, $\mathbf{J}$ is a vector of objectives which includes the economic and controllability objectives. The expressions $\mathbf{h}$ and $\mathbf{g}$ represent material and energy balances, thermodynamic relations, and other constraints. The controllability measures are included in the formulation as $\boldsymbol{\eta}$. The variables in this problem are partitioned as continuous $\mathbf{x}$ and binary $\mathbf{y}$.

Solution of the MOP

One way to address the solution of the MOP is to formulate it using a utility function U which implicitly relates the multiple objectives in terms of some common basis:

$$\begin{cases} \min & U[\mathbf{J}(\mathbf{x},\mathbf{y})] \\ \text{s.t.} & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \le \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q . \end{cases} \tag{2}$$

By introducing the utility function, the vector optimization problem has been reduced to a scalar optimization problem and MINLP techniques can be applied to solve the problem. These MINLP techniques include *generalized Benders decomposition* (GBD) [4,14], *outer approximation* (OA) [2], *outer approximation with equality relaxation* (OA/ER) [8], and *outer approximation with equality relaxation and augmented penalty* (OA/ER/AP) [16]. These methods are discussed in detail in [3].

With the definition of the noninferior solution set, the optimization problem can be formulated as

$$\begin{cases} \min & U[\mathbf{J}(\mathbf{x},\mathbf{y})] \\ \text{s.t.} & G(\mathbf{J}) = 0 . \end{cases} \tag{3}$$

The challenging aspect of the problem is determining the explicit form of the utility function. One possible form of the utility function is a weighted linear sum of the objectives:

$$U[\mathbf{J}(\mathbf{x},\mathbf{y})] = \sum_{i\in I} w_i J_i ,$$

where I is the set of objective functions and w_i are the weights for the objective functions whose value is determined by the DM. The difficulty that arises is that the utility function is generally not known. It is, however, assumed to be convex and continuously differentiable.

The issues surrounding the solution of the multi-objective optimization problem are determining the noninferior solution set, determining the utility function based on information from the DM, and determining the best-compromise solution.

Different techniques have been developed in order to assess the trade-offs among the objectives quantitatively. See [7] for a tutorial in multi-objective optimization. A review is also available in [17]. Much of the fundamental aspects of multi-objective optimization can be found in [1].

Noninferior Solution Sets

The noninferior solution set can be determined in a number of ways. One approach is the formulate the problem as

$$\begin{cases} \min & \sum_{i \in I} w_i J_i(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q , \end{cases} \tag{4}$$

where the weights w_i are selected such that $w_i \geq 0$ for all i and $\sum_{i \in I} w_i = 1$. Through a suitable choice of the weights, the noninferior solution set can be found. This approach can miss some points in the noninferior solution set if the solution region is nonconvex. In order to address this problem, a weighted norm can be used as follows:

$$\begin{cases} \min & \left\{ \sum_{i \in I} \left[w_i J_i(\mathbf{x}, \mathbf{y}) \right]^p \right\}^{1/p} \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q . \end{cases} \tag{5}$$

By increasing the size of p, the curvature of the supporting function is increased and more noninferior points can be found. In the extreme of $p = \infty$, all the noninferior points can be located. Using the ∞-norm, the problem becomes

$$\begin{cases} \min & \max_{i \in I} w_i J_i(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q . \end{cases} \tag{6}$$

The advantage of this formulation is that the weights have a physical meaning for the DM. If the DM knows the desired values for each objective for a given noninferior point, the weights can be set to the reciprocal of these values. The noninferior solution will be the one that is most like the one with the values specified by the DM. The disadvantage of this formulation is that it can be difficult to solve.

Another way to determine the noninferior solution set is through the *ϵ-constraint method* [6]. In this approach, all but one of the objectives is incorporated into the problem as a constraint less than ϵ. This results in the following formulation:

$$\begin{cases} \min & J_1(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} & J_i(\mathbf{x}, \mathbf{y}) \leq \epsilon_i , \quad i = 2, \dots, q , \\ & \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0,1\}^q . \end{cases} \tag{7}$$

By varying the values of ϵ_i, the points of the noninferior solution set can be found.

Choosing the Best-Compromise Solution

To this point, the focus has been on determining the noninferior solution set. Only one of the points can be chosen as the best solution for the problem, and the task of the DM is to determine this point. Once the noninferior solution set is determined, it is presented to the DM who will choose the solution point he prefers. The selection of this point is based on the relative importance of the objectives in the eyes of the decision maker.

Instead of assigning arbitrary weights to the various objectives, a systematic approach can applied which uses the trade-off information in the noninferior solution set. The slope of the noninferior solution set at any point reveals how much one objective will be improved at the expense of another objective. This information is used in an interactive, iterative *cutting plane algorithm* to determine the best compromise solution.

Cutting Plane Algorithm

The cutting plane algorithm described in [11] is based on [5] and [10]. Marginal rates of substitution were used to solve problems of the form (2) where U is unknown, convex, and continuously differentiable. Due to convexity, the partial derivatives of U with respect to each of the arguments in the objective space are positive. This is expressed mathematically as

$$\frac{\partial U(\mathbf{J})}{\partial J_i} > 0.$$

Thus, a decrease in J_i will lead to a decrease in U. In the interactive scheme, the DM is asked for the positive trade-off weights, w_i^k, for a given solution k. This weight is defined as the ratio of the change in the utility function with respect to one function divided by the change in the utility function with respect to another. This is expressed mathematically as

$$w_i^k = \frac{\partial U(\mathbf{J}^k) \,/\, \partial J_i}{\partial U(\mathbf{J}^k) \,/\, \partial J_1}$$

where $\mathbf{J}^k = [J_1(\mathbf{x}^k, \mathbf{y}^k), \ldots, J_1(\mathbf{x}^k, \mathbf{y}^k)]$. A line search along a feasible direction of steepest descent locates an improved solution for the next iteration.

By exploiting the fact that the utility function is convex, cutting planes can be introduced to reduce the search to improving directions [10]. Since U is convex,

$$\begin{aligned} 0 &\geq U(\mathbf{J}^*) - U(\mathbf{J}^k) \\ &\geq \nabla_f U(\mathbf{J}^k)(J^* - \mathbf{J}^k) \\ &\geq \begin{cases} \min & \nabla_f U(\mathbf{J}^k)(\mathbf{J} - \mathbf{J}^k) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0, 1\}^q . \end{cases} \end{aligned} \tag{8}$$

This involves the linearization in the objective space around the point J^k. If the solution to the minimization is zero, then the optimal solution $J^\star$ has been found. If the solution has a negative value, then the direction leads to an improvement in the objective space. This minimization can be performed over a number of points $k = 1, \ldots, K$ to find a direction which improves all of them. Cutting planes in the objective space are formed to find new values of the objectives which improve the utility function according to the trade-off weights, ∇U, which the DM provides. At each iteration of the algorithm, the following problem must be solved:

$$\begin{cases} \min & z \\ \text{s.t.} & z \geq \sum_{i=1}^{p} w_i^k (J_i(\mathbf{x}, \mathbf{y}) - J_i(\mathbf{x}^k, \mathbf{y}^k), \\ & \quad \forall k = 1, \ldots, K. \\ & \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{O} \\ & \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{O} \\ & \mathbf{x} \in \mathbb{R}^p \\ & \mathbf{y} \in \{0, 1\}^q . \end{cases} \tag{9}$$

The steps of the cutting plane algorithm are the following:

```
1  Determine the initial solution point $k = 1$ and
   determine the values of all the objective functions.
   Assign the values of the weights $w_i^k$.
2  Solve (9) to find new values of $\mathbf{x}$ and $\mathbf{y}$.
   Determine the values of the objective functions
   for the new values of $\mathbf{x}$ and $\mathbf{y}$.
3  IF the solution to (9) is zero, THEN go to Step 4
   ELSE set $k = k + 1$, update the values $\mathbf{x}^k$, $\mathbf{y}^k$,
   and $\mathbf{J}^k$, generate new weights, and go to Step 2.
4  Terminate with $\mathbf{x}^k$ and $\mathbf{y}^k$ as the best-compromise solution.
```

Cutting plane algorithm

This algorithm requires the DM to provide only trade-off weights at each iteration. These weights can be estimated by knowledge of the relative importance of the objectives or by information from the noninferior solution set.

Multi-objective Optimization in the Interaction of Design and Control

The interaction of design and control has been recognized as a multi-objective problem by many researchers as the objectives representing the steady-state economic design and dynamic controllability are regarded as noncommensurable. One of the first challenges in this problem is determining a suitable controllability objective. The choice of the controllability objective will dictate the required elements of the mathematical formulation of the problem.

One of the early works which addressed the multi-objective nature of the interaction of design and control was that of [9]. A given set of alternative steady-state designs was assumed to be known. Bounds on the dynamic measures of the designs were determined and used to screen designs and determine the noninferior solution set. No method was provided for determining the best-compromise solution.

In the work of [13], singular value decomposition is used to determine dynamic operability measures. The controllability is formulated through the linearization of the model and is given in terms of the singular values of the transfer function. This modeling leads to an infinite-dimensional problem as all frequencies must be considered for the controllability measure. For the multi-objective optimization, the ϵ-constraint method was used to determine the noninferior solution set. The scalar optimization was addressed by approximating the infinite-dimensional problem and using an gradient-based algorithm to solve the optimization problem and determine the operating parameters for the process.

The previous methods did not take into account that the structure of the process flowsheet as well as the design parameters determine its inherent controllability. In order to consider structural alternatives in the process flowsheet such as the existence of units in the flowsheet, discrete variables are used in the process modeling. This aspect of the process design was considered by [11,12] in the interaction of design and control by using the optimization approach to process synthesis. In this approach, the structure of the process flowsheet and the design parameters are considered simultaneously with the dynamic controllability of the process. The controllability measures employed were the open-loop linear controllability measures (singular value, condition number, relative gain array). The noninferior solution set was determined using the ϵ-constraint method, and the best-compromise solution was found using the cutting plane method described above.

Further development of the above technique was addressed by [15] where nonlinear dynamic models were considered. The problem was formulated as a multi-objective *mixed integer optimal control problem*. The multi-objective problem was again solved using the ϵ-constraint method. The mixed integer optimal control problem was solved by extending the methods for solving mixed integer nonlinear optimization to handle dynamic systems.

Conclusions

Analyzing the interaction of design and control leads to a multi-objective optimization problem. The key issue in solving this problem is quantitatively determining the trade-offs between the steady-state economics and the dynamic controllability. By using multi-objective optimization techniques, these characteristics of the process can be traded off in a systematic manner.

By following the optimization approach to process synthesis, a mathematical framework can be developed. This involves developing a superstructure of design alternatives and effective mathematical models for the different criteria. The algorithmic procedure for solving the multi-objective problem involves the successive solution of scalar optimization problems to determine the noninferior solution set. The final step in the approach is to determine the best-compromise solution from those in the noninferior solution set.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Control Vector Iteration
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Duality in Optimal Control with First Order Differential Equations
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming

- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Optimal Control of a Flexible Arm
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control

References

1. Clark PA, Westerberg AW (1983) Optimization for design problems having more than one objective. Comput Chem Eng 7(4):259–278
2. Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math Program 36:307–339
3. Floudas CA (1995) Nonlinear and mixed integer optimization: Fundamentals and applications. Oxford Univ Press, Oxford
4. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10(4):237–260
5. Geoffrion AM, Dyer JS, Feinberg A (1972) An interactive approach for multi-criterion optimization with an application to the operation of an academic department. Managem Sci 19:357–368
6. Haimes Y, Hall WA, Freedman HT (1975) Multi-objective optimization in water resource systems: The surrogate worth trade-off method. Elsevier, Amsterdam
7. Hwang CL, Paidy SR, Yoon K, Masud ASM (1980) Mathematical programming with multiple objectives: A tutorial. Comput Oper Res 7:5–31
8. Kocis GR, Grossmann IE (1987) Relaxation strategy for the structural optimization of process flow sheets. Industr Eng Chem Res 26(9):1869
9. Lenhoff AM, Morari M (1982) Design of resilient processing plants I: Process design under consideration of dynamic effects. Chem Eng Sci 37(2):245–258
10. Loganathan GV, Sherali HD (1987) A convergent interactive cutting-plane algorithm for multiobjective optimization. Oper Res 35:365–377
11. Luyben ML, Floudas CA (1994) Analyzing the interaction of design and control-1. A multiobjective framework and application to binary distillation synthesis. Comput Chem Eng 18(10):933–969
12. Luyben ML, Floudas CA (1994) Analyzing the interaction of design and control-2. Reactor-separator-recycle system. Comput Chem Eng 18(10):971–994
13. Palazoglu A, Arkun Y (1986) A multiobjective approach to design chemical plants with robust dynamic operability characteristics. Comput Chem Eng 10(6):567–575
14. Paules IV GE, Floudas CA (1989) APROS: Algorithmic development methodology for discrete-continuous optimization problems. Oper Res 37(6):902–915
15. Schweiger CA, Floudas CA (1997) Interaction of design and control: Optimization with dynamic models. In: Hager WW, Pardalos PM (eds) Optimal Control: Theory, Algorithms, and Applications. Kluwer, Dordrecht, pp 388–435
16. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer approximation method for MINLP optimization. Comput Chem Eng 14(7):769–782
17. Zionts S (1979) Methods for solving management problems involving multiple objectives. Working Paper SUNY at Buffalo

Multi-objective Optimization: Interactive Methods for Preference Value Functions

HAROLD P. BENSON
Department Decision and Information Sci.,
University Florida, Gainesville, USA

MSC2000: 90C29

Article Outline

Keywords

Multi-objective optimization; Multiple criteria decision making; Interactive method; Preference value function; Value function

The *multi-objective optimization* (*multiple criteria decision making*) problem is the problem of choosing a most preferred solution when two or more incommensurate, conflicting objective functions (criteria) are to be simultaneously maximized. Interest in multi-objective optimization has risen sharply during the past 30 years. There are at least three reasons for this. First, and most importantly, is the increasing recognition that most applied problems in both the private and public sectors involve multiple objectives rather than one objective. Second, a variety of solution algorithms for multi-objective optimization are now available. Finally, the enormous improvements in the speed and storage of computers make it practical to apply these algorithms to the solution of realistically-sized problem applications.

Formally, the statement of the multi-objective optimization problem of interest here is

$$\text{(V)} \qquad \begin{cases} \text{VMAX} & f(x) = [f_1(x), \dots, f_p(x)], \\ \text{s.t.} & x \in X. \end{cases}$$

Here, $p \geq 2$, X is a nonempty subset of $\mathbf{R}^n$, each f_j, $j = 1, \dots, p$, is a real-valued function defined on X or on some suitable set containing X, and VMAX indicates that, in some unspecified sense, we are to 'vector maximize' the vector $f(x)$ of *objective functions* (*criteria*) over X. The set X is called the set of *decision alternatives* or the *decision set*, and $\{f(x) \in \mathbf{R}^p \colon x \in X\}$, is called the *outcome set*.

There are a large number of diverse solution algorithms for problem (V). All are intended to help the *decision maker* (DM) find a most preferred solution to the problem. In the majority of these algorithms, the notion of efficiency plays an indispensable role. An *efficient* (*nondominated*, *noninferior*, *Pareto optimal*) solution for problem (V) is a solution $\overline{x} \in X$ such that there exists no other solution $x \in X$ that satisfies $f(x) \geq f(\overline{x})$ and $f(x) \neq f(\overline{x})$. Let X_E denote the set of efficient solutions for problem (V). Notice that if $\overline{x} \in X_E$, then there is no other feasible solution for problem (V) that achieves at least as large a value as $\overline{x}$ in each criterion of the problem and a strictly larger value than $\overline{x}$ in at least one criterion of the problem.

In the great majority of instances of problem (V), the *preference value function* (*value function*) v of the DM is unknown. This is a function $v \colon \mathbf{R}^p \to \mathbf{R}$ that maps the outcomes of problem **V** to real numbers in such a way that for any two outcomes y^1 and y^2, the DM prefers y^1 to y^2 if and only if $v(y^1) > v(y^2)$. Although v is unknown, what *is* known is that for each objective function f_j, the DM prefers more of f_j to less of f_j. Mathematically, this means that v is *coordinatewise increasing*, i. e., that whenever $\overline{z}, z \in \mathbb{R}^p$ satisfy $\overline{z} \geq z$ and $\overline{z}_j > z_j$ for some $j = 1, \dots, p$, then $v(\overline{z}) > v(z)$. It is easy to show that when v is coordinatewise increasing, any maximizer x^* of $v\,[f(x)]$ over X must satisfy $x^* \in X_E$. In other words, as long as the DM prefers more to less, the search for a most preferred solution to problem (V) can be confined to X_E. This is one of the key reasons that the concept of efficiency is so important to the majority of the algorithms for problem (V).

The interactive methods constitute one of the most popular categories of algorithms for solving problem (V). An *interactive method* for problem (V) consists of a sequence of DM-computer interactions designed to create a sequence of decision alternatives that terminates with a most preferred solution to the problem. In a majority of cases, the generated alternatives are efficient. Each iteration of the interactive process consists of three steps. First, an initial solution is found with the aid of the computer. Typically, this solution is found by solving a single-objective optimization problem that generates either an efficient point or, at worst, a feasible point. Next, the DM is asked to react to the generated point by answering one or more questions involving his preferences for it. Last, based upon the answers given, the computer generates a new point, typically by modifying parameters in the single-objective optimization problem. This process continues until either the computer or the DM identifies a most preferred solution. The value function v of the DM is never needed and, in fact, is assumed to be unavailable.

There are several advantages to using interactive methods as compared to other categories of methods for problem (V). For instance, the preference information asked of the DM at each iteration is not difficult to supply. Furthermore, the DM thereby learns about his value function, which is often initially vague or mostly unknown. As the search continues, the DM also learns about the decision or efficient decision alternatives available and the trade-offs in the objective functions across these decision alternatives. The optimizations required of the computer are also usually not difficult to perform. Finally, because the DM is highly involved in the process, his confidence in the most preferred solution that is eventually found is enhanced.

A frequent criticism of the interactive methods is that, in practice, the work required of the DM during the iterations seems to be burdensome for him in many cases. This may cause the DM to prematurely terminate the search so that a most preferred solution is not found.

There are literally hundreds of interactive algorithms for problem (V). Many are limited to cases where problem (V) is a multiple objective linear programming problem. Others apply when problem (V) is a multiple objective convex, nonlinear programming problem, a multiple objective integer programming problem, or some other type of multiple objective optimization problem. Instead of examining these algorithms individually, we will describe them by groups according to the characteristics that they possess.

One of the key characteristics of the interactive algorithms concerns the type of information required of the DM at each iteration. For instance, at each iteration, the DM may be asked to intuitively assign or reassign *weights* to the criteria according to his current assessment of their relative importance. R.E. Steuer [13] has shown some important stumbling blocks to this approach, however. Other algorithms may instead elicit *relaxation quantities* from the DM. In these cases, the DM is asked how much he would be willing to relax the level of one objective function in order to obtain possible improvements in the levels of other objective functions. Some of the oldest interactive algorithms use this approach [1,9]. Still other types of algorithms ask the DM various types of *trade-off questions*. The trade-off questions are designed to obtain an estimate of the gradient of the value function of the DM at the current solution. This approach is also relatively old, but difficult for the DM to accomplish [5,14]. Finally, a number of algorithms call for the DM to make *paired comparisons* at each iteration. In a paired comparison, the DM is given two solutions to compare and must give his preference for one or the other. Usually, the DM can accomplish this. But when the two solutions are quite similar, difficulties can arise [15]. In addition, algorithms that use paired comparisons can sometimes call for excessive numbers of these comparisons [12].

A second dimension where the interactive algorithms differ is in the approach used to explore the feasible region X or the efficient set X_E. Some algorithms use *feasible direction methods* [2]. In these algorithms, at each iteration, the direction to move from a point that was last found and the distance to move along the direction are determined with the aid of the DM. By moving along the direction by the specified amount, the next solution point is found. In many algorithms, all such points are efficient. In another group of algorithms, *feasible region reduction* is used to explore X or X_E. As points in X or in X_E are examined in these methods, portions of X are removed, usually via linear cuts. Another set of algorithms uses *weighting space reduction*. In these algorithms, a weighted sum of f_j, $j = 1, \dots, p$, is maximized at each iteration, thereby yielding a point in X_E. Based upon the DM's responses to these maximizations, portions of the weighting space are removed. Eventually, the portion of the weighting space remaining is so small that the DM can pick out the set of weights associated with a most preferred solution.

Other approaches used to explore X or X_E include the *trade-off cutting plane* method [10], *Lagrange multiplier* methods, *visual interactive* methods (see, e. g. [7]), and the *branch and bound* method [8], among others. For further reading concerning these methods, see [3,4,6,11,12,13].

Another way to group the interactive algorithms for problem (V) is according to whether or not they handle inconsistencies in the DM's preference responses. As human beings, DM's are prone to giving preference responses over the course of the solution procedure that imply inconsistencies such as asymmetries or intransitivities of preference. Some algorithms take no account of these possible inconsistencies and have been criticized for this [12]. Others attempt to reduce inconsistency by either minimizing the DM's cognitive burden or by incorporating tests for inconsistency that are used as the interactive solution process proceeds.

W.S. Shin and A. Ravindran [12] have compared various of the classes of interactive algorithms according to four criteria that are important in practice. These criteria are the DM's cognitive burden, the ease with which the single-objective optimizations called for can be used, implemented and solved, the handling of inconsistency, and the overall quality of the solution process and the answers obtained. Although preliminary, these comparisons seem to show the relative superiority of the weighting space reduction and other criterion weight space search methods, and of the visual interactive methods. Readers should note, however, that the rankings in the study are subjectively-obtained by the authors [7].

For further general reading on interactive methods, see [2,3,4,6,11,12,13,14].

See also

- ▶ Bi-objective Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Benayoun R, Montgolfier J, Tergny J, Laritchev O (1971) Linear programming with multiple objective functions: Step method (STEM). Math Program 1:366–375
2. Benson HP, Aksoy Y (1991) Using efficient feasible directions in interactive multiple objective linear programming. Oper Res Lett 10:203–209
3. Buchanan JT, Daellenbach HG (1987) A comparative evaluation of interactive solution methods for multiple objective decision models. Europ J Oper Res 29:353–359
4. Evans GW (1984) An overview of techniques for solving multiobjective mathematical programs. Managem Sci 30:1268–1282
5. Geoffrion AM, Dyer JS, Feinberg A (1972) An interactive approach for multicriterion optimization, with an application to the operation of an academic department. Managem Sci 19:357–368
6. Goicoechea A, Hansen DR, Duckstein L (1982) Multiobjective decision analysis with engineering and business applications. Wiley, New York
7. Kohornen P, Laakso J (1986) A visual interactive method for solving the multiple criteria problem. Europ J Oper Res 24:277–287
8. Marcotte O, Soland R (1985) An interactive branch-and-bound algorithm for multiple criteria optimization. Managem Sci 32:61–75
9. Monarchi DE, Kisiel CC, Duckstein L (1973) Interactive multiobjective programming in water resources: A case study. Water Resources Res 9:837–850
10. Musselman K, Talavage J (1980) A tradeoff cut approach to multiple objective optimization. Oper Res 28:1424–1435
11. Rosenthal RE (1985) Concepts, theory and techniques: Principles of multiobjective optimization. Decision Sci 16:133–152
12. Shin WS, Ravindran A (1991) Interactive multiple objective optimization. Survey I: Continuous case. Comput Oper Res 18:97–114
13. Steuer RE (1986) Multiple criteria optimization: Theory, computation, and application. Wiley, New York
14. Wallenius J (1975) Comparative evaluation of some interactive approaches to multicriterion optimization. Managem Sci 21:1387–1396
15. Zionts S, Wallenius J (1983) An interactive multiple objective linear programming method for a class of underlying nonlinear utility functions. Managem Sci 29:519–529

Multi-objective Optimization: Lagrange Duality

HIROTAKA NAKAYAMA
Department Applied Math., Konan University, Kobe, Japan

MSC2000: 90C29, 90C30

Article Outline

Keywords

Vector inequality; Efficient point; Vector valued Lagrangian

As is well known, duality in mathematical programming is based on the property that any closed convex set can be also represented by the intersection of closed half spaces including it. Let the multi-objective optimization problem to be considered here be given by

$$(\mathrm{P}) \quad \begin{cases} \min & f(x) := (f_1(x), \dots, f_p(x)) \\ \text{over} & x \in X, \end{cases}$$

where

$$X = \left\{ x \in X' \colon \begin{array}{l} g_i(x) \leqq 0, \\ i = 1, \dots, m, X' \subset \mathbb{R}^n \end{array} \right\}.$$

Note here that vector inequalities are commonly used: for any n-vectors a and b, $a > b$ means $a_i > b_i$ ($i = 1, \dots, n$). Also, $a \geqq b$ means $a_i \geqq b_i$ ($i = 1, \dots, n$). On the other hand, $a \geq b$ means $a \geqq b$ but $a \neq b$. Hereafter, vector inequalities such as $g(x) \leqq 0$ will be used instead of $g_i(x) \leqq 0$ ($i = 1, \dots, m$).

Defining a dual problem (D) in some appropriate way associated with the problem (P), our aim is to show the property min(P) = max(D). Here min(P) denotes the set of efficient points of the problem (P) in the objective function space $\mathbf{R}^p$, and similarly max(D) the one of the dual problem (D).

Unlike the usual mathematical programming, the optimal value of the primal problem (and the dual problem) are not necessarily determined uniquely in multi-objective optimization. Hence, there have been developed several kinds of formulation of dual problem in order to get the desirable property min(P) = max(D). Regarding Lagrange duality, three typical dualizations can be seen in linear cases, nonlinear cases and geometric approaches [6].

Linear Cases

The first result on duality for multi-objective optimization seems the one given in [1] for linear cases. This is formulated as a matrix optimization including the vector optimization as a special case. Although there have been several related works, the probably most attractive one is given in [2] because it is formulated as a natural extension of traditional linear programming: Let A be an $m \times n$ matrix, C a $p \times n$ matrix, and b an m-vector. Then the *primal problem* (P) in linear cases is formulated as

$$(\mathrm{P_I}) \quad \begin{cases} \min & Cx \\ \text{s.t.} & Ax \geqq b \\ & x \geqq 0. \end{cases}$$

Associated with ($\mathrm{P_I}$), H. Iserman [2] defined the *dual problem* as

$$(\mathrm{D_I}) \quad \begin{cases} \max & \Lambda b \\ \text{s.t.} & \Lambda A \ngeqq C \\ & \Lambda \geqq 0. \end{cases}$$

Here, the multiplier $\Lambda \geqq 0$ is a $p \times m$ matrix whose elements are all nonnegative.

Then Isermann's duality is given by

i) $\Lambda\, b \ngeq Cx$ for all feasible x and Λ.

ii) Suppose that $\overline{\Lambda} b = C\overline{x}$ for some feasible $\overline{x}$ and some feasible $\overline{\Lambda}$. Then $\overline{\Lambda}$ is an efficient solution to ($\mathrm{D_I}$) and $\overline{x}$ is an efficient solution to ($\mathrm{P_I}$).

iii) min($\mathrm{P_I}$) = max($\mathrm{D_I}$).

Nonlinear Cases

The most natural dualization in nonlinear multi-objective optimization seems to be the one given in [10].

Consider the problem (P), and assume the following:

i) X' is a nonempty compact convex set.

ii) f is continuous, and $f(X) + \mathbf{R}^n_+$ is convex in $\mathbf{R}^p$.

iii) $g_i (i = 1, \dots, m)$ are continuous and convex.

Under these assumptions, it can be readily shown that for every $u \in \mathbf{R}^m$, both sets $X(u) = \{x \in X' \colon g(x) \leqq u\}$ and $Y(u) = f[X(u)] = \{y \in \mathbb{R}^p \colon y = f(x), x \in X', g(x) \leqq u\}$ are compact and convex.

The primal problem (P) can be embedded as ($\mathrm{P_0}$) in a family of perturbed problems (P_u) given by

$$(\mathrm{P_u}) \quad \min Y(u).$$

Defining $\Gamma = \{u \in \mathbf{R}^m \colon X(u) \neq \emptyset\}$, the set Γ is convex. Now in a similar fashion to the ordinary mathematical programming, the perturbed map can be defined by

$$W(u) = \min \left\{ f(x) \colon\ x \in X',\ g(x) \leqq u \right\}.$$

It is known that for every $u \in \Gamma$, $W(u) + \mathbf{R}^p_+$ is convex and

$$W(u) + \mathbb{R}^p_+ = Y(u) + \mathbb{R}^p_+ .$$

In addition, the map W is monotone and convex on Γ.

Now, define the vector valued Lagrangian function with a $p \times m$ matrix multiplier Λ as

$$L(x, \Lambda) = f(x) + \Lambda g(x).$$

Associated with this definition, the dual map can be defined as

$$\Phi(\Lambda) = \min \Omega(\Lambda),$$

where

$$\Omega(\Lambda) = \left\{L(x, \Lambda)\colon\ x \in X'\right\}.$$

Under the terminology, the dual problem associated with the primal problem (P) can be given by

$$(\mathrm{D_{TS}}) \quad \max \bigcup_{\Lambda \in \mathcal{L}} \Phi(\Lambda).$$

It can be shown that Φ is concave point-to-set map on Γ, namely

$$\begin{aligned}&\Phi(\alpha\Lambda^1 + (1-\alpha)\Lambda^2)\\ &\subset \alpha\Phi(\Lambda^1) + (1-\alpha)\Phi(\Lambda^2) + \mathbb{R}^p_+\end{aligned}$$

and $\Phi(\Lambda) + \mathbf{R}^p_+$ is a convex set in $\mathbf{R}^p$ for each $\Lambda \in \mathcal{L}$. Here $\mathcal{L}$ is the set of all $p \times m$ matrices whose components are all positive.

T. Tanino and Y. Sawaragi [10] presented the following as duality in multi-objective optimization:

Theorem 1

i) *For any $x \in X$ and $y \in \Phi(\Lambda)$*

$$y \not\geq f(x).$$

ii) *Suppose that $\widehat{x} \in X$, $\widehat{\Lambda} \in \mathcal{L}$ and $f(\widehat{x}) \in \Phi(\widehat{\Lambda})$. Then $\widehat{y} = f(\widehat{x})$ is an efficient point to the primal problem (P) and also to the dual problem ($\mathrm{D_{TS}}$).*

iii) *Suppose that any efficient solutions to (P) are all proper and that Slater's constraint qualification is satisfied. Then*

$$\min (P) \subset \max (D_{TS}).$$

Remark 2 The above theorem is not complete in the sense that the relation min(P) = max(D) does not hold. Regarding conjugate duality, there have been reports presenting w-min(P) = w-max(D) (see, e. g., [4] and [9]). Several studies based on geometric consideration have been made for deriving the relation min(P) = max(D) using vector valued Lagrangian. This will be stated in the following

Geometric Duality

Geometric considerations are made in [3], based on the supporting hyperplanes for epi$W$, and in [5], based on the supporting conical varieties for epi$W$, which is denoted by G here.

Define

$$G = \left\{(u, y) \in \mathbb{R}^p \times \mathbb{R}^p\colon \begin{array}{l} y \geqq f(x),\\ u \geqq g(x)\\ \text{for some } x \in X' \end{array}\right\},$$

$$Y_G = \left\{y\colon\ (0, y) \in G, 0 \in \mathbb{R}^m,\ y \in \mathbb{R}^p\right\}.$$

Associates with the primal problem (P), we consider the following two kinds of dual problems:

$$(\mathrm{D_N}) \quad \max \bigcup_{\Lambda \in \mathcal{L}} Y_{S(\Lambda)},$$

where

$$Y_{S(\Lambda)} = \left\{y \in \mathbb{R}^p\colon\ f(x) + \Lambda g(x) \nleq y,\ \forall x \in X'\right\}$$

and

$$(\mathrm{D_J}) \quad \bigcup_{\substack{\mu > 0\\ \lambda \geqq 0}} Y_{H^-(\lambda,\mu)},$$

where

$$Y_{H^-(\lambda,\mu)} = \left\{y \in \mathbb{R}^p\colon \begin{array}{l} \langle \mu, f(x)\rangle + \langle \lambda, g(x)\rangle \geqq \langle \mu, y\rangle\\ \forall x \in X' \end{array}\right\}.$$

Theorem 3

i) *For any feasible x in (P) and for any feasible y in ($\mathrm{D_N}$) or ($\mathrm{D_J}$),*

$$y \not\geq f(x).$$

ii) *Assume that G is closed, that there exists at least an efficient solution to the primal problem, and that* *these solutions are all proper. Then, under the condition of Slater's constraint qualification, the following holds:*

$$\min (P) = \max (D_N) = \max (D_J).$$

Remark 4 In the above duality, we assumed that the convex set G is closed and that Slater's constraint qualification is satisfied, which seem relatively restrictive. Instead of these conditions, J. Jahn [3] assumed that Y_G is closed and some normality condition.

Define

$$A_{G(\mu)} = \left\{\alpha\colon\ (0,\alpha) \in G(\mu),\ 0 \in \mathbb{R}^m,\ \alpha \in \mathbb{R}^1\right\}$$
$$Y_G = \{y\colon\ (0,y) \in G,\ 0 \in \mathbb{R}^m,\ y \in \mathbb{R}^m\}.$$

Definition 5 The primal problem (P) is said to be *J-normal*, if for every $\mu > 0$

$$\mathrm{cl}(A_{G(\mu)}) = A_{\mathrm{cl}\,G(\mu)}.$$

The primal problem (P) is said to be *J-stable*, if it is J-normal and for an arbitrary $\mu > 0$ the problem

$$\sup_{\lambda \geqq 0} \inf_{x\in X} \langle \mu, f(x)\rangle + \langle \lambda, g(x)\rangle$$

has at least one solution.

On the other hand, J.W. Nieuwenhuis [7] suggested another normality condition:

Definition 6 The primal problem (P) is said to be *N-normal*, if

$$\mathrm{cl}\, Y_G = Y_{\mathrm{cl}\,G}.$$

Lemma 7 *Slater's constraint qualification ($\exists \widehat{x}$, $g(\widehat{x}) > 0$) yields J-stability and N-normality.*

Theorem 8 *Suppose that Y_G is closed, $\min_D(P) \neq \emptyset$, and the efficient solutions to (P) are all proper. Then, under the condition of J-stability,*

$$\min (P) = \max (D_N) = \max (D_J).$$

Duality for Weak Efficiency

Define

$$Y_{S'(\Lambda)} = \left\{y \in \mathbb{R}^p\colon\ f(x) + \Lambda g(x) \nless y,\ \forall x \in X'\right\}.$$

Theorem 9 *Suppose that Y_G is a nonempty subset in $\mathbf{R}^p$ and $Y_G + \mathbf{R}^p_+$ is bounded. Then under the condition of N-normality*

$$\begin{aligned} w\text{-}\min \mathrm{cl}\, Y_G &= w\text{-}\max \mathrm{cl} \bigcup_{\Lambda\in\mathcal{L}} Y_{S'(\Lambda)} \\ &= w\text{-}\max \mathrm{cl} \bigcup_{\substack{\mu\in\mathbb{R}^p_+\setminus\{0\}\\ \lambda\geqq 0}} Y_{H^-(\lambda,\mu)}. \end{aligned}$$

Remark 10 As can be readily seen, by defining inf A, for a set $A \in \mathbf{R}^p$, as essentially min cl$(A + \mathbf{R}^p_+)$ and similarly sup A as essentially min cl$(A - \mathbf{R}^p_+)$, we can have inf(P) = sup($\mathrm{D_{TS}}$) = sup ($\mathrm{D_N}$) = sup($\mathrm{D_J}$) under some appropriate stability condition [9].

See also

- ▶ Bi-objective Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ Lagrange, Joseph-Louis
- ▶ Lagrangian Multipliers Methods for Convex Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Gale D, Kuhn HW, Tucker AW (1951) Linear programming and the theory of games. In: Koopmans TC (ed) Activity Analysis of Production and Allocation. Wiley, New York, pp 317–329
2. Isermann H (1978) On some relations between a dual pair of multiple objective linear programs. Z Oper Res 22:33–41
3. Jahn J (1983) Duality in vector optimization. Math Program 25:343–353
4. Kawasaki H (1982) A duality theorem in multiobjective nonlinear prgramming. Math Oper Res 7:95–110
5. Nakayama H (1984) Geometric consideration of duality in vector optimization. J Optim Th Appl 44:625–655
6. Nakayama H (1999) Duality in multi-objective optimization. In: Gal T, Stewart TJ, Hanne T (eds) Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory and Applications. Kluwer, Dordrecht, pp 3.1–3.29
7. Nieuwenhuis JW (1980) Supremal points and generalized duality. Math Operationsforsch Statist Ser Optim 11:41–59
8. Sawaragi Y, Nakayama H, Tanino T (1985) Theory of multiobjective optimization. Acad Press, New York
9. Tanino T (1988) Supremum of a set in a multi-dimensional space. J Math Anal Appl 130:386–397
10. Tanino T, Sawaragi Y (1979) Duality theory in multiobjective programming. J Optim Th Appl 27:509–529

Multi-objective Optimization: Pareto Optimal Solutions, Properties

HAROLD P. BENSON
Department Decision and Information Sci., University Florida, Gainesville, USA

MSC2000: 90C29

Article Outline

Keywords

Multi-objective optimization; Multiple criteria decision making; Efficient solution; Pareto optimal solution; Noninferior solution; Nondominated solution; Weakly efficient solution; Weakly Pareto optimal solution; Weakly noninferior solution; Weakly nondominated solution; Properly efficient solution

The *multi-objective optimization* (*multiple criteria decision making*) problem is the problem of choosing a most preferred solution when two or more incommensurate, conflicting objective functions (criteria) are to be simultaneously maximized. A central difficulty in such problems is that, unlike in single objective maximization problems, there is no obvious or simple way to define the concept of a most preferred solution. Nevertheless, because the applications of multi-objective optimization abound, there has been great interest during the past 30 years in seeking appropriate definitions for a most preferred solution and in developing algorithms that aid the decision maker (DM) to find such a solution. These applications are in a wide variety of areas, including, for example, production planning, finance, environmental conservation, academic planning, nutrition planning, advertising, facility location, auditing, blending techniques, transportation planning, and scheduling, to name just a few.

There are several alternate mathematical formulations of the multi-objective optimization problem [13]. For purposes of modeling the deterministic multiple objective optimization problems found in management science/operations research, however, the most popular form of the problem is denoted

$$(\mathrm{V})\quad \begin{cases} \mathrm{VMAX} & [f_1(x), \dots, f_p(x)] \\ \text{s.t.} & x \in X. \end{cases}$$

Here, $p \geq 2$, X is a nonempty subset of $\mathbf{R}^n$, each f_j, $j = 1, \dots, p$, is a real-valued function defined on X or on a suitable set containing X, and VMAX indicates that we are to, in some as-yet unspecified sense, 'vector maximize' the vector

$$f(x) = [f_1(x), \dots, f_p(x)]$$

of *objective functions* (*criteria*) over X. The set X is called the set of *alternatives* or the *decision set*.

Of all of the solution concepts proposed for helping the DM find a most preferred solution for problem (V), the concept of efficiency has proven to be of overriding importance. An *efficient* (*Pareto optimal*, *noninferior*, *nondominated*) *solution* for problem (V) is a point $\overline{x} \in X$ such that there exists no other point $x \in X$ that satisfies $f(x) \geq f(\overline{x})$ and $f(x) \neq f(\overline{x})$. Letting X_E denote the set of all efficient points for problem (V), we see that whenever $\overline{x} \in X_E$, there is no other feasible point that does at least as well as $\overline{x}$ in all of the criteria for problem (V) and strictly better in at least one criterion. A point $\overline{x} \in X$ is called *dominated* when for some other point $x \in X$, $f(x) \geq f(\overline{x})$ and, for at least one $j = 1, \ldots, p$, $f_j(x) > f_j(\overline{x})$. Thus, we have the alternate definition for efficiency that states that a point $\overline{x}$ is an *efficient* solution for problem (V) when $\overline{x} \in X$ and there are no other points in X that dominate $\overline{x}$.

One of the reasons for the fundamental importance of the efficiency concept is that it has proven to be highly useful in a variety of algorithms for problem (V). Among these algorithms are the *satisficing* methods, *compromise programming*, most *interactive methods*, and the *vector maximization method*. The latter method, for instance, seeks to generate either all of X_E or key parts of X_E. The generated set is shown to the DM. Then, based upon the DM's internal utility (or value) function, the DM chooses from the generated set a most preferred solution. For details concerning these methods for problem (V), see [7,10,12,13,14].

In some cases, it is useful to consider a slightly relaxed concept of efficiency called weak efficiency. A point $\overline{x} \in X$ is called a *weakly efficient* (*weakly Pareto optimal*, *weakly noninferior*, *weakly nondominated*) *solution* for problem (V) when there is no other point $x \in X$ such that $f(x) > f(\overline{x})$. Let X_{WE} denote the set of all weakly efficient points for problem (V). Notice that X_E is a subset of X_{WE}. In some cases of problem (V), such as when the objective functions are ratios of linear functions, it is easier to analyze and generate points in X_{WE} than points in X_E.

Let U represent a utility function defined on the space $\mathbf{R}^p$ of the objective functions of problem (V). Suppose that U is *coordinatewise increasing*, i. e., that whenever $\overline{z}, z \in \mathbb{R}^p$ satisfy $\overline{z} \geq z$ and $\overline{z}_j > z_j$ for some $j = 1, \ldots, p$, then $U(\overline{z}) > U(z)$. Suppose that x^* is an optimal solution to the single objective problem

$$\text{(S)} \qquad \max_{x \in X} U[f_1(x), \ldots, f_p(x)].$$

Then x^* must be an efficient solution for problem (V) (cf. [11]).

The property in the previous paragraph explains to a great extent why the concept of efficiency is of such fundamental value. The assumption that the utility function U in the above paragraph is coordinatewise increasing implies that in problem (S), for each $j = 1, \ldots, p$, more of f_j is preferred to less of f_j. Thus, if we imagine that U is the utility (or value) function of the DM over the objective function space of problem (V), then the previous paragraph implies that whenever the DM prefers more to less in each objective function of problem (V), any point that maximizes the DM's utility for $f(x)$ over X must be an efficient point in problem (V). In short, as long as we know that the DM prefers more to less, we can confine the search for a most preferred solution to X_E. Although the utility function of the DM is generally not actually available, in virtually all applications the DM does, indeed, prefer more to less in each objective function of problem (V). Thus, in essentially all cases, any most preferred solution for problem (V) will be found in X_E.

Because of the central importance of efficiency, a great deal of effort has been made by researchers to delineate the properties of the efficient points and of the efficient set for problem (V). In what follows, we shall briefly highlight some of the most important of these properties.

Consider the single-objective optimization problem

$$\text{(W)} \qquad \begin{cases} \max & \displaystyle\sum_{j=1}^{p} w_j f_j(x), \\ \text{s.t.} & x \in X. \end{cases}$$

Here, w_j, $j = 1, \ldots, p$, are parameters, which are often thought of as weights associated with the objective functions f_j, $j = 1, \ldots, p$, of problem (V). A number of so-called *scalarization properties* for efficient points of problem (V) are expressed in terms of problem (W). To present some of these, another efficiency concept, called proper efficiency, is needed. A point x° is said to be a *properly efficient solution* for problem (V) when $x^\circ \in X_E$ and, for some sufficiently large number M, whenever $f_i(x) > f_i(x^\circ)$ for some $i = 1, \ldots, p$ and some $x \in X$, there exists some $j = 1, \ldots, p$ such that $f_j(x) < f_j(x^\circ)$ and

$$\frac{f_i(x) - f_i(x^\circ)}{f_j(x^\circ) - f_j(x)} \leq M.$$

In words, for each properly efficient solution of problem (V), for each criterion, the possible marginal gains in that criterion relative to the losses in the criteria that have losses *cannot all* be unbounded from above. Let

X_{PRE} denote the set of properly efficient solutions for problem (V), and let $w^{\mathsf{T}} = (w_1, \dots, w_p)$. Then some key scalarization properties are as follows.

1) If $\overline{x}$ is the unique optimal solution to problem (W) for some $w \geq 0$, $w \neq 0$, then $\overline{x} \in X_E$.
2) If $\overline{x}$ is an optimal solution to problem (W) for some $w \geq 0$, $w \neq 0$, then $\overline{x} \in X_{WE}$.
3) Assume that for each $j = 1, \dots, p$, f_j is a concave function on the convex set X. Then $\overline{x} \in X_{PRE}$ if and only if $\overline{x}$ is an optimal solution to problem (W) for some $w > 0$.
4) Under the assumptions in property 3), $\overline{x} \in X_{WE}$ if and only if $\overline{x}$ is an optimal solution to problem (W) for some $w \geq 0$, $w \neq 0$.
5) Under the assumptions of property 3), if $\overline{x} \in X_E$ but $\overline{x} \notin X_{PRE}$, then there exists a $w \geq 0$, $w \neq 0$ with $w_j = 0$ for at least one $j = 1, \dots, p$ such that $\overline{x}$ is an optimal solution to problem (W).
6) If each f_j, $j = 1, \dots, p$, is a linear function and X is a polyhedron, $X_{PRE} = X_E$.

The scalarization properties can be used for various purposes, including the generation of points in X_E, X_{WE} and X_{PRE}. For instance, when each f_j, $j = 1, \dots, p$, is a linear function and X is a polyhedron, from properties 3) and 6), points in X_E, including, at least potentially, all of X_E, can be generated by solving problem (W) as the parameter $w > 0$ is varied. Under the assumptions of property 3), the same process will generate points in X_{PRE}, including, at least potentially, all of X_{PRE}. However, from properties 3)–5), it is apparent that no such simple process for generating X_E exists, even under the assumptions of property 3). This is another motivation for the proper efficiency concept.

Another important issue in efficiency concerns testing. One may want to test a given point for efficiency in problem (V), and one may want to test whether X_E and X_{PRE} are empty or not. We will present several of the properties of efficiency that provide some of the theory for these tests. These properties all utilize the single-objective problem

$$(\mathrm{T}) \quad \begin{cases} \max & \displaystyle\sum_{j=1}^{p} f_j(x), \\ \text{s.t.} & f_j(x) \geq f_j(x^\circ), \\ & j = 1, \dots, p, \\ & x \in X. \end{cases}$$

Here, x° is an arbitrary element of $\mathbf{R}^n$. The properties are as follows.

7) The point $x^\circ \in \mathbf{R}^n$ belongs to X_E if and only if x° is an optimal solution to problem (T).
8) Suppose that $x^\circ \in X$ in problem (T), and that problem (T) has no finite maximum value. Then $X_{PRE} = \emptyset$ [1].
9) Suppose that the assumptions of property 3) hold, that $x^\circ \in X$ in problem (T), and that problem (T) has no finite maximum value. Then, if the set
$$Z = \left\{ z \in \mathbb{R}^p \colon\ z \leq f(x) \text{ for some } x \in X \right\}$$
is closed, $X_E = \emptyset$.
10) Assume that each f_j, $j = 1, \dots, p$, is a linear function and that X is a polyhedron. Suppose that $x^\circ \in X$ in problem (T), and that problem (T) has no finite maximum value. Then $X_E = \emptyset$.
11) Any optimal solution to problem (T) belongs to X_E.

Notice from these properties that solving problem (T) is a useful tool for both testing a point for efficiency and for investigating the issues of whether X_E and X_{PRE} are empty or not. In the case of testing a point x° for efficiency, property 7) shows that problem (T) can be used to obtain a definitive answer, i. e., using property 7), we will always detect whether or not $x^\circ \in X_E$. Furthermore, when property 7) shows that $x^\circ \notin X_E$, but problem (T) has an optimal solution x^*, then, by property 11), $x^* \in X_E$. Notice also that in this case, x^* dominates x°.

In the case of investigating whether or not X_E and X_{PRE} are empty, however, definitive answers cannot usually be obtained by using these properties. This is because none of the properties addresses the issue of whether or not X_E and X_{PRE} are empty when, instead of having an optimal solution or having no finite maximum value, problem (T) has a finite but unattained maximum value. The one case where the properties *can* be used to definitely detect whether or not X_E and X_{PRE} are empty is the case where the objective functions of problem (V) are all linear and X is a polyhedron. In that case, problem (T) cannot have a finite but unattained maximum value. Therefore, properties 7), 10) and 11) can be used to detect whether or not $X_E = X_{PRE}$ is empty in such cases.

One of the main challenges computationally to generating all or parts of X_E or X_{WE} for the DM to consider is that both X_E and X_{WE} are, except for trivial cases, *nonconvex sets*. Although some researchers have suggested ways to mitigate this problem [5], it generally remains a major stumbling block for algorithm development. In many common cases, however, X_E or X_{WE} possesses a useful, although less valuable, property than convexity upon which algorithms can be based. This property is called *connectedness*. In particular, a set $Z \subseteq \mathbf{R}^n$ is *connected* if, whenever A and B are nonempty subsets of $\mathbf{R}^n$ such that A has no points in common with the closure of B, and B has no points in common with the closure of A, $Z \neq A \cup B$. Some common cases of problem (V) where X_E or X_{WE} is connected are given in the following properties.

12) Assume that for each $j = 1, \dots, p$, f_j is a quasiconcave function on X, and that X is a compact convex set. Then X_{WE} is connected.
13) Assume that for each $j = 1, \dots, p$, f_j is a concave function on $\mathbf{R}^n$, and that X is a compact convex set. Then X_E is connected.

Recall that a concave function on a convex set is also quasiconcave on the set. Therefore, from property 12), it follows that X_{WE} is connected when each objective function in problem (V) is a concave function on X, and X is a compact, convex set.

There are a variety of other properties of efficient points and of the efficient set for problem (V). These include, for instance, density properties, stability-related properties, the *domination property* [2,3,8], and complete efficiency-related properties [4,6]. For further reading, see [5,7,9,10,12,13,14].

See also

- ▶ Bi-objective Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Benson HP (1979) An improved definition of proper efficiency for vector maximization with respect to cones. J Math Anal Appl 71:232–241
2. Benson HP (1983) On a domination property for vector maximization with respect to cones. J Optim Th Appl 39:125–132
3. Benson HP (1984) Errata corrige. J Optim Th Appl 43:477–479
4. Benson HP (1991) Complete efficiency and the initialization of algorithms for multiple objective programming. Oper Res Lett 10:481–487
5. Benson HP, Sayin S (1997) Towards finding global representations of efficient sets in multiple objective mathematical programming. Naval Res Logist 44:47–67
6. Benveniste M (1977) Testing for complete efficiency in a vector maximization problem. Math Program 12:285–288
7. Goicoechea A, Hansen DR, Duckstein L (1982) Multiobjective decision analysis with engineering and business applications. Wiley, New York
8. Henig MI (1986) The domination property in multicriteria optimization. J Math Anal Appl 114:7–16
9. Luc DT (1989) Theory of vector optimization. Springer, Berlin
10. Sawaragi Y, Nakayama H, Tanino T (1985) Theory of multiobjective optimization. Acad Press, New York
11. Soland RM (1979) Multicriteria optimization: A general characterization of efficient solutions. Decision Sci 10:26–38
12. Steuer R (1986) Multiple criteria optimization: Theory, computation and application. Wiley, New York
13. Yu PL (1985) Multiple-criteria decision making. Plenum, New York
14. Zeleny M (1982) Multiple criteria decision making. McGraw-Hill, New York

Multiparametric Linear Programming

VIVEK DUA, EFSTRATIOS N. PISTIKOPOULOS
Imperial College, London, UK

MSC2000: 90C31, 90C05

Article Outline

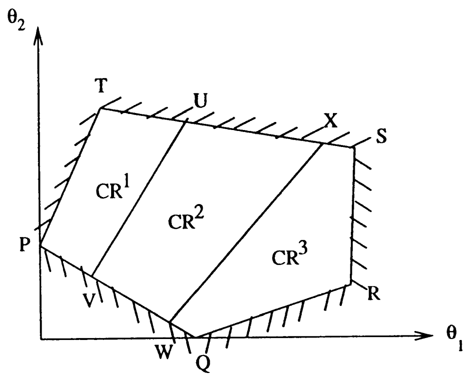
Keywords

Sensitivity analysis with respect to right-hand side changes; Critical region

In this article we will describe some results for sensitivity analysis and parametric programming for linear models. The solution approach that is described here is based upon the extension of simplex algorithm for linear programs (LP) [3,5]. Here we mention some references ([1,2,6,7,8,9,10,11,12,13,14,15,16,18,19,20,21], and [17]); however [3] is recommended for an extensive list of references and [4] for a historical outline on parametric linear programming.

We will consider right-hand side (RHS) multiparametric linear programming problems, where uncertain parameters are assumed to be bounded in a convex region. The solution algorithm is based upon characterizing the given initial convex region by a number of nonoverlapping smaller convex regions and obtaining optimal solutions associated with each of these regions. The basic assumptions for the application of the algorithm are:

- The given region must be finite and connected.
- One should be able to characterize at least one (smaller) region.
- One should be able to identify all regions that are adjacent to a given region.



Multiparametric Linear Programming, Figure 1
Definition of critical regions

Consider the following multiparametric linear programming problem, when parameters are present on the right-hand side of the constraints:

$$\begin{cases} z(\theta) = \min\limits_x & c^\top x \\ \text{s.t.} & Ax = b + F\theta \\ & x \geq 0 \\ & x \in \mathbb{R}^n, \quad \theta \in \mathbb{R}^s, \end{cases} \tag{1}$$

where x is a vector of continuous variables; A and F are constant matrices, and c and b are constant vectors of appropriate dimensions; θ is a vector of uncertain parameters, such that for each $\theta \in K$, $\theta \in \mathbf{R}^s$, (1) has a finite optimal solution, and has no optimal solution for $\theta \in \mathbf{R}^s - K$. Further, consider the following restriction on $\theta \in \Xi$, $\Xi = \{\theta : G\theta \leq g\}$, where G is a constant matrix and g is a constant vector; see Fig. 1 for a graphical interpretation for the two parametric case where θ is bounded in the region given by PQRST.

The simplex tableau associated with (1) is given as follows:

$$Yx - {}^\rho F\theta = x_B,$$
$$z + {}^\rho z^\top x - f_{m+1}\theta = z^{(\rho)},$$

where

$$\begin{aligned} &Y = B^{-1}A, \quad {}^\rho F = B^{-1}F, \quad x_B = B^{-1}b, \\ &z = c^\top x, \quad {}^\rho z = c_B^\top Y - c^\top, \\ &f_{m+1}^\top = c_B^\top {}^\rho F, \quad z^{(\rho)} = c_B^\top x_B, \end{aligned} \tag{2}$$

where ρ corresponds to the index of basic variables and B is the corresponding matrix. The (critical) region within which the above (optimal) tableau is valid can then be derived as follows. The *critical region*, CR, where an optimal solution, $z^{(\rho)}(\theta) = c_B^\top x_B(\theta)$, preserves its optimality, is given by the initial conditions on θ:

$$G\theta \leq g \tag{3}$$

together with the conditions of *primal feasibility*. The conditions of primal feasibility are derived as follows. The basis B is said to be *primal feasible* if the condition:

$$B^{-1}b(\theta) = x_B(\theta) \geq 0, \qquad (4)$$

where $b(\theta) = b + F\theta$ and $x_B(\theta) = x_B + {}^{\rho}F\theta$, is satisfied. Then using (2) and (4), the condition of primal feasibility is given by:

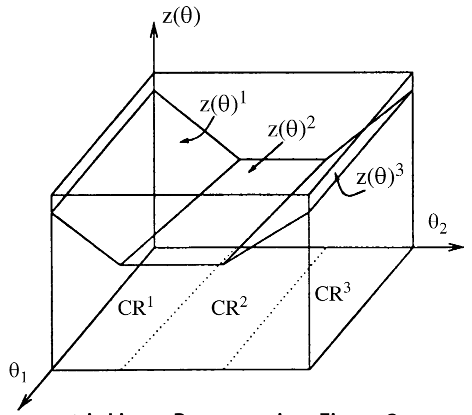
$$-{}^{\rho}F\theta \leq x_B. \qquad (5)$$

Thus, the critical region corresponding to ρ is given by (5) and (3). For illustration purposes, say in Fig. 1, the initial region of θ (condition (3)) is given by PQRST and the condition of primal feasibility is given by UVWX (condition (5)), then CR^2 is the corresponding critical region. Note that CR^2 is obtained by removing the redundant constraints, PT, QR and RS. In order to devise a procedure to obtain 'all' the critical regions (CR^1 and CR^3), and optimal solutions associated with them, we first state the following:

- Two optimal bases are said to be *neighbors* if
  - there exists some $\theta^* \in K$ such that both the bases are optimal, and,
  - it is possible to pass from one basis to another by one dual step.
- The critical regions associated with two different optimal bases are said to be *neighbors* if their corresponding bases are neighbors.
- Two neighboring critical regions lie in opposite half spaces.
- The optimal value function, $z(\theta)$, is continuous and convex; see Fig. 2 for a graphical interpretation for the case of two parameters.

Based upon the above statements, the solution algorithm for identifying all the critical regions can now be described. The algorithm consists of two major parts. In the first part, an initial feasible solution is obtained and the critical region which corresponds to the initial solution is characterized. The second part then starts with this critical region and identifies all the regions and corresponding optimal solutions. The major steps of the algorithm are as follows:

1) Find a feasible solution:
   - Solve (1) by treating θ as a free variable to obtain θ^*. If no feasible solution exists, stop; (1) is infeasible.
   - Fix $\theta = \theta^*$ and solve (1) to obtain an initial basis B and corresponding critical region.
2) Find all optimal solutions:
   - Construct two lists V and W, where V consists of those optimal bases whose neighboring bases have been identified, and W consists of those bases whose neighbors have yet not been identified.
   - Select any basis from W and identify all its neighboring bases. From all the identified bases, insert in W those bases which are neither in V nor in W. The optimal solutions (and corresponding critical regions) are then determined by moving from the basis to its neighbors by one dual step.
   - Repeat the procedure until $W = \{\emptyset\}$.



Multiparametric Linear Programming, Figure 2
z*(*θ*)is a continuous and convex function of *θ

See also

- ▶ Bounds and Solution Vector Estimates for Parametric NLPs
- ▶ Global Optimization in Multiplicative Programming
- ▶ Linear Programming
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Multiplicative Programming
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Parametric Global Optimization: Sensitivity
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Parametric Optimization: Embeddings, Path Following and Singularities
- ▶ Selfdual Parametric Method for Linear Programs

References

1. Gal T (1992) Putting the LP survey into perspective. OR/MS Today 19(6):93
2. Gal T (1992) Weakly redundant constraints and their impact on postoptimal analysis in linear programming. Europ J Oper Res 60:315–336
3. Gal T (1995) Postoptimal analyses, parametric programming, and related topics. de Gruyter, Berlin
4. Gal T (1997) Advances in sensitivity analysis and parametric programming. Kluwer, Dordrecht
5. Gal T, Nedoma J (1972) Multiparametric linear programming. Managem Sci 18:406–422
6. Granot D, Granot F, Johnson EL (1982) Duality and pricing in multiple right-hand choice linear programming problems. Math Oper Res 7:545–556
7. Greenberg HJ (1986) An analysis of degeneracy. Naval Res Logist Quart 33:635–655
8. Greenberg HJ (1993) How to analyze the results of linear programs - Part 1: Preliminaries. Interfaces 23(4):56–67
9. Greenberg HJ (1993) How to analyze the results of linear programs - Part 2: Price Interpretation. Interfaces 23(5):97–114
10. Greenberg HJ (1993) How to analyze the results of linear programs - Part 3: Infeasibility Diagnosis. Interfaces 23(6):120–139
11. Greenberg HJ (1994) How to analyze the results of linear programs - Part 4: Forcing Structures. Interfaces 24(1):121–130
12. Greenberg HJ (1994) The use of optimal partition in linear programming solution for postoptimal analysis. Oper Res Lett 15:179–185
13. Greenberg HJ (1996) The ANALYZE rulebase for supporting LP analysis. Ann Oper Res 65:91–126
14. Hansen PM, Labbe M, Wendell RE (1989) Sensitivity analysis in multiple objective linear programming: the tolerance approach. Europ J Oper Res 38(1):63–69
15. Magnati TL, Orlin JB (1988) Parametric linear programming and anti-cycling pivoting rules. Math Program 41:317–325
16. Murty K (1980) Computational complexity of parametric linear programming. Math Program 19:213–219
17. Roos C, Terlaky T, Vial -Ph J (1997) Theory and algorithms for linear optimization, an interior point approach. Wiley, New York
18. Wang H-F, Huang C-S (1993) Multiparametric analysis of the maximum tolerance in a linear programming problem. Europ J Oper Res 67(1):75–87
19. Ward JE, Wendell RE (1990) Approaches to sensitivity analysis in linear programming. Ann Oper Res 27:3–38
20. Wendell RE (1985) The tolerance approach to sensitivity analysis in linear programming. Managem Sci 31:564–578
21. Wendell RE (1997) Linear programming 3: The tolerance approach. In: Gal T, Greenberg HJ (eds) Advances in Sensitivity Analysis and Parametric Programming. Kluwer, Dordrecht

Multiparametric Mixed Integer Linear Programming

VIVEK DUA, EFSTRATIOS N. PISTIKOPOULOS
Imperial College, London, UK

MSC2000: 90C31, 90C11

Article Outline

Keywords

Parametric bounds; Branch and bound; Comparison of parametric solutions

In this article we describe theoretical and algorithmic developments in the field of parametric programming for linear models involving 0–1 integer variables. We will consider two cases of the problem: single parametric (when a single uncertain parameter is present) and multiparametric (when more than one uncertain parameters are present in the model). For the case when a single uncertain parameter is present, solution approaches are based upon

a) enumeration [11,12,13];
b) cutting planes [6]; and
c) branch and bound techniques [8,10].

For the multiparametric case, solution algorithm that has been proposed is based upon branch and bound fundamentals [1,2]. While most of the work on single parametric problems has been reviewed in the two excellent papers [5] and [7], and has been borrowed here for the sake of completeness, the work on multiparametric problems, the focus of this article, is quite recent and is described in detail. It may be mentioned that while solution approaches for single parametric case are available for uncertainty in objective function coefficients or right-hand side of constraints, for the case of more than one uncertain parameter the solution approach is available only for the right-hand side case. Next we will describe solution approaches for

a) single parametric mixed integer linear programs for objective function coefficients parametrization; and
b) single parametric pure integer programs when the uncertain parameter is present on the right-hand side of the constraints.

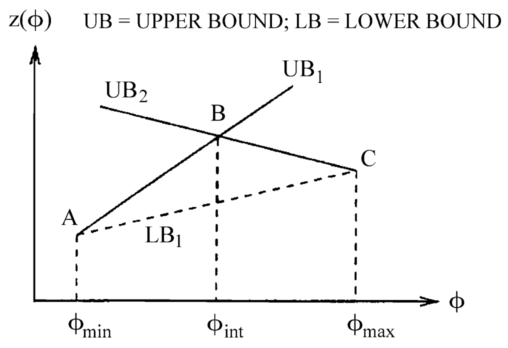
These illustrate some concepts which are based upon some basic observations. For other solution approaches, see the literature cited above. Finally we will present a solution approach for right-hand side multiparametric mixed integer linear programs.

Mixed Integer Linear Programming Problems Involving a Single Uncertain Parameter in Objective Function Coefficients

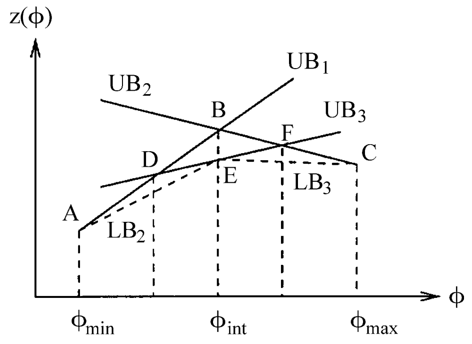
These can be stated as follows:

$$\begin{cases} z(\phi) & = \min\limits_{x,y} (c^\top + c'\phi)x + d^\top y \\ \text{s.t.} & Ax + Ey \le b, \\ & x \in \mathbb{R}^n, \quad y \in \{0,1\}^l, \\ & \phi_{\min} \le \phi \le \phi_{\max}, \end{cases} \tag{1}$$

where x is a vector of continuous variables; y is the vector of 0–1 integer variables; ϕ is a scalar uncertain parameter bounded between its lower and upper bounds $\phi_{\min}$ and $\phi_{\max}$ respectively; A is an $(m \times n)$ matrix; E is an $(m \times l)$ matrix; c, c', d and b are vectors of appropriate dimensions. Solution procedure for (1) is based upon following two features of the formulation in (1). First feature of this formulation is that, since the uncertain parameter is present in the objective function only, the feasible region of (1) remains constant for all the fixed values of ϕ in $[\phi_{\min}, \phi_{\max}]$. And the second feature is that, the optimal value of (1) for $\phi_{\min} \le \phi \le \phi_{\max}$ is piecewise linear, continuous, and concave on its finite domain. The solution is then approached by deriving valid upper and lower bounds, using the concavity property of the objective function value, and sharpening these bounds until they converge to the same value, as described next. Solving (1) for ϕ fixed at its endpoints $\phi_{\min}$ and $\phi_{\max}$, gives upper bounds AB and BC respectively (see Fig. 1); and a linear interpolation, AC, between the endpoints provides a lower bound to the solution. The region ABC within which the solution will



Multiparametric Mixed Integer Linear Programming, Figure 1
Derivation of bounds



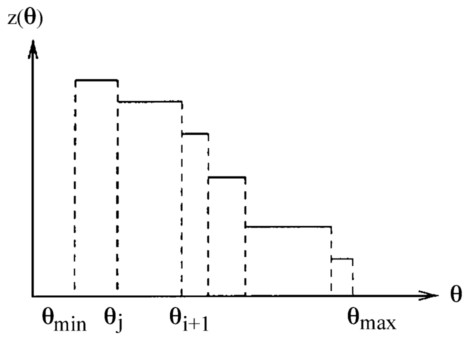
Multiparametric Mixed Integer Linear Programming, Figure 2
Sharpening of bounds

lie is then reduced by solving (1) at ϕ_{int}, the intersection point of two upper bounds AB and BC. This results (see Fig. 2) in two smaller regions, ADE and EFC, within which the solution will exist. This procedure is continued until the difference between upper and lower bounds becomes zero.

Integer programming problem involving a single uncertain parameter on the right-hand side of the constraints can be stated as follows:

$$\begin{cases} z(\theta) & = \min\limits_{y} d^\top y \\ \text{s.t.} & Ey \le b + r\theta, \\ & \theta_{\min} \le \theta \le \theta_{\max} \\ & y \in \{0,1\}^l, \end{cases} \tag{2}$$

where r is a scalar constant and θ is a scalar uncertain parameter bounded between $\theta_{\min}$ and $\theta_{\max}$ respectively. For a special case of (2) when $r \ge 0$, it may be



Multiparametric Mixed Integer Linear Programming, Figure 3
Step function nature of objective function value

noted that as θ is increased from $\theta_{\min}$ to $\theta_{\max}$, the feasible region will enlarge, and hence the objective function value will decrease or remain the same, i. e., $z(\theta_i) \geq z(\theta_{i+1})$ for $\theta_i \leq \theta_{i+1}$. Further, since only integer variables are present in (2), a solution will remain optimal for some interval of θ and then suddenly another solution will become optimal, and remain so for the next interval (see Fig. 3). The problem thus reduces to solving (2) at an end point, say $\theta_{\min}$, and then finding a point θ_i at which the current solution becomes infeasible. Solving (2) at $\theta_i + \epsilon$ will give another integer solution. This procedure is continued until we hit the other end point, $\theta_{\max}$.

Consider a multiparametric mixed integer linear programming problem (mp-MILP) of the following form:

$$\begin{cases} z(\theta) & = \min\limits_{x,y} c^\top x + d^\top y \\ \text{s.t.} & Ax + Ey \leq b + F\theta, \\ & G\theta \leq g, \\ & x \in \mathbb{R}^n, \quad y \in \{0,1\}^l, \quad \theta \in \mathbb{R}^s, \end{cases} \tag{3}$$

where θ is a vector of uncertain parameters; F is an $(m \times s)$ matrix, G is an $(r \times s)$ matrix, and g is a constant vector. Solving (3) implies obtaining the optimal solution to (3) for every θ that lies in $\Xi = \{\theta{:}G\theta \leq g, \theta \in \mathbf{R}^s\}$. The algorithm for the solution of (3) proposed in [1] is based upon simultaneously using the concepts of

- branch and bound method for solving mixed integer linear programming (MILP) problems (see, e. g., [9]); and,
- simplex algorithm for solving multiparametric linear programming (mp-LP) problems [4].

While a solution of (3) by relaxing the integrality condition on y (at the root node) represents a *parametric lower bound*, a solution where all the y variables are fixed (e. g., at a terminal node) represents a *parametric upper bound*. The algorithm proceeds from the root node (lower bound) towards terminal nodes (upper bound) by fixing y variables at the intermediate nodes. The complete enumeration of the tree is avoided by fathoming those intermediate nodes which guarantee a suboptimal solution.

At the root node, by relaxing the integrality condition on y, i. e., considering y as a continuous variable bounded between 0 and 1, (3) is transformed to an mp-LP of the following form:

$$\begin{cases} \check{z}(\theta) & = \min\limits_{x,\check{y}} c^\top x + d^\top \check{y} \\ \text{s.t.} & Ax + E\check{y} \leq b + F\theta, \\ & G\theta \leq g, \\ & 0 \leq \check{y} \leq 1, \\ & x \in \mathbb{R}^n, \quad \theta \in \mathbb{R}^s. \end{cases} \tag{4}$$

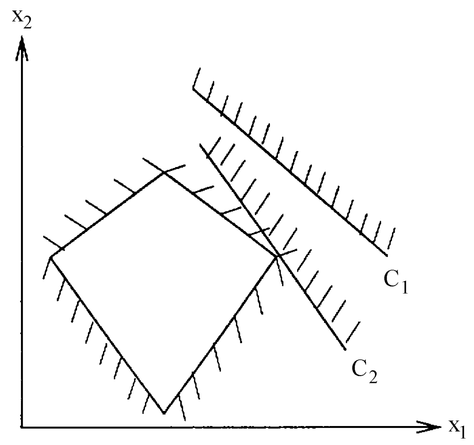
The solution of (4), given by linear parametric profiles, $\check{z}(\theta)^i$, valid in their corresponding *critical regions*, $\check{\mathrm{CR}}^i$, represents a parametric lower bound.

Similarly, at a node where all y are fixed, $y = \widehat{y}$, (3) is transformed to an mp-LP of the following form:

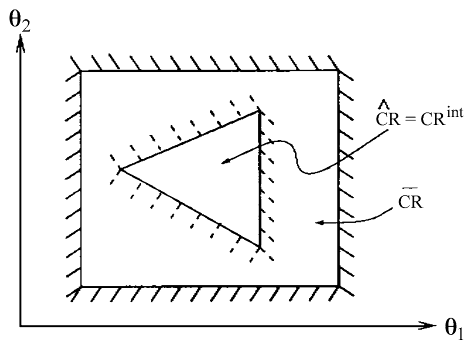
$$\begin{cases} \widehat{z}(\theta) & = \min\limits_{x,\widehat{y}} c^\top x + d^\top \widehat{y} \\ \text{s.t.} & Ax + E\widehat{y} \leq b + F\theta, \\ & G\theta \leq g, \\ & \widehat{y} = \{0,1\}^l, \\ & x \in \mathbb{R}^n, \quad \theta \in \mathbb{R}^s. \end{cases} \tag{5}$$

The solution of (5), $\widehat{z}(\theta)^i$, valid in its corresponding critical regions, $\widehat{\mathrm{CR}}^i$, represents a parametric upper bound.

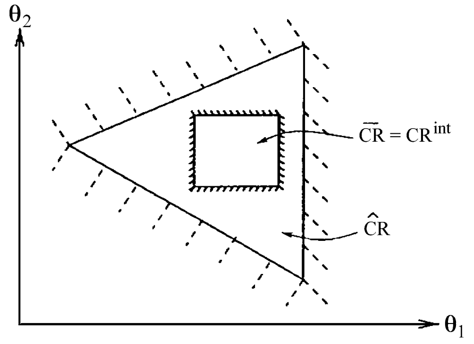
Starting from the root node, some of the y variables are systematically fixed (to 0 and 1) to generate intermediate nodes of the branch and bound tree. At an intermediate node, where some y are fixed and some are



Multiparametric Mixed Integer Linear Programming, Figure 4
Redundant constraints



Multiparametric Mixed Integer Linear Programming, Figure 5
Definition of CR^{int}; Case 1



Multiparametric Mixed Integer Linear Programming, Figure 6
Definition of CR^{int}; Case 2

relaxed, an mp-LP of the following form is formulated:

$$\begin{cases} \overline{z}(\theta) & = \min\limits_{x,y} c^\top x + d_j^\top \widehat{y}_j + d_k^\top \check{y}_k \\ \text{s.t.} & Ax + E_j \widehat{y}_j + E_k \check{y}_k \leq b + F\theta, \\ & G\theta \leq g, \\ & \widehat{y}_j = \{0,1\}, \\ & 0 \leq \check{y}_k \leq 1, \\ & x \in \mathbb{R}^n, \quad \theta \in \mathbb{R}^s, \end{cases} \tag{6}$$

where the subscripts j and k correspond to y that are fixed and y that are free, respectively. The solution at an intermediate node, $\overline{z}(\theta)^i$, valid in its corresponding critical regions, $\overline{\text{CR}}^i$, is then analyzed, to decide whether to explore subnodes of this intermediate node or not, by using the following fathoming criteria. A given space in any node can be discarded if one of the following holds:

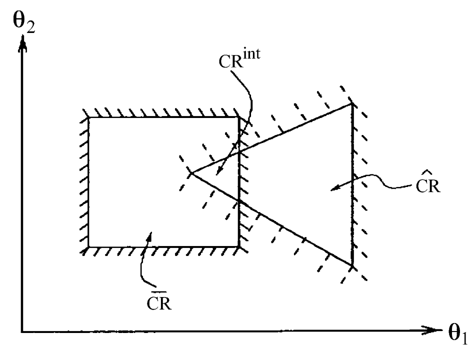
- (*infeasibility criterion*) Problem (6) is infeasible in the given space.
- (*integrality criterion*) An integer solution is found in the given space.
- (*dominance criterion*) The solution of the node is greater than the current upper bound in the same space.

If all the regions of a node are discarded the node can be fathomed. While the first two fathoming criteria (Infeasibility and Integrality) are easy to apply, in order to apply the third one (dominance criteria) we need a comparison procedure, which is described next.

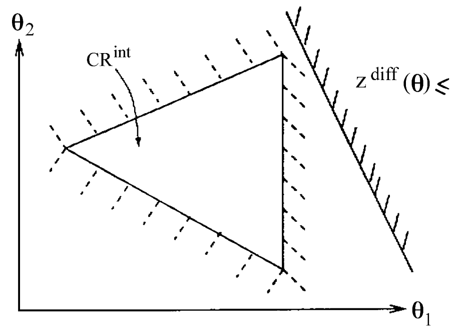
The comparison procedure consists of two steps. In the first step, a region, $\text{CR}^{\text{int}} = \widehat{\text{CR}} \cap \overline{\text{CR}}$, where the solution of the intermediate node and the current upper bound are valid is defined. This is achieved by removing the redundant constraints from the set of constraints which define $\widehat{\text{CR}}$ and $\overline{\text{CR}}$ (for a procedure to eliminate redundant constraints see [3]); graphical interpretation of redundant constraints is given in Fig. 4, where C_1 is a strongly redundant constraint and C_2 is a weakly redundant constraint.

The results of this *redundancy test*, which belong to one of the following 4 cases, are then analyzed as follows:

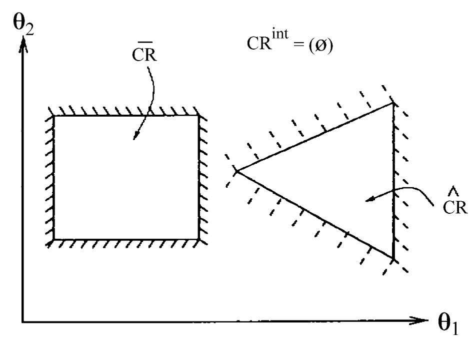
- (case 1; Fig. 5) All constraints from $\overline{\text{CR}}$ are redundant. This implies that $\overline{\text{CR}} \supseteq \widehat{\text{CR}}$, and therefore $\text{CR}^{\text{int}} = \widehat{\text{CR}}$.



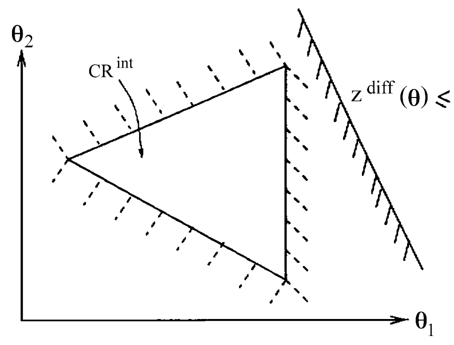
Multiparametric Mixed Integer Linear Programming, Figure 7
Definition of CR^{int}; Case 3



Multiparametric Mixed Integer Linear Programming, Figure 9
Compare $\bar{z}(\theta) : \hat{z}(\theta)$; Case 1



Multiparametric Mixed Integer Linear Programming, Figure 8
Definition of CR^{int}; Case 4



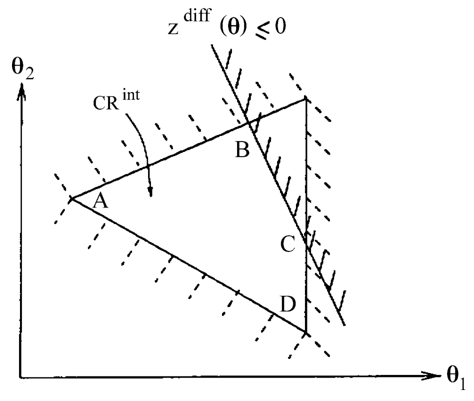
Multiparametric Mixed Integer Linear Programming, Figure 10
Compare $\bar{z}(\theta) : \hat{z}(\theta)$; Case 2

- (case 2; Fig. 6) All constraints from $\widehat{CR}$ are redundant. This implies that $\widehat{CR} \supseteq \overline{CR}$, and therefore $CR^{int} = \overline{CR}$.
- (case 3; Fig. 7) Constraints from both regions are nonredundant. This implies that two spaces intersect with each other, and CR^{int} is given by the space delimited by the nonredundant constraints.
- (case 4; Fig. 8) The problem is infeasible. This implies that two spaces are apart from each other and $CR^{int} = \{\emptyset\}$.

Once CR^{int} has been defined, the second step is to compare $\bar{z}$ to $\hat{z}$, so as to find which of the two is lower. This is achieved by defining a new constraint:

$$z^{diff}(\theta) = \bar{z}(\theta) - \hat{z}(\theta) \leq 0$$

and checking for redundancy of this constraint in CR^{int}. This redundancy test results in following 3 cases:



Multiparametric Mixed Integer Linear Programming, Figure 11
Compare $\bar{z}(\theta) : \hat{z}(\theta)$; Case 3

- (case 1; Fig. 9) The new constraint is redundant. This implies that $\bar{z}(\theta) \leq \hat{z}(\theta)$ and therefore the space must be kept for further analysis.

- (case 2; Fig. 10) The problem is infeasible. This implies that $\overline{z}(\theta) \geq \widehat{z}(\theta)$ and therefore the space can be discarded from further analysis.
- (case 3; Fig. 11) The new constraint is nonredundant. This implies that $\overline{z}(\theta) \leq \widehat{z}(\theta)$ in *ABCD*, and therefore the rest of the space can be discarded from further analysis.

Based upon the above theoretical framework, the steps of the algorithm can be summarized as follows:

```
1  Set an upper bound of $\widehat{z}(\theta) = \infty$.
2  Solve the fully relaxed problem (4).
   IF an integer solution is found in a critical region, THEN update the upper bound and discard the region from further analysis.
3  Fix one of the $y$ variables to 0 and 1 to create two new nodes.
   IF no new nodes can be generated, THEN stop.
4  Solve the resulting problem (6).
   IF the problem is infeasible THEN go back to Step 3,
   ELSE compare the solution to the current upper bound.
5  IF all regions from a node have been analyzed, THEN go to Step 3.
```

See also

- ▶ Bounds and Solution Vector Estimates for Parametric NLPs
- ▶ Branch and Price: Integer Programming with Column Generation
- ▶ Decomposition Techniques for MILP: Lagrangian Relaxation
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multiparametric Linear Programming
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Parametric Global Optimization: Sensitivity
- ▶ Parametric Linear Programming: Cost Simplex Algorithm
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Parametric Optimization: Embeddings, Path Following and Singularities
- ▶ Selfdual Parametric Method for Linear Programs
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-dependent Traveling Salesman Problem

References

1. Acevedo J, Pistikopoulos EN (1997) A multiparametric programming approach for linear process engineering problems under uncertainty. Industr Eng Chem Res 36:717–728
2. Acevedo J, Pistikopoulos EN (1999) An algorithm for multiparametric mixed integer linear programming problems. Oper Res Lett 24:139–148
3. Gal T (1995) Postoptimal analyses, parametric programming, and related topics. de Gruyter, Berlin
4. Gal T, Nedoma J (1972) Multiparametric linear programming. Managem Sci 18:406–422
5. Geoffrion AM, Nauss R (1977) Parametric and postoptimality analysis in integer linear programming. Managem Sci 23(5):453–466
6. Holm S, Klein D (1984) Three methods for postoptimal analysis in integer linear programming. Math Program Stud 21:97–109
7. Jenkins L (1990) Parametric methods in integer linear programming. Ann Oper Res 27:77–96
8. Marsten RE, Morin TL (1977) Parametric integer programming: The right-hand side case. Ann Discret Math 1:375–390
9. Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
10. Ohtake Y, Nishida N (1985) A branch-and-bound algorithm for 0-1 parametric mixed-integer programming. Oper Res Lett 4(1):41–45
11. Piper CJ, Zoltners AA (1976) Some easy postoptimality analysis for zero-one programming. Managem Sci 22(7):759–765
12. Roodman GM (1972) Postoptimality analysis in zero-one programming by implicit enumeration. Naval Res Logist Quart 19:435–447

13. Roodman GM (1974) Postoptimality analysis in zero-one programming by implicit enumeration: The mixed-integer case. Naval Res Logist Quart 21:595–607

Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach

John L. Klepeis, Christodoulos A. Floudas
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 92C40, 65K10

Article Outline

Keywords

Protein folding; Multiple minima; Global optimization; αBB

Motivation

Proteins are arguably the most complex molecules in nature. This complexity arises from an intricate balance of intra- and inter-molecular interactions that define the native three-dimensional structure of the system, and subsequently its biological functionality. The underlying goal of *protein folding* research is to understand the formation of these native tertiary structures. Genetic engineering can be used to produce proteins with specific amino acid sequences. The next step involves developing the link between the primary protein sequence and the native structure. The ability to predict the folding of proteins promises to have important practical and theoretical ramifications, especially in the areas of medicinal and biophysical chemistry.

Experimental studies have shown that proteins, under native physiological conditions, spontaneously refold to their unique, native structure after denaturation. This implies that the formation of the native structure is controlled primarily by the amino acid sequence. According to Anfinsen's hypothesis the native structure is in a state of thermodynamic equilibrium corresponding to the conformation with the lowest free energy. Through mathematical modeling of protein interaction energies, the protein folding problem can be addressed as a *conformational search* for the global minimum energy.

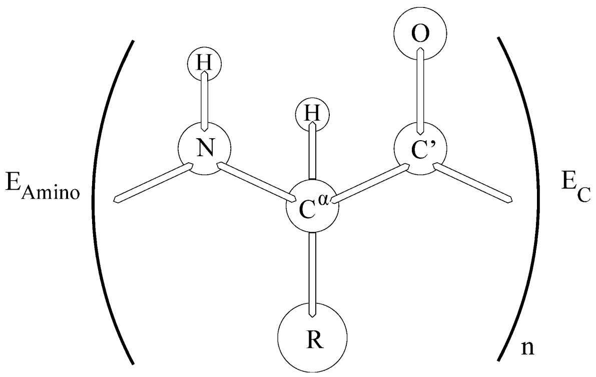
There exists two fundamental problems associated with protein folding in the context of a conformational search. The first is the ability to correctly model protein interactions using detailed mathematical equations. The second is associated with searching the highly nonconvex energy hypersurface that describes a given protein. This complexity, coupled with an exponential growth in the number of local minima as the size of protein increases, has become known as the *multiple minima* problem. There exists an obvious need for the development of efficient global optimization techniques. An efficient method which has been successfully applied to detailed atomistic models of protein folding is the αBB [1,2,3,17] global optimization algorithm.

Mathematical Description

Proteins are essentially polymer chains composed of a predefined set of amino acid residues in which neighboring residues are linked by peptidic bonds. Naturally occurring proteins consist of only 20 different amino acid residues, and the form of the side chain R (e. g., methyl, butyl, benzoic, etc.) defines the differences between these constituent groups. The chemical structure of a generic protein is illustrated in Fig. 1. The repeating unit $- NC^{\alpha}C' -$ defines the backbone of the protein. The protein also possesses amino and carboxyl end groups, denoted by E_{Amino} and E_{Carboxyl}, respectively.

The geometry of a protein can be fully described by assigning a three-dimensional coordinate vector r_i:

$$r_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} .$$



Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach, Figure 1
Generic primary protein structure

These r_i specify the position of each atom in the protein molecule. The bond vector between two atoms (i, j) connected with a covalent bond is defined as:

$$r_{ij} = \begin{pmatrix} x_j - x_i \\ y_j - y_i \\ z_j - z_i \end{pmatrix}.$$

The corresponding bond length is then equal to the Euclidean distance between these two atoms:

$$|r_{ij}| = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}$$

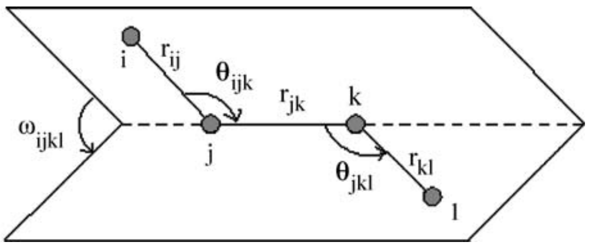
A covalent bond angle, θ_{ijk}, formed by the two adjacent bond vectors r_{ij} and r_{jk} can be computed by the following formulas:

$$\cos(\theta_{ijk}) = \frac{r_{ij} \cdot r_{jk}}{|r_{ij}|\,|r_{jk}|}, \quad \sin(\theta_{ijk}) = \frac{r_{ij} \times r_{jk}}{|r_{ij}|\,|r_{jk}|}.$$

Here, $r_{ij} \cdot r_{jk}$ is the dot product of the bond vectors r_{ij} and r_{jk} and $r_{ij} \times r_{jk}$ is the cross product.

The dihedral angle ω_{ijkl} measures the relative orientation of two adjacent covalent angles θ_{ijk} and θ_{jkl}. This angle is defined as the angle between the normals through the planes defined by atoms i, j, k and j, k, l respectively, and can be calculated from the following relations:

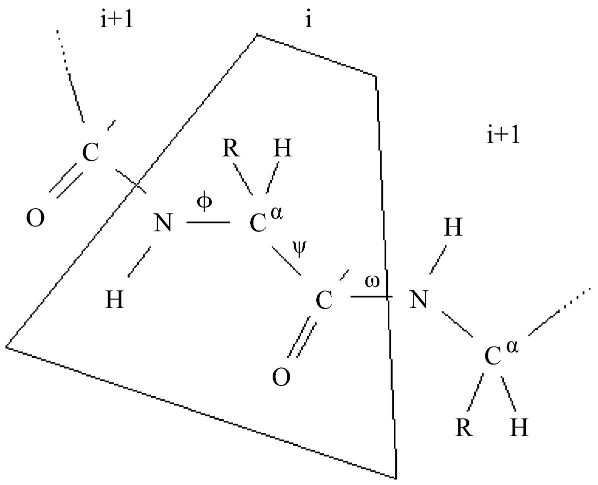
$$\cos(\omega_{ijkl}) = \frac{(r_{ij} \times r_{jk}) \cdot (r_{jk} \times r_{kl})}{|r_{ij} \times r_{jk}|\,|r_{jk}, \times r_{kl}|},$$
$$\sin(\omega_{ijkl}) = \frac{(r_{kl} \times r_{ij}) \cdot r_{jk}\,|r_{jk}|}{|r_{ij} \times r_{jk}|\,|r_{jk} \times r_{kl}|}.$$



Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach, Figure 2
Illustration of dihedral angle

An alternative to specifying the coordinate vector for all atoms in a protein molecule is to set bond lengths, covalent bond angles and independent dihedral angles. A common approximation is to assume rigid bond lengths and bond angles so that the dihedral angles can be used to fully characterize the shape of the protein molecule.

The names of the dihedral angles of a protein chain follow a standard nomenclature. The dihedral angle between the normals of the planes formed by atoms $C_{i-1}{}'N_iC_i^\alpha$ and $N_iC_i^\alpha C_i'$ respectively, is called ϕ_i, where $i-1$ and i are two adjacent amino acid residues. The angle defined by the planes $N_iC_i^\alpha C_i'$ and $C_i^\alpha C_i' N_{i+1}$, respectively, is called ψ_i, where i and $i+1$ are two adjacent amino acid residues. Also, ω_i is the dihedral angle defined by the planes $C_i^\alpha\, C_i'\, N_{i+1}$ and $C_i'N_{i+1}C_{i+1}^\alpha$.



Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach, Figure 3
Dihedral angle conventions

The letter χ is utilized to denote the dihedral angles which are associated with the side groups R_i. Finally, the letter θ is used to name the dihedral angles associated with the two end groups. These conventions are illustrated in Fig. 3.

Potential Energy Modeling

A number of empirically based *molecular mechanics* models have been developed for protein systems, including AMBER [24], CHARMM [7], ECEPP/3 [19], GROMOS [11], MM3 [4]. These models, also known as *force fields*, are typically expressed as summations of several potential energy components, with the mathematical form of individual energy terms based on the phenomenological nature of that term. A general total potential energy equation should include terms for bond stretching (E_{bond}), angle bending (E_{angle}), torsion (E_{tor}) and nonbonded (E_{nb}) interactions:

$$E_{\text{potential}} = E_{\text{bond}} + E_{\text{angle}} + E_{\text{tor}} + E_{\text{nb}}$$

When rigid body approximations are employed, bond stretching and angle bending energies can be neglected. For these force fields, torsion angles define a set of independent variables that effectively describe any protein conformation. This approximately reduces the number of variables by a factor of 3 over those force fields that use a Cartesian coordinate system to describe flexible molecular geometries.

One example of a rigid body atomistic level potential energy model is the ECEPP/3 force field. In this case, the nonbonded energy terms, E_{nb}, include electrostatic, E_{elec}, van der Waals, E_{vdw}, and hydrogen bonding, E_{hbond}, interactions. These energies are calculated for those atoms that are separated by more than two atoms; that is, the atoms possess at least a 1–4 relationship. Electrostatic energies, E_{elec}, are calculated as Coulombic forces based on atomic point charges:

$$E_{\text{elec}} = \frac{Q_i Q_j}{\epsilon R_{ij}}$$

Here, Q_i and Q_j represent the two point charges, while R_{ij} equals the distance between these two points. The ϵ term describes the dielectric nature of the protein environment.

General nonbonded van der Waals interactions, E_{vdw}, are modeled using a 6–12 Lennard–Jones potential energy term, which consists of a repulsion and attraction term:

$$E_{\text{vdw}} = \epsilon_{ij}\left[\left(\frac{R^*_{ij}}{R_{ij}}\right)^{12} - 2\left(\frac{R^*_{ij}}{R_{ij}}\right)^{6}\right].$$

The energy minimum for a given atomic pair is described by the potential depth, ϵ_{ij}, and position, R^*_{ij}. For those atomic pairs that may form a hydrogen bond, the 6–12 potential energy term is replaced by a modified 10–12 Lennard–Jones type term:

$$E_{\text{hbond}} = \epsilon_{ij}\left[5\left(\frac{R^*_{ij}}{R_{ij}}\right)^{12} - 6\left(\frac{R^*_{ij}}{R_{ij}}\right)^{10}\right].$$

Finally, corrective torsional energies, E_{tor}, which are represented by a three term Fourier series expansion, are also added:

$$E_{\text{tor}} = \frac{E_1}{2}(1-\cos\phi) + \frac{E_2}{2}(1-\cos 2\phi) + \frac{E_3}{2}(1-\cos 3\phi).$$

Each term can be interpreted physically. The 1-x (cos ϕ) symmetry term accounts for those nonbonded interactions not included in general nonbonded terms. The 2-x (cos 2 ϕ) symmetry term is related to the interactions of orbitals, while the 3-x (cos 3 ϕ) symmetry term describes steric contributions.

Other specific potential energy terms may also be added to the general energy equation depending on the exact protein sequence. For example, the formation of disulfide bridges can be enforced by adding a penalty term to constrain the values of particular atomic distances. Correction terms have also been used to adjust conformational energies according to the configurations of proline and hydroxyproline residues.

Solvation Energy Modeling

In general, the energetic description of a protein must also include *solvation effects*. A theoretically simple approach would be to explicitly surround the peptide with solvent molecules and compute potential energy contributions for intra-and inter-molecular interactions. These explicit calculations tend to greatly increase the computational cost of the simulation. In addition, solvent configurations are not rigid, so these calculations must consider an average solvent-peptide configuration, which is typically generated by a number of *Monte-Carlo* (MC) or molecular dynamics (MD) simulations [14]. Therefore, most simulations of this type are limited to restricted conformational searches.

An alternative way for effectively considering average solvent effects is to use implicit solvation models. One complication involves the solvent's influence on electrostatic interaction energies because of the implicit relationship between dielectric effects and solvation. A simple solution has been to modify the representation of the dielectric term. In reality, however, the rigorous treatment of electrostatic interactions involves the solution of the Poisson–Boltzmann equation.

Other simple and computationally feasible implicit solvation models are based on empirical representations of the solvation energy. In these cases, the solvation energy of each functional group is related to the interaction of the solvent with a hydration shell for the particular group. The individual terms are then summed together to provide a total solvation energy for the system. These solvation contributions can be described by the following general equation:

$$E_{\text{solv}} = \sum_{i=1}^{N} S_i \sigma_i .$$

Typically, S_i represents either the solvent-accessible surface area, A_i, or the solvent-accessible volume of hydration layer, VHS_i, for the functional group, and σ_i is an empirically derived free energy density parameter.

A number of algorithms have been developed for calculating solvent-accessible surface areas [8,9,22]. Although several of these are relatively efficient, the appearance of discontinuities has been one complication in considering solvent accessible surface areas. In addition, a large number of parameterization strategies (JRF, OONS, WE, etc.) have been used to derive appropriate σ_i parameters [21,23,25]. In the case of the JRF parameter set, discontinuities can be avoided because the surface-accessible solvation energies are only included at local minimum conformations [23]. This is because the parameters were derived from low energy solvated configurations of actual tetrapeptides.

Several methods have also been developed for calculating the hydration volumes and corresponding free energy parameters [6,12]. A recent and computationally inexpensive method, RRIGS, is based on a Gaussian approximation for the volume of a hydration layer [6]. This method also inherently avoids numerical problems associated with possible discontinuities so that the solvation energy contributions can easily be added at every step of local minimizations.

Problem Formulation

For protein folding, the *energy minimization* problem can be formulated as a nonconvex, nonlinear global optimization problem in which the energy, E, must be globally minimized with respect to the dihedral angles of the protein:

$$\begin{cases} \min & E(\phi_i, \psi_i, \omega_i, \chi_i^k, \theta_j^N, \theta_j^C) \\ \text{subject to} & -\pi \le \phi_i \le \pi \\ & -\pi \le \psi_i \le \pi \\ & -\pi \le \omega_i \le \pi \\ & -\pi \le \chi_i^k \le \pi \\ & -\pi \le \theta_j^N \le \pi \\ & -\pi \le \theta_j^C \le \pi . \end{cases}$$

The index $i = 1, \ldots, N_{\text{RES}}$ defines the number of residues, N_{RES}, in the protein. In addition, $k = 1, \ldots, K^i$ denotes the number of dihedral angles in the side chain of the ith residue, and $j = 1, \ldots, J^N$ and $j = 1, \ldots, J^C$ indicates the indices of the amino and carboxyl end groups, respectively. The energy, E, represents the total potential energy function, $E_{\text{potential}}$, plus the free energy of solvation, E_{solv}. In most cases, this is the exact formulation; that is, energetic and gradient contributions can be added at each step of the minimization. However, in the case of surface-accessible hydration using the JRF parameters, the potential energy function is minimized before adding the hydration energy contributions. In other words, gradient contributions from solvation are not considered.

Even after reducing this optimization problem to a function of internal variables, the multidimensional surface that describes the energy function possesses an astronomically large number of local minima. In addition, evaluation of the energy, especially with the addition of solvation, is computationally expensive, which makes even local minimization slow. A large number of techniques have been developed to search this nonconvex conformational space. Many methods employ stochastic search procedures, while others rely on simplifications of the potential model and/or mathematical transformations. In addition, the use of statistical and/or heuristic conformational information is often required. In general, the major limitation is that there is no guarantee for convergence to the global minimum energy structure. A number of recent reviews have focused on global optimization issues for these systems [10,20].

The αBB global optimization approach has been extremely effective in identifying global minimum energy conformations of peptides described by detailed atomistic models. The development of this deterministic *branch and bound* method was motivated by the need for an algorithm that could guarantee convergence to the global minimum of nonlinear optimization problems with twice-differentiable functions. The application of this algorithm to the minimization of potential energy functions was first introduced for microclusters [16]. The algorithm has also been shown to be successful for isolated [5,15], as well as solvated peptide systems [13].

Global Minimization Using αBB

The αBB global optimization algorithm effectively brackets the global minimum solution by developing converging sequences of lower and upper bounds. These bounds are refined by iteratively partitioning the initial domain. Upper bounds on the global minimum are obtained by local minimizations of the original energy function, E. Lower bounds belong to the set of solutions of the convex lower bounding functions, which are constructed by augmenting E with the addition of separable quadratic terms. By using ϕ_i^L, ψ_i^L, ω_i^L, $\chi_i^{k,L}$, $\theta_j^{N,L}$, $\theta_j^{C,L}$ and ϕ_i^U, ψ_i^U, ω_i^U, $\chi_i^{k,U}$, $\theta_j^{N,U}$, $\theta_j^{C,U}$ to refer to lower and upper bounds on the corresponding dihedral angles, the lower bounding function, L, of the energy hypersurface can be expressed in the following manner:

$$\begin{aligned} L = E &+ \sum_{i=1}^{N_{\text{RES}}} \alpha_{\phi,i} \left(\phi_i^L - \phi_i\right)\left(\phi_i^U - \phi_i\right) \\ &+ \sum_{i=1}^{N_{\text{RES}}} \alpha_{\psi,i} \left(\psi_i^L - \psi_i\right)\left(\psi_i^U - \psi_i\right) \\ &+ \sum_{i=1}^{N_{\text{RES}}} \alpha_{\omega,i} \left(\omega_i^L - \omega_i\right)\left(\omega_i^U - \omega_i\right) \\ &+ \sum_{i=1}^{N_{\text{RES}}} \sum_{k=1}^{K^i} \alpha_{\chi,i,k} \left(\chi_i^{k,L} - \chi_i^k\right)\left(\chi_i^{k,U} - \chi_i^k\right) \\ &+ \sum_{j=1}^{J^N} \alpha_{j,\theta^N} \left(\theta_j^{N,L} - \theta_j^N\right)\left(\theta_j^{N,U} - \theta_j^N\right) \\ &+ \sum_{j=1}^{J^C} \alpha_{j,\theta^C} \left(\theta_j^{C,L} - \theta_j^C\right)\left(\theta_j^{C,U} - \theta_j^C\right). \end{aligned}$$

The α represent nonnegative parameters which must be greater or equal to the negative one-half of the minimum eigenvalue of the Hessian of E over the defined domain. The overall effect of these terms is to overpower the nonconvexities of the original nonconvex terms by adding the value of 2 α to the eigenvalues of the Hessian of E. The convex lower bounding functions, L, possess a number of important properties which guarantee global convergence [18]:

i) L is a valid underestimator of E;
ii) L matches E at all corner points of the current box constraints;
iii) L is convex in the current box constraints;
iv) the maximum separation between L and E is bounded. This property ensures that feasibility and convergence tolerances can be reached for a finite size partition element;
v) the underestimators L constructed over supersets of the current set are always less tight than the underestimator constructed over the current box constraints for every point within the current box constraints.

Once solutions for the upper and lower bounding problems have been established, the next step is to modify the problem for the next iteration. This is accomplished by successively partitioning the initial domain into smaller subdomains. One obvious strategy is to subdivide the original hyper-rectangle by bisecting the longest dimension. In order to ensure nondecreasing lower bounds, the hyper-rectangle to be bisected is chosen by selecting the region which contains the infimum of the minima of lower bounds. A nonincreasing sequence for the upper bound is found by solving the nonconvex problem locally and selecting it to be the minimum over all the previously recorded upper bounds. If the single minimum of L for any hyper-rectangle is greater than the current upper bound, this hyper-rectangle can be discarded because the global minimum cannot be within this subdomain (fathoming step).

The computational requirement of the αBB algorithm depends on the number of variables (global) on which branching occurs. Therefore, these global variables need to be chosen carefully. Qualitatively, the branching variables should correspond to those variables which substantially influence the nonconvexity of the surface and the location of the global minimum. In terms of the protein folding problem, it is generally accepted that the backbone dihedral angles (ϕ and ψ) are the most influential variables. Therefore, in larger problems, the global variable set should include only the set of ϕ and ψ variables. In this case, the dihedral angles associated with the peptide bond (ω) and the side chains (χ) are treated as local variables.

Algorithmic Description

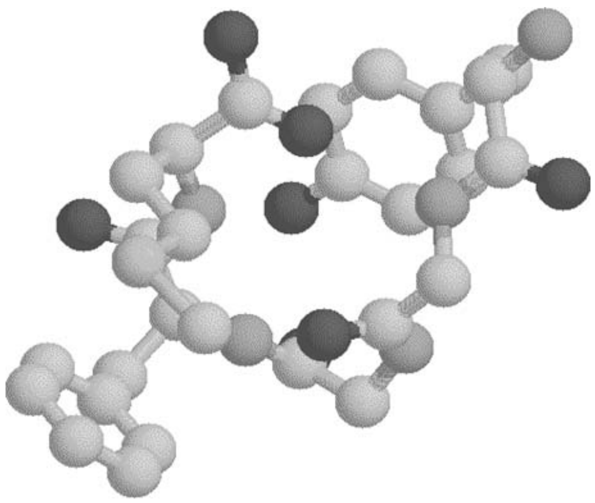
The basic steps of the algorithm are as follows:

1) The initial best upper bound is set to an arbitrarily large value. The original domain is partitioned along one of the global variable dimensions.
2) A convex function L is constructed in each hyper-rectangle and minimized using a local nonlinear solver, with function calls to potential and solvation models. If a solution is greater than the best upper bound the entire subregion can be fathomed, otherwise the solution is stored.
3) The local minima for L are used as initial starting points for local minimizations of the upper bounding function E in each hyper-rectangle. In solving the upper bounding problems, all variable bounds are expanded to $(-\pi, \pi)$ domain. These solutions are upper bounds on the global minimum solution in each hyper-rectangle.
4) The current best upper bound is updated to be the minimum of those thus far stored. If a new upper bound (from step 3) is selected, a separate module is called to ensure that the absolute value of each gradient in the objective function gradient vector is below a specified tolerance (kcal/mol/deg). The second derivative matrix is also evaluated to verify that the upper bound solution is a local minimum.
5) The hyper-rectangle with the current minimum value for L is selected and partitioned along one of the global variables.
6) If the best upper and lower bounds are within an ϵ tolerance the program will terminate, otherwise it will return to Step 2.

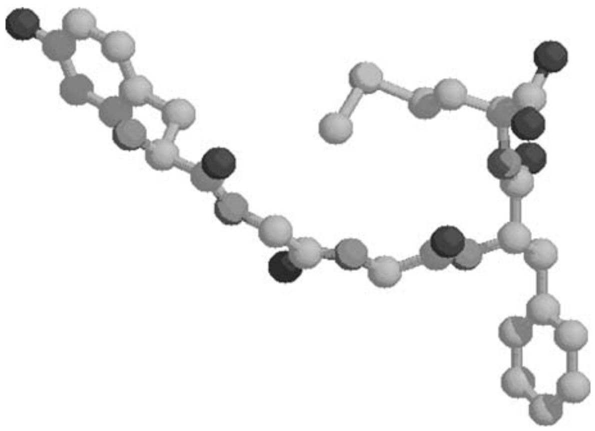
A novel approach has also been proposed for the initialization of the αBB algorithm [5]. Specifically, an analysis of 98 proteins from the Brookhaven X-ray data bank was used to develop dihedral angle distributions in the form of histograms from $-\pi$ to π for each dihedral angle of each of the naturally occurring amino acids. Using this information, a set of reduced domains can be defined for every dihedral angle of every residue in the peptide sequence. Overall initialization domains correspond to the Cartesian products of all the sub-domains of individual residues in the protein. This approach maintains the guarantee of global optimality over the considered search space of the reduced domains, and is deterministic in those subdomains that possess convex underestimators. In addition, all variable bounds are expanded to the $[-\pi, \pi]$ when solving the upper bounding problem. Therefore, although the initial point of an upper bounding minimization is restricted to the search space of the corresponding lower bounding problem, the solution may lie outside the original subdomain.

Example 1 Met-enkephalin (H-Tyr-Gly-Gly-Phe-Met-OH) is an endogenous opioid pentapeptide found in the human brain, pituitary, and peripheral tissues. Its biological function involves a large variety of physiological processes, most notably the endogenous response to pain. The peptide consists of 24 dihedral angles and a total of 75 atoms, and has played the role of a benchmark molecular conformation problem. The energy hypersurface is extremely complex with the number of local minima estimated on the order of 10^{11}. The unsolvated global minimum energy conformation, which is efficiently located using the αBB algorithm, has been shown to exhibit a type II' β-bend along the N-C' peptidic bond of Gly^3 and Phe^4 [5], as shown in Fig. 4.

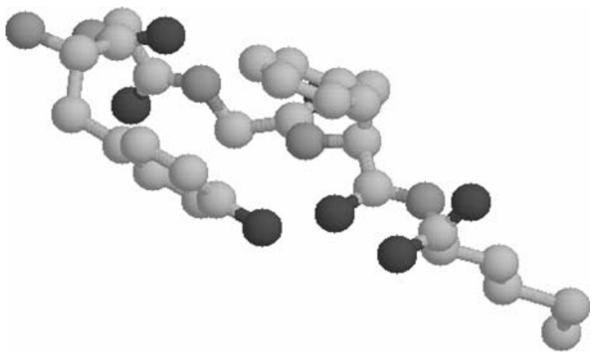
The algorithm has also successfully predicted global minimum energy structures of met-enkephalin using both solvent-accessible surface area (JRF) and volume of hydration (RRIGS) models [13]. In both cases, extended structures were identified, which qualitatively



Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach, Figure 4
Global minimum energy structure of unsolvated met-enkephalin



Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach, Figure 5
Global minimum energy structure of met-enkephalin using area based hydration



Multiple Minima Problem in Protein Folding: αBB Global Optimization Approach, Figure 6
Global minimum energy structure of met-enkephalin using volume based hydration

agrees with experimental results. However, differences in the role of nonbonded energies and the side chain conformations have been identified. The global minimum energy conformations of the surface area and volume of hydration models are shown in Fig. 5 and Fig. 6, respectively.

See also

- ► Adaptive Simulated Annealing and its Application to Protein Folding
- ► Genetic Algorithms
- ► Global Optimization in Lennard–Jones and Morse Clusters
- ► Global Optimization in Protein Folding
- ► Molecular Structure Determination: Convex Global Underestimation
- ► Monte-Carlo Simulated Annealing in Protein Folding
- ► Packet Annealing
- ► Phase Problem in X-ray Crystallography: Shake and Bake Approach
- ► Protein Folding: Generalized-ensemble Algorithms
- ► Simulated Annealing
- ► Simulated Annealing Methods in Protein Folding

References

1. Adjiman CS, Androulakis IP, Floudas CA (1998) A global optimization method, αBB, for general twice-differentiable NLPs-II. Implementation and computational results. Comput Chem Eng 22:1159–1179
2. Adjiman CS, Androulakis IP, Maranas CD, Floudas CA (1996) A global optimization method, αBB, for process design. Comput Chem Eng 20:S419–S424
3. Adjiman CS, Dallwig S, Floudas CA, Neumaier A (1998) A global optimization method, αBB, for general twice-differentiable NLPs-I. Theoretical advances. Comput Chem Eng 22:1137–1158
4. Allinger NL, Yuh YH, Lii J-H (1989) Molecular mechanics. The MM3 force field for hydrocarbons. J Amer Chem Soc 111:8551–8566
5. Androulakis IP, Maranas CD, Floudas CA (1997) Global minimum potential energy conformations of oligopeptides. J Global Optim 11:1–34

6. Augspurger JD, Scheraga HA (1996) An efficient, differentiable hydration potential for peptides and proteins. J Comput Chem 17:1549–1558
7. Brooks BR, Bruccoleri RE, Olafson BD, States DJ, Swaminathan S, Karplus M (1983) CHARMM: A program for macromolecular energy minimization and dynamics calculations. J Comput Chem 4:187–217
8. Connolly ML (1983) Analytical molecular surface calculation. J Appl Crystallogr 16:548–558
9. Eisenhaber F, Argos P (1993) Improved strategy in analytic surface calculation for molecular systems: Handling of singularities and computational efficiency. J Comput Chem 14:1272–1280
10. Floudas CA, Klepeis JL, Pardalos PM (1998) Global optimization approaches in protein folding and peptide docking. In: DIMACS, vol 47. Amer Math Soc, Providence, pp 141–171
11. van Gunsteren WF, Berendsen HJC (1987) GROMOS. Groningen Mol Sim
12. Kang YK, Némethy G, Scheraga HA (1987) Free energies of hydration of solute molecules 1. Improvement of hydration shell model by exact computations of overlapping volumes. J Phys Chem 91:4105–4109
13. Klepeis JL, Androulakis IP, Ierapetritou MG, Floudas CA (1998) Predicting solvated peptide conformations via global minimization of energetic atom-to-atom interactions. Comput Chem Eng 22:765–788
14. Kollman PA (1993) Free energy calculations: Applications to chemical and biochemical phenomena. Chem Rev 93:2395–2417
15. Maranas CD, Androulakis IP, Floudas CA (1996) A deterministic global optimization approach for the protein folding problem. In: DIMACS, vol 23. Amer Math Soc, Providence, pp 133–150
16. Maranas CD, Floudas CA (1992) A global optimization approach for Lennard–Jones microclusters. J Chem Phys 97:7667–7677
17. Maranas CD, Floudas CA (1994) A deterministic global optimization approach for molecular structure determination. J Chem Phys 100:1247–1261
18. Maranas CD, Floudas CA (1994) Global minimum potential energy conformations of small molecules. J Global Optim 4:135–170
19. Némethy G, Gibson KD, Palmer KA, Yoon CN, Paterlini G, Zagari A, Rumsey S, Scheraga HA (1992) Energy parameters in polypeptides 10. Improved geometrical parameters and nonbonded interactions for use in the ECEPP/3 algorithm with application to proline containing peptides. J Phys Chem 96:6472–6484
20. Neumaier A (1997) Molecular modeling of proteins and mathematical prediction of protein structure. SIAM Rev 39:407–460
21. Oobatake M, Némethy G, Scheraga HA (1987) Accessible surface areas as a measure of the thermodynamic parameters of hydration of peptides. Proc Nat Acad Sci USA 84:3086–3090
22. Perrot G, Cheng B, Gibson KD, Vila J, Palmer KA, Nayeem A, Maigret B, Scheraga HA (1992) MSEED: A program for the rapid analytical determination of accessible surface areas and their derivatives. J Comput Chem 13:1–11
23. Vila J, Williams RL, Vásquez M, Scheraga HA (1991) Empirical solvation models can be used to differentiate native from near-native conformations of bovine pancreatic trypsin inhibitor. PROTEINS: Struct Funct Genet 10:199–218
24. Weiner SJ, Kollman PA, Case DA, Singh UC, Ghio C, Alagona G, Profeta S, Weiner P (1984) A new force field for molecular mechanical simulation of nucleic acids and proteins. J Amer Chem Soc 106:765–784
25. Wesson L, Eisenberg D (1992) Atomic solvation parameters applied to molecular dynamics of proteins in solution. Protein Sci 1:227–235

Multiple Objective Dynamic Programming

MICHAEL M. KOSTREVA, LAURA C. LANCASTER
Department Math. Sci., Clemson University, Clemson, USA

MSC2000: 90C39, 90C31

Article Outline

Keywords

Dynamic programming; Multiple objective programming; Efficient set

Dynamic programming has been an area of active research since its introduction by R. Bellman [1]. More recently, with the recognition that many applied optimization problems require more than one objective, the study of multicriteria optimization has become a growing area of research. Included in this area of multicriteria optimization is the study of multiple objective dynamic programming (*MODP*). MODP was first used to replace multiple objective linear programming (MOLP) where it was not applicable, such as in problems with discrete variables. Many of the techniques used are extensions of classical dynamic programming. The following is a discussion of some of the research that has been developed in the area of MODP.

Using multiple objective dynamic programming to find the 'shortest' path through a network with constant costs is one of the more straightforward uses of MODP. Work has been done on both forward and backward MODP in this area. First, we consider a general network containing a set of nodes $N = \{1, \dots, n\}$ and a set of arcs $A = \{(i_0, i_1), (i_2, i_3), (i_4, i_5), \dots\} \subset N \times N$ which indicates connections between nodes. Each arc (i, j) has an associated cost vector, $c_{ij} = (c_{ij1}, \dots, c_{ijm}) \subset \mathbf{R}^m$. A path from node i_0 to i_p is the sequence of arcs $P = \{(i_0, i_1), \dots, (i_{p-1}, i_p)\}$ where the first node of each arc is the same as the terminal node of the preceding arc and each node in the path is unique. Let Π_i be the set of all paths from node 1 to node i. The cost to traverse a path p in Π_i is $[c(p)] = \sum_{(i,j) \in p} [c_{ij}]$. A path in Π_i is *nondominated* if there is no other path p^* in Π_i with $[c(p^*)]_r \leq [c(p)]_r$ for $r = 1, \dots, m$ and $[c(p^*)]r < [c(p)]_r$ for some $r \in \{1, \dots, m\}$.

```
0  $k = 1$.
1  Evaluate $S_i^k$ for all nodes using $S_i^k = \{c_{ij} + S_i^{k-1}\}$.
2  If $k < N$, set $k = k + 1$ and return to step 1; otherwise:
3  For each nondominated solution at each node determined in step 1 and for each $r$, $r = 1, \dots, m$, define $T^r$ as $T^r = \min_{i_N, \dots, i_0} \sum_n c^r_{i_n} j_{n-1}$, where $i_n$ is the originated node at stage $n$ and $I_n$ is the set of nodes that can be reached from node $n$.
4  Given weights $W^m \in \mathbf{R}^m_+$, compute the MIN-SUM as
   $\min \left[ \sum_{r=1}^{m} \left\{ W^r \frac{\sum_{k=1}^{N} c_{ij} - T^r}{T^r} \right\} \right]$.
```

H.G. Daellenbach and C.A. DeKluyver [5] gave one of the earliest algorithms for backward MODP with constant costs, which finds nondominated paths from all nodes to the destination node. Their method is basically an extension of the principle of optimality to a multicriteria context. They state a *principle of Pareto optimality of MODP*: 'A nondominated policy has the property that regardless of how the process entered a given state, the remaining decisions must belong to a nondominated subpolicy.' Let S_i^k be the nondominated vector of objective values for a node i, exactly k links from its destination, t. Then the algorithm is given above.

The resulting S_i^k vectors give nondominated solutions for the network, but maybe not all of them. They solve an example in which the weights are not specified.

A few years later, R. Hartley [6] proposed a similar algorithm that also uses backward MODP to find all Pareto paths from all nodes in the network to a specified node. The algorithm is as follows:

Let $V_0(i) = \{\infty, \dots, \infty\}$ for $k = 0, 1, \dots,$ and let $V_k(t) = \{0, \dots, 0\}$.

$V_k(i) = \mathrm{eff}[\cup \{c_{ij} + V_{k-1}(j) : j \in \Gamma(i)\}]$ for $i \in N$ $(i \neq t)$ and $k = 1, 2, \dots,$ where $\Gamma(i)$ is the set of nodes such that $(i, j) \in A$. The 'eff' operator finds all nondominated vectors in the set. The associated paths must be handled separately.

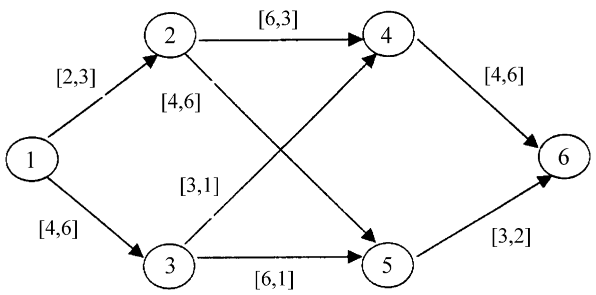
H.W. Corley and I.D. Moon [4] used forward MODP to find all nondominated paths from a specified node to all other nodes in a network with multiple constant costs. They assumed that the network contains no loops and that $c_{ij} \neq \{0, \dots, 0\}$ for any $(i, j) \in A$. Letting $G_i^{(k)}$ be the set of vector costs of all Pareto paths from node 1 to node i containing k or fewer arcs, the algorithm follows:

```
1  Set $c_{ii} = (0, \dots, 0)$, $i = 1, \dots, n$ and $c_{ij} = (\infty, \dots, \infty)$, $i \neq j$, if no arc exists from $i$ to $j$. Set $k = 1$ and let $G_i^{(1)} = \{c_{1i}\}$, $i = 1, \dots, n$.
2  For $i = 1, \dots, n$, set $G_i^{k+1} = \mathrm{Vmin} \cup_{j=1}^{n} \{c_{ij} + g_j^k : g_j^k \in G_j^{(k)}\}$.
3  If $G_i^{(k+1)} = G_i^{(k)}$, $i = 1, \dots, n$, stop, otherwise go to step 4.
4  If $k = n - 1$, stop. Else, $k = k + 1$ and go to step 2.
```

Vmin is an operation that computes the vector costs of all nondominated paths in a set of vector costs. An algorithm for Vmin is given in their paper.

Multiple Objective Dynamic Programming, Table 1

	$k = 1$	$k = 2$	$k = 3$
$G_1^{(k)}$	$\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\end{pmatrix}$
$G_2^{(k)}$	$\begin{pmatrix}2\\3\end{pmatrix}$	$\begin{pmatrix}2\\3\end{pmatrix}$	$\begin{pmatrix}2\\3\end{pmatrix}$
$G_3^{(k)}$	$\begin{pmatrix}4\\6\end{pmatrix}$	$\begin{pmatrix}4\\6\end{pmatrix}$	$\begin{pmatrix}4\\6\end{pmatrix}$
$G_4^{(k)}$	$\mathrm{V}\min\left\{\begin{pmatrix}8\\6\end{pmatrix},\begin{pmatrix}7\\7\end{pmatrix}\right\}$	$\left\{\begin{pmatrix}8\\6\end{pmatrix},\begin{pmatrix}7\\7\end{pmatrix}\right\}$	$\left\{\begin{pmatrix}8\\6\end{pmatrix},\begin{pmatrix}7\\7\end{pmatrix}\right\}$
$G_5^{(k)}$	$\mathrm{V}\min\left\{\begin{pmatrix}6\\9\end{pmatrix},\begin{pmatrix}10\\7\end{pmatrix}\right\}$	$\begin{pmatrix}6\\9\end{pmatrix},\begin{pmatrix}10\\7\end{pmatrix}$	$\begin{pmatrix}6\\9\end{pmatrix},\begin{pmatrix}10\\7\end{pmatrix}$
$G_6^{(k)}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}$	$\mathrm{V}\min\left\{\begin{pmatrix}12\\12\end{pmatrix},\begin{pmatrix}11\\13\end{pmatrix},\begin{pmatrix}9\\11\end{pmatrix},\begin{pmatrix}13\\9\end{pmatrix}\right\}$	$\begin{pmatrix}9\\11\end{pmatrix},\begin{pmatrix}13\\9\end{pmatrix}$



Multiple Objective Dynamic Programming, Figure 1

Table 1 gives the results of the algorithm. The resulting Pareto optimal paths from node 1 to node 6 are $\{(1, 2), (2, 5), (5, 6)\}$ and $\{(1, 3), (3, 5), (5, 6)\}$.

The following example uses the *Corley–Moon algorithm* to solve a dynamic routing problem for the network in Fig. 1.

Using multiple objective dynamic programming to find the shortest path through a network with time-dependent costs is considerably more complicated than MODP with constant costs. The monotonicity assumptions necessary for the principle of optimality in dynamic programming can easily be broken when dealing with time-dependent costs. Reaching a node later may be less costly than reaching it earlier. M.M. Kostreva and M.M. Wiecek [7] extended the work done by K.L. Cooke and E. Halsey [3] on dynamic programming with one time-dependent cost (travel time) to dynamic programming with multiple time-dependent costs. This method uses backward dynamic programming on a discrete time grid to find all nondominated paths from every node in the network to the destination node.

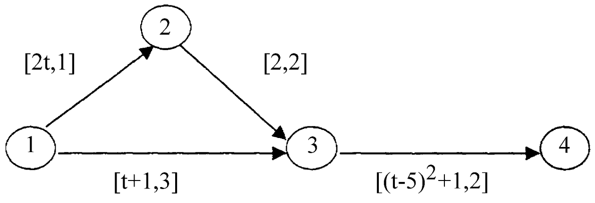
Assume the discrete time grid $S_T = \{t_0, \dots, t_0 + \mathrm{T}\}$, $t_0 > 0$ and the cost functions $[c_{ij}(t)]_k > 0$, $(i, j) \in A$, for all $t \in S_T$. T is the upper bound on total time to travel from any node in the network to the destination node, N_d. Also assume that $[c_{ij}(t)]_1$ is the time to travel from node i to node j when the arrival time at node i is time t. For all $i \in N \setminus N_d$ and all $t \in S_T$, define $\{[F_i(t)]\}$ as the set of nondominated vectors associated with the paths that leave node i at time t and reach node N_d and define $\{[F_i(t)(k)]\}$ as the set of nondominated vectors associated with the paths that leave node i at time t and reach node N_d in at most $k + 1$ links before time $t_0 + T$, where $k = 0, 1, \dots$. The following is the principle of optimality used for this algorithm: 'A nondominated path p, leaving node i at time $t \in S_T$ and reaching node N at or before time $t_0 + T$, has the property that for each node j lying on this path, a subpath p_1, that leaves node j at time $t_j \in S_T$, $t_j > t$, and arrives at node N at or before time $t_0 + T$, is nondominated.' The algorithm is as follows:

```
1  Find a time grid of discrete values $S_T = \{t_0, \dots, t_0 + T\}$, $t_0 > 0$ and compute $[c_{ij}(t)]$ for all $t \in S_T$ and all $(i, j) \in A$.
2  Modify $[c_{ij}(t)]$ for all $t \in S_T$ and all $(i, j) \in A$ as follows:
     $[c_{ij}(t)]' = \begin{cases} [c_{ij}(t)] & \text{if } t+[c_{ij}(t)]_l \leq t_0+T, \\ \infty & \text{if } t+[c_{ij}(t)]_l > t_0+T. \end{cases}$
3  Find the initial array $[\{[F_i(t)^{(0)}]\}]$, $i = 1, \dots, N$, for all $t \in S_T$, where $\{[F_{N_d}(t)^{(0)}]\} = \{0\}$, and $\{[F_i(t)^{(0)}]\} = [c_{iN_d}(t)]'$ for $i \in N \backslash N_d$.
4  Find the arrays $[\{[F_i(t)^{(k)}]\}]$, $i = 1, \dots N$, for all $t \in S_T$, for $k = 1, 2, \dots$ as follows:
     $\{[F_i(t)^{(k)}]\}$
     $= \mathrm{VMIN}\{[c_{ij}(t)]' + \{[F_j(t+[c_{ij}(t)]'_1)^{(k-1)}]\}\}$,
     $i \in N \backslash \{N_d\}$,
     $\{[F_i(t)^{(k)}]\} = \{0\}$.
5  The sequence of sets $\{[F_i(t_0)^{(k)}]\}$, $k = 1, 2, \dots$, converges to the set $\{[F_i(t_0)]\}$, the set of nondominated vectors associated with the paths that leave node $i$ at time $t_0$ and reach node $N_d$.
```

The following example uses Algorithm One [7] to solve a dynamic routing problem for the network in Fig. 2. A grid of discrete values of time $S_{19} = \{1, 2, \dots, 20\}$ for $t_0 = 1$ is established.

Table 2 shows the initial array and the two subsequent arrays. So, the set $\{\mathrm{Eff}(E_I(t_0))\}$ of all nondominated paths that leave node 1, 2, and 3 at time $t_0 = 1$ are $\{(1, 2), (2, 3), (3, 4)\}$, $\{(2, 3), (3, 4)\}$, and $\{(3, 4)\}$.

Kostreva and Wiecek [7] also developed an algorithm which uses forward dynamic programming to



Multiple Objective Dynamic Programming, Figure 2

find all nondominated paths from an origin node to every other node in the network without using a time grid. Thus, assume t is a continuous variable, $t \geq 0$, and $[c_{ij}(t)]_1 > 0$. An assumption must be made about the cost functions so that the principle of optimality will hold for these networks: For any arc $(i, j) \in A$ and all $t_1, t_2 \geq 0$, if $t_1 \leq t_2$, then:

a) $t_1 + [c_{ij}(t_1)]_1 \leq t_2 + [c_{ij}(t_2)]$ 1, and
b) $[c_{ij}(t_1)]_r \leq [c_{ij}(t_2)]_r$ for all $r \in \{2, \dots, m\}$. Assuming the cost functions are monotone increasing with respect to time satisfies this assumption.

```
1  Find the initial vector $\{[G_j^{(0)}]\}$, $j = 1, \dots, N$, where $\{[G_1^{(0)}]\} = \{0\}$ and $\{[G_j^{(0)}]\} = [c_{1j}(0)]$, $j = 2, \dots, N$.
2  Calculate the vectors $\{[G_j^{(k)}]\}$, $j = 1, \dots, N$, for $k = 1, 2, \dots$, as follows:
     $\{[G_j^l(t_l)^{(k)}],\ l =, \dots, N_j\}$
     $= \mathrm{VMIN}\{[G_i^n(t^n)^{(k-1)}] + [c_{ij}(t^n)]$,
     $n = 1, \dots, N_i\}$,
     $j = 2, \dots, N$,
     $\{[G_1^l(t^l)^{(k)}], l = 1\} = \{0\}$.
3  $\{[G_j^{(k)}]\}$, $k = 1, 2, \dots$, converges to $\{[G_j]\}$, the set of vector costs of all nondominated paths which leave the origin node at time $t = 0$ and lead to node $j$.
```

Assume that node 1 is the origin node. For nodes $j = 2, \dots, N$, let $[G_j^u(t^u)^{(k)}]$ be the vector cost of the nondominated path u which is of at most k links leaving the origin node at time $t = 0$ and leading to node j, where t^u is the arrival time of this path at node j. Also, let $\{[G_j^{(k)}]\}$ be the set of vector costs of all nondominated paths which are of at most k links leaving the origin node at time $t = 0$ and leading to node j, where N_j is the number of nondominated paths. Let $\{[G_j]\}$ be the set of vector costs of all nondominated paths which leave the origin node at time $t = 0$ and lead to node j. The algorithm is as listed above.

Another way to get around the monotonicity assumption of dynamic programming is to use generalized dynamic programming techniques. See [2] for

Multiple Objective Dynamic Programming, Table 2
Sequence of arrays

Time	$[\{[F_I(t)^{(0)}]\}]$	$[\{[F_I(t)^{(1)}]\}]$	$[\{[F_I(t)^{(2)}]\}]$
1	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}17\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}7\\5\end{pmatrix}\begin{pmatrix}7\\4\end{pmatrix}\begin{pmatrix}17\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}5\\5\end{pmatrix}\begin{pmatrix}7\\4\end{pmatrix}\begin{pmatrix}17\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$
2	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}10\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}4\\5\end{pmatrix}\begin{pmatrix}4\\4\end{pmatrix}\begin{pmatrix}10\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}4\\5\end{pmatrix}\begin{pmatrix}4\\4\end{pmatrix}\begin{pmatrix}10\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$
3	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}5\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}9\\5\end{pmatrix}\begin{pmatrix}3\\4\end{pmatrix}\begin{pmatrix}5\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}9\\5\end{pmatrix}\begin{pmatrix}3\\4\end{pmatrix}\begin{pmatrix}5\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$
4	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}2\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}4\\4\end{pmatrix}\begin{pmatrix}2\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}4\\4\end{pmatrix}\begin{pmatrix}2\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$
5	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}1\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}7\\4\end{pmatrix}\begin{pmatrix}1\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}7\\4\end{pmatrix}\begin{pmatrix}1\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$
6	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}2\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}12\\4\end{pmatrix}\begin{pmatrix}2\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}12\\4\end{pmatrix}\begin{pmatrix}2\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$
7	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}5\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}5\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}5\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$
8	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}10\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}10\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}10\\2\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$
9	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$
...	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}\infty\\\infty\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$

a way to use generalized DP with a multicriteria preference function. Basically, generalized DP uses a weaker principle of optimality than Bellman's famous version [1]. Generalized DP finds partial solutions that may lead to optimal solutions even though locally they are not optimal solutions according to the preference function.

In [2] generalized DP is applied to the multicriteria best path problem. Assuming node 1 to be the origin and node N to be the destination, let Π be the set of all paths in the network. Let

$$P(j) = \{p \in \Pi\colon\ i_1 = 1,\ i_n = j\}$$

be the set of all paths from the origin to node j. Let

$$X(j) = \{p \in \Pi\colon\ i_1 = j,\ i_n = N\}$$

be the set of all paths from node j to the destination node. The vector cost along each arc is called an *arc length vector*, $l_{ij} = (l^1_{ij}, \dots, l^m_{ij}) \in \mathbf{R}^m$. A path length function $z\colon \Pi \to \mathbf{R}^m$ assigns a path length vector to every path $p \in \Pi$ where $\circ$ is a binary operator on $\mathbf{R}^m$:

$$z(p) = l_{1,2} \circ \cdots \circ l_{i_{n-1},i_n}.$$

Thus, each different objective can have a different binary operation. For example, distance would have an additive binary operator and probabilities would have a multiplicative binary operator. Let

$$Z(j) = \{z(p)\colon\; p \in P(j)\}$$

be the set of all length vectors of all paths from the origin to node j. A multicriteria preference function u: $\mathbf{R}^m \to \mathbf{R}$ is defined on the set of path length vectors. The objective is to maximize this preference function. The monotonicity assumption says that for all $z, z' \in Z(j), u(z) \leq u(z') \Rightarrow u(z \circ l_{jk}) \leq u(z' \circ l_{jk})$ for all j, $k \in S$ such that $(j, k) \in T$. Unfortunately, with multiobjective problems this assumption is easily violated. Generalized DP tries to get around this monotonicity assumption by having local preference relations defined as $\rho_j \subseteq Z(j) \times Z(j)$: for $z, z' \in Z(j)$, where $z\rho_j z'$ implies that any subpath from the origin to node j whose length is z cannot be used in a path to produce a better overall path from the origin to the destination node than using the subpath from the origin to node j whose length is z'. So, subpath length vector z' is more locally preferred even though subpath length vector z may be globally preferred, $u(z') \leq u(z)$. So, for $z, z' \in Z(j)$, $z\rho_j z'$ if and only if $\exists p' \in X(j)$ such that $u(z \circ z(p)) \leq u(z' \circ z(p'))$ for all $p \in X(j)$. These local preference relations are used to form the *weak principle of optimality*. An optimal path must be composed of subpaths that can be part of an optimal path.

Unfortunately, in order to get these preference relations one would have to complete all paths from every node in the network. Since this is too computationally intense, the preference relations are relaxed to the refining local preference relations $\prec_j$ where $z \prec_j z'$ implies $z\ \rho_j\ z'$. Using $\prec_j$ avoids having to find the entire relation ρ_j. Using this relation means that a larger set of maximal path length vectors will be kept by using ρ_j than if ρ_j were used. A maximal path length vector is a vector where there does not exist another vector at that state that is strictly more preferred. Let

$$\operatorname{maxl}(X, \rho) = \left\{x \in X\colon\; \exists x' \in X\colon x\rho x' \text{ and } x'\rho x\right\}.$$

The following are the equations of generalized DP:

$$\begin{aligned} &f(1) = \{z_1\},\\ &f(j) = \operatorname{maxl}\left(\cup_{(i,j)\in A}(f(i) \circ l_{ij}) \prec_j\right)\\ &\text{for } j = 2, \ldots, N, \end{aligned}$$

where $\{f(i) \circ l_{ij}\} = \{z \circ l_{ij}\colon z \in f(i)\}$.

When the monotonicity assumptions are satisfied, the $\prec_j$ relation can be replaced with the multicriteria preference function, u, thus reducing to the conventional DP problem. However, when the monotonicity assumption does not hold the $\prec_j$ relation must be defined by trying exploit any special structures of each individual problem. Also, using dynamic programming to find the entire Pareto optimal set can be seen as another special case of generalized DP where $z_k \geq z_k'$ for all $k = 1, \ldots, m \Rightarrow z \prec_j z'$ (assuming minimization of each criteria).

The subject of multiple objective dynamic programming has developed into a viable body of knowledge capable of providing solutions to applied problems in which trade-offs among objectives is important. Among the multiple objective techniques, it is distinctive in its ability to provide the entire Pareto optimal set. To gain such an advantage, one must be willing to perform computationally intensive operations on large sets of vectors.

See also

- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Neuro-dynamic Programming

References

1. Bellman RE (1957) Dynamic programming. Princeton University Press, Princeton

2. Carraway RL, Morin TL, Moskowitz H (1990) Generalized dynamic programming for multicriteria optimization. Europ J Oper Res 44:95–104
3. Cooke KL, Halsey E (1966) The shortest route through a network with time-dependent internodal transit times. J Math Anal Appl 14:493–498
4. Corley HW, Moon ID (1985) Shortest paths in networks with vector weights. J Optim Th Appl 46:79–86
5. Daellenbach HG, DeKluyver CA (1980) Note on multiple objective dynamic programming. J Oper Res Soc 31:591–594
6. Hartley R (1984) Vector optimal routing by dynamic programming. In: Serafini P (ed) Mathematics of Multiobjective Optimization, pp 215–224
7. Kostreva MM, Wiecek MM (1993) Time dependency in multiple objective dynamic programming. J Math Anal Appl 173:289–307

Multiple Objective Programming Support

Pekka Korhonen[1,2]
[1] Internat. Institute Applied Systems Analysis, Laxenburg, Austria
[2] Helsinki School Economics and Business Adm., Helsinki, Finland

MSC2000: 90C29

Article Outline

Keywords

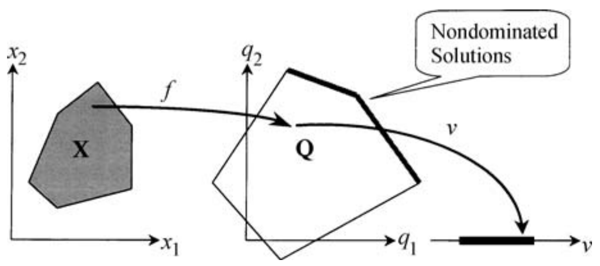
Multiple criteria decision making; Multiple objective programming; Multiple objective programming support; Scalarizing function; Value function

This article gives a brief introduction into multiple objective programming support. We will overview basic concepts, formulations, and principles of solving multiple objective programming problems. To solve those problems requires the intervention of a decision-maker. That's why behavioral assumptions play an important role in multiple objective programming. Which assumptions are made affects which kind of support is given to a decision maker. We will demonstrate how a free search type approach can be used to solve multiple objective programming problems.

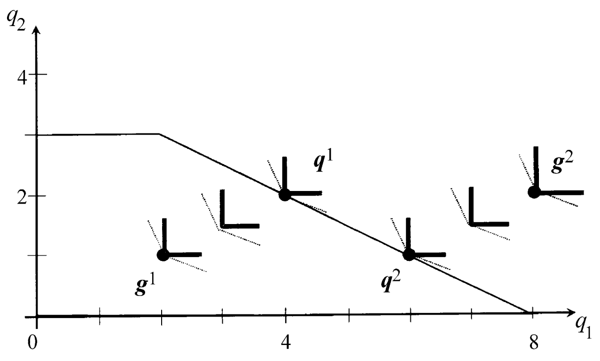
Introduction

Before we can consider the concept of *multiple objective programming support* (MOPS), we have to first explain the concept of *multiple criteria decision making* (MCDM). Even if there is a variation of different definitions, most researchers working in the field might accept the following general definition: Multiple Criteria Decision Making (MCDM) refers to the solving of decision and planning problems involving multiple (generally conflicting) criteria. 'Solving' means that a decision-maker (DM) will choose one 'reasonable' alternative from among a set of available ones. It is also meaningful to define that the choice is irrevocable. For an MCDM problem it is typical that no unique solution for the problem exists. Therefore to find a solution for MCDM problems requires the intervention of a decision-maker (DM). In MCDM, the word 'reasonable' is replaced by the words 'efficient/nondominated'. They will be defined later on.

Actually the above definition is a strongly simplified description of the whole (multiple criteria) decision making process. In practice, MCDM problems are not often so well-structured, that they can be considered just as a choice problem. Before a decision problem is ready to be 'solved', the following questions require a lot of preliminary work: How to structure the problem? How to find essential criteria? How to handle uncertainty? These questions are by no means outside the interest area of MCDM-researchers. The outranking method by B. Roy [17] and the AHP (the analytic hierarchy process) developed by T.L. Saaty [18] are examples of the MCDM-methods, in which a lot of effort is devoted to problem structuring. Both methods are well known and widely used in practice. In both methods,



Multiple Objective Programming Support, Figure 1
A variable, criterion, and value space



Multiple Objective Programming Support, Figure 2
Illustrating the projection of a feasible and an infeasible aspiration level point onto the nondominated surface

the presence of multiple criteria is an essential feature, but the structuring of a problem is an even more important part of the solution process.

When the term 'support' is used in connection with MCDM, we may adopt a broad perspective and refer with the term to all research associated with the relationship between the problem and the decision-maker. In this article we take a narrower perspective and focus on a very essential supporting problem in multiple criteria decision making: How to assist a DM to find the 'best' solution from among a set of available 'reasonable' alternatives, when the alternatives are evaluated by using several criteria? Available alternatives are assumed to be defined explicitly or implicitly by means of a mathematical model. The term *multiple objective programming* is usually used to refer to dealing with this kind of model.

The following considerations are general in the sense that usually it is not necessary to specify how the alternatives are defined. It is enough to assume that they belong to set Q. However, in Fig. 1 and Fig. 2 and the numerical example we consider a multiple objective linear programming model in which all constraints and objectives are defined using linear functions.

The article consists of seven sections. In Sect. "A Multiple Objective Programming Problem", we give a brief introduction to some foundations of multiple objective programming. How to generate potential 'reasonable' solutions for a DM's evaluation is considered in Sect. "Generating Nondominated Solutions", and in Sect. "Solving Multiple Objective Problems", we will review general principles to solve a multiple objective programming problem. In Sect. "Example of a Decision Support System: VIG", a multiple criteria decision support system VIG is introduced, and a numerical example is solved in Sect. "Numerical Illustrations". Concluding remarks are given in Sect. "Conclusion".

A Multiple Objective Programming Problem

A multiple objective programming (MOP) problem in a so-called *criterion space* can be defined as follows:

$$\begin{cases} \text{`max'} & q \\ \text{s.t.} & q \in Q , \end{cases} \tag{1}$$

where set $Q \subset \mathbf{R}^k$ is a so-called *feasible region* in a k-dimensional criterion space $\mathbf{R}^k$. The set Q is of special interest. Most considerations in multiple objective programming are made in a criterion space.

Set Q may be convex/nonconvex, bounded/unbounded, precisely known or unknown, consist of finite or infinite number of alternatives, etc. When Q consists of a finite number of elements which are explicitly known in the beginning of the solution process, we have an important class of problems which may be called e. g. (multiple criteria) *evaluation problems*. Sometimes those problems are referred to as *discrete multiple criteria problems* or *selection problems* (for a survey see for example [16]).

When the number of alternatives in Q is infinite and not countable, the alternatives are usually defined using a mathematical model formulation, and the problem is called continuous. In this case we say that the alternatives are only implicitly known. This kind of problem is referred as a *multiple criteria design problem* (the terms 'evalution' and 'design' are adopted from A. Arbel) or a *continuous multiple criteria problem*. In this case, the set Q is not specified directly, but by means of decision variables as usually done in single optimization problems:

$$\begin{cases} \max & q = f(x) = (f_1(x), \dots, f_k(x)) \\ \text{s.t.} & x \in X, \end{cases} \tag{2}$$

where $X \subset \mathbf{R}^n$ is a *feasible set* and $f : \mathbf{R}^n \to \mathbf{R}^k$. The space $\mathbf{R}^n$ is called a *variable space* (see Fig. 1). The functions f_i, $i = 1, \dots, k$, are *objective functions*. The feasible region Q can now be written as $Q = \{q: q = f(x), x \in X\}$.

The MOP-problem has seldom a unique solution, i. e. an optimal solution that simultaneously maximizes all objectives. Conceptually the multiple objective mathematical programming problem may be regarded as a *value* (*utility*) *function* maximization program:

$$\begin{cases} \max & v(q) \\ \text{s.t.} & q \in Q, \end{cases} \tag{3}$$

where v is a real-valued function, which is strictly increasing in the criterion space and defined at least in the feasible region Q. It is mapping the feasible region into a one-dimensional value space (see Fig. 1). Function v specifies the DM's preference structure over the feasible region. However, the key assumption in multiple objective programming is that v is unknown. Generally, if the value function is estimated explicitly, the system is considered to be in the MAUT category, see for example [7], (MAUT stands for multiple attribute utility theory) and can then be solved without any interaction of the DM. Typically, MAUT-problems are not even classified under the MCDM-category. If the value function is implicit (assumed to exist but is otherwise unknown) or no assumption about the value function is made, the system is usually classified under MCDM [2] or MOP.

Solutions of the MOP-problems are all those alternatives which can be the solutions of some value function v: $Q \to \mathbf{R}$. Those solutions are called *efficient* or *nondominated* depending on the space where the alternatives are considered. The term nondominated is used in the criterion space and efficient in the variable space. (Some researchers use the term efficient to refer to efficient and nondominated solutions without making any difference.) Any choice from among the set of efficient (nondominated) solutions is an acceptable and 'reasonable' solution, unless we have no additional information about the DM's preference structure.

Nondominated solutions are defined as follows:

Definition 1 In (1), $q^* \in Q$ is *nondominated* if and only if there does not exist another $q \in Q$ such that $q \geq q^*$ and $q \neq q^*$.

Definition 2 In (1), $q^* \in Q$ is *weakly nondominated* if and only if there does not exist another $q \in Q$ such that $q > q^*$.

Correspondingly, efficient solutions are defined as follows:

Definition 3 In (2), $x^* \in X$ is *efficient* if and only if there does not exist another $x \in X$ such that $f(x) \geq f(x^*)$ and $f(x) \neq f(x^*)$.

Definition 4 In (2), $x^* \in X$ is *weakly efficient* if and only if there does not exist another $x \in X$ such that $f(x) > f(x^*)$.

The final ('best') solution $q \in Q$ of the problem (1) is called the *most preferred solution*. It is a solution preferred by the DM to all other solutions. At the conceptual level, we may think it is the solution maximizing an (unknown) value function in problem (3). How to find it? That is the problem we now proceed to consider.

Unfortunately, the above characterization of the most preferred solution is not very operational, because no system can enable the DM to simultaneously compare the final solution to all other solutions with an aim to check if it is really the most preferred or not. It is also as difficult to maximize a function we do not know. Some properties for a good system are, for example, that it makes the DM convinced that the final solution is the most preferred one, does not require too much time from the DM to find the final solution, to give reliable enough information about alternatives, etc. Even if it is impossible to say which system provides the best support for a DM for his multiple criteria problem, all proper systems have to be able to recognize, generate and operate with nondominated solutions. To generate nondominated solutions for the DM's evaluation is thus one key issue in multiple objective programming. In the next section, we will consider some principles.

Generating Nondominated Solutions

Despite many variations among different methods of generating nondominated solutions, the ultimate principle is the same in all methods: a single objective optimization problem is solved to generate a new solution or solutions. The objective function of this single objective problem may be called a *scalarizing function*, according to [25]. It typically has the original objectives and a set of parameters as its arguments. The form of the scalarizing function as well as what parameters are used depends on the assumptions made concerning the DM's preference structure and behavior.

Two classes of parameters are widely used in multiple objective optimization:

1) weighting coefficients for objective functions; and
2) reference/aspiration/reservation levels for objective function values.

Based on those parameters, there exist several ways to specify a scalarizing function. An important requirement is that this function completely characterizes the set of nondominated solutions:

> for each parameter value, all solution vectors are nondominated, and for each nondominated criterion vector, there is at least one parameter value, which produces that specific criterion vector as a solution

(see, for theoretical considerations, e. g. [26]).

A Linear Scalarizing Function

A classic method to generate nondominated solutions is to use the weighted-sums of objective functions, i. e. to use the following linear scalarizing function:

$$\max\left\{\lambda' f(x):\ x \in X\right\}. \tag{4}$$

If $\lambda > 0$, then the solution vector x of (4) is efficient, but if we allow that $\lambda \geq 0$, then the solution vector is weakly-efficient. (see, e. g. [21, p. 215; 221]). Using the parameter set $\Lambda = \{\lambda\colon \lambda > 0\}$ in the weighted-sums linear program we can completely characterize the efficient set provided the constraint set is convex. However, Λ is an open set, which causes difficulties in a mathematical optimization problem. If we use $\mathrm{cl}(\Lambda) = \{\lambda\colon \lambda \geq 0\}$ instead, the efficiency of x cannot be guaranteed anymore. It is surely weakly-efficient, and not necessarily efficient. When the weighted-sums are used to specify a scalarizing function in multiple objective linear program (MOLP) problems, the optimal solution corresponding to nonextreme points of X is never unique. The set of optimal solutions always consists of at least one extreme point, or the solution is unbounded. In early methods, a common feature was to operate with weight vectors $\lambda \in \mathbf{R}^k$, limiting considerations to efficient extreme points (see, e. g., [29]).

A Chebyshev-Type Scalarizing Function

Currently, most solution methods are based on the use of a so-called Chebyshev-type scalarizing function first proposed by A. Wierzbicki [25]. We will refer to this function by the term *achievement* (*scalarizing*) *function*. The achievement (scalarizing) function projects any given (feasible or infeasible) point $g \in \mathbf{R}^k$ onto the set of nondominated solutions. Point g is called a *reference point*, and its components represent the desired values of the objective functions. These values are called *aspiration levels*.

The simplest form of achievement function is:

$$s(g,q,w) = \max_{k\in K} \frac{g_k - q_k}{w_k}, \tag{5}$$

where $w > 0 \in \mathbf{R}^k$ is a (given) vector of weights, $g \in \mathbf{R}^k$, and $q \in Q = \{f(x)\colon x \in X\}$. By minimizing $s(g, q, w)$ subject to $q \in Q$, we find a weakly nondominated solution vector q^* (see, e. g. [25,26]). However, if the solution is unique for the problem, then q^* is nondominated. If $g \in \mathbf{R}^k$ is feasible, then $q^* \in Q$, $q^* \geq g$. To guarantee that only nondominated (instead of weakly nondominated) solutions will be generated, more complicated forms for the achievement function have to be used, for example:

$$s(g,q,w,\rho) = \max_{k\in K}\left[\frac{g_k - q_k}{w_k}\right] + \rho\sum_{i=1}^{k}(g_i - q_i), \tag{6}$$

where $\rho > 0$. In practice, we cannot operate with a definition 'any positive value'. We have to use a prespecified value for ρ. Another way is to use a lexicographic formulation [10].

The applying of the scalarizing function (6) is easy, because given $g \in \mathbf{R}^k$, the minimum of $s(g, v, w, \rho)$ is found by solving the following LP-problem:

$$\begin{cases} \min & \epsilon + \rho \sum_{i=1}^{k} (g_i - q_i) \\ \text{s.t.} & x \in X \\ & \epsilon \geq \frac{g_i - q_i}{w_i}, \quad i = 1, \dots, k. \end{cases} \tag{7}$$

Problem (7) can be further written as:

$$\begin{cases} \min & \epsilon + \rho \sum_{i=1}^{k} (g_i - q_i) \\ \text{s.t.} & x \in X \\ & q + \epsilon w - z = g \\ & z \geq 0. \end{cases} \tag{8}$$

To illustrate the use of the achievement scalarizing function, consider a two-criteria problem with a feasible region having four extreme points (0, 0), (0, 3), (2, 3), (8, 0), as shown in Fig. 2. In Fig. 2, the thick solid lines describe the indifference curves when $\rho = 0$ in the achievement scalarizing function. The thin dotted lines stand for the case $\rho > 0$. Note that the line from (2, 3) to (8, 0) is nondominated and the line from (0, 3) to (2, 3) (excluding the point (2, 3)) is weakly-nondominated, but dominated. Let us assume that the DM first specifies a feasible aspiration level point $g^1 = (2, 1)$. Using a weight vector $w = [2, 1]'$, the minimum value of the achievement scalarizing function (-1) is reached at a point $v^1 = (4, 2)$ (cf. Fig. 2). Correspondingly, if an aspiration level point is infeasible, say $g^2 = (8, 2)$, then the minimum of the achievement scalarizing function (+ 1) is reached at point $v^2 = (6, 1)$. When a feasible point dominates an aspiration level point, then the value of the achievement scalarizing function is always negative; otherwise it is nonnegative. It is zero, if an aspiration level point is weakly-nondominated.

Solving Multiple Objective Problems

Several dozen procedures and computer implementations have been developed from the 1970s onwards to address both *multiple criteria evaluation* and *design problems*. The multiple objective decision procedures always requires the intervention of a DM at some stage in the solution process. A popular way to involve the DM in the solution process is to use an interactive approach.

The specifics of these procedures vary, but they have several common characteristics. For example, at each iteration, a solution, or a set of solutions, is generated for a DM's examination. As a result of the examination, the DM inputs information in the form of trade-offs, pairwise comparisons, aspiration levels, etc. (see [20] for a more detailed discussion). The responses are used to generate a presumably, improved solution. The ultimate goal is to find the most preferred solution of the DM. Which search technique and termination rule is used is heavily dependent on the underlying assumptions postulated about the behavior of the DM and the way in which these assumptions are implemented. In MCDM-research there is a growing interest in the behavioral realism of such assumptions.

Based on the role that the value function (3) is supposed to play in the analysis, we can classify the assumptions into three categories:

1) Assume the existence of a value function v, and assess it explicitly.
2) Assume the existence of a stable value function v, but do not attempt to assess it explicitly. Make assumptions of the general functional form of the value function.
3) Do not assume the existence of a stable value function v, either explicitly, or implicitly.

The first assumption is adopted in multi-attribute utility or decision analysis (see, e. g. [7]). Interactive software implementing such approaches on personal computers exists.

The second assumption was a basic paradigm used in interactive multiple criteria approaches in the 1970s. A classical example is the GDF-method [3]. DM's responses to specific questions were used to guide the solution process towards an 'optimal' or 'most preferred' solution (in theory), assuming that the DM behaves according to some specific (but unknown) underlying value function (see for surveys, e. g. [5,20,21], and [24]). Interactive software that implements such systems for a computer have often been developed by the authors of the above procedures for experimental purposes.

The approaches based on the assumption on 'no stable value/utility function' typically operate with a DM's aspiration levels regarding the objectives on the feasible region. The aspiration levels are projected via minimizing so called achievement scalarizing functions (6) [23,25]. No specific behavioral assumptions e. g. transitivity are necessary.

In essence, this approach seeks to help the DM more or less freely to search the set of efficient solutions. Interactive software that implements such systems for a computer have been developed like ADBASE [22], DIDAS [14], VIG [8], and VIMDA [9]. For an excellent review of several interactive multiple criteria procedures, see [21]. Other well-known books that provides a deeper background and additional references especially in the field of multiple objective optimization include [1,4,5,6,19,27] and [28].

Multiple objective linear programming (MOLP) is the most commonly studied problem in *multiple criteria decision making* (MCDM). Most solution methods are developed for this problem.

Example of a Decision Support System: VIG

Today, many systems use aspiration level projections, where the projection is performed using Chebyshev-type achievement scalarizing functions as explained above. These functions can be controlled either by varying weights (keeping aspiration levels fixed) or by varying the aspiration levels (keeping weights fixed). Instead of aspiration levels, some algorithms asks the DM to specify the reservation levels for the criteria (see, e. g. [15]).

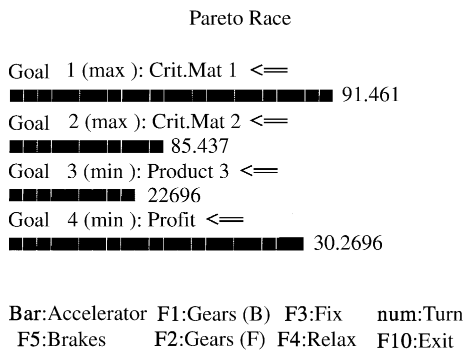
An achievement scalarizing function projects one aspiration (reservation) level point at a time onto the nondominated frontier. By parametrizing the function, it is possible to project the whole vector onto the nondominated frontier as originally proposed by [11]. The vector to be projected is called a *reference direction vector* and the method *reference direction method*, correspondingly. When a direction is projected onto the nondominated frontier, a curve traversing across the nondominated frontier is obtained. Then an interactive line search is performed along this curve. The idea enables the DM to make a continuous search on the nondominated frontier. The corresponding mathematical model is a simple modification from the original model (8) developed for projecting one point:

$$\begin{cases} \min & \epsilon + \rho \sum_{i=1}^{k} (g_i - q_i) \\ \text{s.t.} & x \in X \\ & q + \epsilon w - z = g + tr, \quad z \geq 0, \end{cases} \tag{9}$$

where t: $0 \to \infty$ and $r \in \mathbf{R}^k$ is a reference direction. In the original approach, a reference direction was specified as a vector starting from the current solution and passing through the aspiration levels. The DM was asked to give aspiration levels for criteria.

The original reference direction approach has been further developed into many directions. First, [12] improved upon the original procedure by making the specification of a reference direction dynamic. The dynamic version was called *Pareto race*. In Pareto race, the DM can freely move in any direction on the nondominated frontier he/she likes, and no restrictive assumptions concerning the DM's behavior are made. Furthermore, the objectives and constraints are presented in a uniform manner. Thus, their role can also be changed during the search process. The method and its implementation is called Pareto race. The whole software package consisting of Pareto race is called VIG.

In Pareto race, a reference direction r is determined by the system on the basis of preference information received from the DM. By pressing number keys corresponding to the ordinal numbers of the objectives, the DM expresses which objectives he/she would like to improve and how strongly. In this way he/she implicitly specifies a reference direction. Figure 3 shows the


Pareto Race

Goal 1 (max): Crit.Mat 1 <==
91.461
Goal 2 (max): Crit.Mat 2 <==
85.437
Goal 3 (min): Product 3 <==
22696
Goal 4 (min): Profit <==
30.2696

Bar:Accelerator F1:Gears (B) F3:Fix num:Turn
F5:Brakes F2:Gears (F) F4:Relax F10:Exit

Multiple Objective Programming Support, Figure 3
Example Pareto race screen

Pareto race interface for the search, embedded in the VIG software [8].

Thus Pareto race is a visual, dynamic, search procedure for exploring the nondominated frontier of a multiple objective linear programming problem. The user sees the objective function values on a display in numeric form and as bar graphs, as he/she travels along the nondominated frontier. The keyboard controls include an accelerator, gears, brakes, and a steering mechanism. The search on the nondominated frontier is like driving a car. The DM can, e. g., increase/decrease the speed, make a turn and brake at any moment he/she likes.

To implement those features, Pareto race uses certain control mechanisms, which are controlled by the following keys:

- (SPACE) BAR, an 'accelerator': Proceed in the current direction at constant speed.
- F1, 'gears (backward)': Increase speed in the backward direction.
- F2, 'gears (forward)': Increase speed in the forward direction.
- F3, 'fix': Use the current value of objective *i* as the worst acceptable value.
- F4, 'relax': Relax the 'bound' determined with key F3.
- F5, 'brakes': Reduce speed.
- F10, 'exit'.
- num, 'turn': Change the direction of motion by increasing the component of the reference direction corresponding to the goal's ordinal number $i \in [1, k]$ pressed by DM.

An example of the Pareto race screen is given in Fig. 3. The screen is associated with the numerical example described in the next section.

Pareto race does not specify restrictive behavioral assumptions for a DM. He/she is free to make a search on the nondominated surface, until he/she believes that the solution found is his/her most preferred one.

Pareto race is only suitable for solving moderate size problems. When the size of the problem becomes large, computing time makes the interactive mode inconvenient. To solve large scale problems [13] proposed a method based on Pareto race. An (interactive) free search is performed to find the most preferred direction. Based on the direction, an nondominated curve can be generated in a batch mode if desired.

Numerical Illustrations

For illustrative purposes, we will consider the following production planning problem, where a decision maker (DM) tries to find the 'best' product-mix for three products: Product 1, Product 2, and Product 3. The production of these products requires the use of one machine (mach. hours), man-power (man hours), and two critical materials (crit. mat. 1 and crit. mat. 2). Selling the products results in profit (profit). Assume that the DM describes his/her decision problem as follows:

> Of course, I would like to make as much profit as possible. Because it is difficult and quite expensive to obtain critical materials, I would like to use them as little as possible, but never more than I have presently in storage (96 units of each). Only one machine is used to produce the products. It operates without any problems for at least 9 hours. The length of the regular working day is 10 hours. People are willing to work overtime which is costly and they are tired the next day. Therefore, if possible, I would like to avoid it. Finally, product 3 is very important to a major customer, and I cannot totally exclude it from the production plan.

The traditional single objective programming considers the problem as a profit maximization problem. The other 'requirements' are taken as constraints. The multiple objective programming takes a 'softer' perspective. We may, for instance, consider the problem as a four objective problem. The DM would like to make as much profit as possible, but simultaneously, he/she would like to use those two critical materials as little as possible, and in addition to maximize the use of product 3. Machine hours and man hours are considered as constraints, but during the search process the role of

Multiple Objective Programming Support, Table 1
The coefficient matrix of the production planning problem

	Prod. 1	Prod. 2	Prod. 3
mach. hours	1.5	1	1.6
man hours	1	2	1
crit. mat. 1	9	19.5	7.5
crit. mat. 2	7	20	9
profit	4	5	3

Multiple Objective Programming Support, Table 2
A sample of solutions for the multiple criteria problem

	I	II	III	IV
Objectives:				
crit. mat. 1	91.46	94.50	93.79	90.00
crit. mat. 2	85.44	88.00	89.15	84.62
profit	30.27	31.00	30.42	29.82
product 3	0.23	0.00	0.50	0.44
Constraints:				
mach. hours	9.00	9.00	9.00	9.00
man hours	9.73	10.00	10.00	9.62
Decision Variables:				
product 1	3.88	4.00	3.45	3.71
product 2	2.81	3.00	3.03	2.74
product 3	0.23	0.00	0.50	0.44

constraints and objectives may also be changed, if necessary.

We assume that the problem can be modeled as an MOLP-model. The coefficient matrix of the problem is given in Table 1.

Thus, we have the following multiple objective linear programming model:

$$\begin{array}{lrcrcrl} \text{crit. mat. 1:} & 9P_1 & + & 19.5P_2 & + & 7.5P_3 & \to \min \\ \text{crit. mat. 2:} & 7P_1 & + & 20P_2 & + & 9P_3 & \to \min \\ \text{profit:} & 4P_1 & + & 5P_2 & + & 3P_3 & \to \max \\ \text{product 3:} & & & & & P_3 & \to \max \end{array}$$

subject to:

$$\begin{array}{lrcrcrl} \text{mach. hours:} & 1.5P_1 & + & P_2 & + & 1.6P_3 & \leq 9 \\ \text{man hours:} & P_1 & + & 2P_2 & + & P_3 & \leq 10 \end{array}$$

The problem has no unique solution. Using the Pareto race (see Fig. 3) or any other software developed for multiple objective programming enables a DM to search nondominated solutions. Which solution he/she will choose as a final one depends entirely on his/her own preferences. Actually, all sample solutions except solution II are somehow consistent with his/her statement above. In solution II, product 3 is excluded from the production plan.

Conclusion

In this article, we have provided an overview on multiple objective programming support. The emphasis was how to find the most preferred alternative from among a set of reasonable (nondominated) alternatives. This kind of the approach is unique for the multiple criteria decision making. We have left other features like structuring the problem, finding relevant criteria etc. beyond this presentation. They are important, but also relevant in the considerations of any decision support system.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Decision Support Systems with Multiple Criteria
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Cohon J (1978) Multiobjective programming and planning. Acad. Press, New York
2. Dyer J, Fishburn P, Wallenius J, Zionts S (1992) Multiple criteria decision making, multiattribute utility theory – The next ten years. Managem Sci 38:645–654
3. Geoffrion A, Dyer J, Feinberg A (1972) An interactive approach for multi-criterion optimization, with an application to the operation of an academic department. Managem Sci 19:357–368
4. Haimes Y, Tarvainen K, Shima T, Thadathil J (1990) Hierarchical multiobjective analysis of large-scale systems. Hemisphere, Washington

5. Hwang C, Masud A (1979) Multiple objective decision making – Methods and applications: A state-of-the-art survey. Springer, Berlin
6. Ignizio J (1976) Goal programming and extensions. D.C. Heath, Lexington
7. Keeney RL, Raiffa H (1976) Decisions with multiple objectives: Preferences and value tradeoffs. Wiley, New York
8. Korhonen P (1987) VIG - A visual interactive support system for multiple criteria decision making. Belgian J Oper Res Statist Comput Sci 27:3–15
9. Korhonen P (1988) A visual reference direction approach to solving discrete multiple criteria problems. Europ J Oper Res 34:152–159
10. Korhonen P, Halme M (1996) Using lexicographic parametric programming for searching a nondominated set in multiple objective linear programming. J Multi-Criteria Decision Anal 5:291–300
11. Korhonen P, Laakso J (1986) A visual interactive method for solving the multiple criteria problem. Europ J Oper Res 24:277–287
12. Korhonen P, Wallenius J (1988) A Pareto race. Naval Res Logist 35:615–623
13. Korhonen P, Wallenius J, Zionts S (1992) A computer graphics-based decision support system for multiple objective linear programming. Europ J Oper Res 60:280–286
14. Lewandowski A, Kreglewski T, Rogowski T, Wierzbicki A (1989) Decision support systems of DIDAS family (Dynamic Interactive Decision Analysis and Support. In: Lewandowski A, Wierzbicki A (eds) Aspiration Based Decision Support Systems. Springer, Berlin, pp 21–47
15. Michalowski W, Szapiro T (1992) A bi-reference procedure for interactive multiple criteria programming. Oper Res 40:247–258
16. Olson D (1996) Decision aids for selection problems. Ser Oper Res. Springer, Berlin
17. Roy B (1973) How outranking relation helps multiple criteria decision making. In: Cochrane J, Zeleny M (eds) Multiple Criteria Decision Making. University South Carolina Press, Columbia, pp 179–201
18. Saaty T (1980) The analytic hierarchy process. McGraw-Hill, New York
19. Sawaragi Y, Nakayama H, Tanino T (1985) Theory of multiobjective optimization. Acad. Press, New York
20. Shin W, Ravindran A (1991) Interactive multiple objective optimization: Survey I – Continuous case. Comput Oper Res 18:97–114
21. Steuer RE (1986) Multiple criteria optimization: Theory, computation, and application. Wiley, New York
22. Steuer R (1992) Manual for the ADBASE multiple objective linear programming package. Department Management Sci, University Georgia, Athens
23. Steuer R, Choo E-U (1983) An interactive weighted Tchebycheff procedure for multiple objective programming. Math Program 26:326–344
24. White D (1990) A bibliography on the applications of mathematical programming multiple-objective methods. J Oper Res Soc 41:669–691
25. Wierzbicki A (1980) The use of reference objectives in multiobjective optimization. In: Fandel G, Gal T (eds) Multiple Objective Decision Making, Theory and Application. Springer, Berlin
26. Wierzbicki A (1986) On the completeness and constructiveness of parametric characterizations to vector optimization problems. OR Spektrum 8:73–87
27. Yu PL (1985) Multiple criteria decision making: Concepts, techniques, and extensions. Plenum, New York
28. Zeleny M (1982) Multiple criteria decision making. McGraw-Hill, New York
29. Zionts S, Wallenius J (1976) An interactive programming method for solving the multiple criteria problem. Managem Sci 22:652–663

Multiplicative Programming

TAKAHITO KUNO
University Tsukuba, Ibaraki, Japan

MSC2000: 90C26, 90C31

Article Outline

Keywords

Optimization; Nonconvex minimization; Low-rank nonconvexity

Multiplicative programming refers to a class of optimization problems containing products of real-valued functions in the objective and/or in the constraints. A product of convex functions is called a *convex multiplicative function*; similar definitions hold for *concave* and *linear* multiplicative functions. Multiplicative functions appear in various areas, including microeconomics [4], VLSI chip design [10] and modular design [2]. Especially in multiple objective decision making, they play important roles [3]. A typical example is a bond portfolio optimization studied in [7], where a number of performance indices such as average coupon rate, average terminal yields and average length of maturity associated with a portfolio (a bundle of assets) are to be optimized (either minimized or maximized) subject to a number of constraints. One handy approach to simultaneously optimizing multiple objectives without a common scale is to optimize the geometric mean, or equivalently the product of these objectives. Thus, we are led to consider a multiplicative programming problem.

The simplest subclass of multiplicative programming problems is a *linear multiplicative program*, which is a quadratic program of minimizing a product of two affine functions $\mathbf{c}_1^\top \mathbf{x} + c_{10}$, $i = 1, 2$, over a polytope $D \subset \mathbf{R}^n$:

$$\begin{cases} \min & f(\mathbf{x}) = (\mathbf{c}_1^\top \mathbf{x} + c_{10})(\mathbf{c}_2^\top \mathbf{x} + c_{20}) \\ \text{s.t.} & \mathbf{x} \in D. \end{cases} \tag{1}$$

This problem was first studied by K. Swarup [13] many years ago, but had attracted little attention until the late 1980s when an intensive research was undertaken [8,12,14]. In general, the objective function f is indefinite; it is quasiconcave on a region where the signs of $c_i^\top \mathbf{x} + c_{i0}s$ are the same, but quasiconvex on a region where the signs are different [1,8]. Therefore, to solve (1), we need to solve a quasiconcave minimization problem:

$$\begin{cases} \min & f(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in D \cap S, \end{cases} \tag{2}$$

and a quasiconcave maximization problem:

$$\begin{cases} \max & f(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in D \cap S, \end{cases} \tag{3}$$

where $S = \{\mathbf{x} \in \mathbf{R}^n \colon \mathbf{c}_i^\top \mathbf{x} + c_{i0} \geq 0,\ i = 1, 2\}$. While (2) belongs to multi-extremal global optimization [6] and is known to be *NP*-hard [11] (cf. also ▶ Complexity Classes in Optimization; ▶ Computational Complexity Theory), problem (3) can be solved using a standard convex minimization technique because maximizing $f(\mathbf{x})$ amounts to minimizing a convex function $-\log(\mathbf{c}_1^\top \mathbf{x} + c_{10}) - \log(\mathbf{c}_2^\top \mathbf{x} + c_{20})$. For the same reason as (3), certain linear programs with additional linear multiplicative constraints, e. g. the modular design problem with $x_i\, y_j \geq \mathrm{b}_{ij}$ [2], can be handled within the framework of convex programming, if $x_i, y_j \geq 0$.

A generalization of (1) is a *convex multiplicative program*, which minimizes a product of several convex functions $f_i(\mathbf{x})$, $i = 1, \ldots, p$, over a compact convex set $D \subset \mathbf{R}^n$:

$$\begin{cases} \min & f(\mathbf{x}) = \prod_{i=1}^{p} f_i(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in D. \end{cases} \tag{4}$$

In most of the existing solutions to (4), the convex functions f_i are assumed to be nonnegative-valued on D. When $f_i(\mathbf{x}') = 0$ for some i and for some $\mathbf{x}' \in D$, the minimum value of (4) is zero; and $\mathbf{x}'$ is a globally optimal solution. We may therefore assume for each i that $f_i(\mathbf{x}) > 0$ for all $\mathbf{x} \in D$. If f is a concave multiplicative function instead of a convex one, the problem is equivalent to a concave minimization problem because log $\mathrm{f}(\mathbf{x}) = \sum_i^{p=1} \log f_i(\mathbf{x})$ is concave. The convex multiplicative program (4) itself can also be transformed into a concave minimization problem (cf. ▶ Concave Programming), though f is not a concave function. For example, introducing additional variables y_i, $i = 1, \ldots, p$, we have an equivalent problem:

$$\begin{cases} \min & \sum_{i=1}^{p} \log y_i \\ \text{s.t.} & \mathbf{x} \in D \\ & f_i(\mathbf{x}) \leq y_i, \quad i = 1, \ldots, p, \\ & \mathbf{y} \geq 0. \end{cases} \tag{5}$$

The number p of $f_i s$ is often very small in comparison with the dimension n of $\mathbf{x}$; e. g. five or so in applications to multiple objective optimization. Owing to this *low-rank nonconvexity* [9], problem (5) can be solved far more efficiently than the usual concave minimization problem of the same size.

In addition to (1) and (4), there are a number of studies on problems with *generalized convex multiplicative functions* of the forms $f(\mathbf{x}) = \prod_{i=1}^{p} f_i(\mathbf{x}) + g(\mathbf{x})$ and $f(\mathbf{x}) = \sum_{i=1}^{p} f_{2i-1}(\mathbf{x})\, f_{2i}(\mathbf{x}) + g(\mathbf{x})$, where the $f_i s$ and g are convex functions. These are all nonconvex minimization problems, each of which has an enormous number of local minima. Nevertheless, algorithms developed in the 1990s can locate a globally optimal solution in a reasonable amount of time, by exploiting special structures of f such as low-rank nonconvexity. A comprehensive review of the algorithms are given by H. Konno and T. Kuno in [5].

See also

- ▶ Global Optimization in Multiplicative Programming
- ▶ Linear Programming
- ▶ Multiparametric Linear Programming
- ▶ Parametric Linear Programming: Cost Simplex Algorithm

References

1. Avriel M, Diewert WE, Schaible S, Zang I (1988) Generalized concavity. Plenum, New York
2. Evans DH (1963) Modular design: A special case in nonlinear programming. Oper Res 11:637–647
3. Geoffrion M (1967) Solving bicriterion mathematical programs. Oper Res 15:39–54
4. Henderson JM, Quandt RE (1971) Microeconomic theory. McGraw-Hill, New York
5. Horst R, Pardalos PM (1995) Handbook of global optimization. Kluwer, Dordrecht
6. Horst R, Tuy H (1993) Global optimization: deterministic approaches. second Springer, Berlin
7. Konno H, Inori M (1989) Bond portfolio optimization by bilinear fractional programming. J Oper Res Japan 32:143–158
8. Konno H, Kuno T (1992) Linear multiplicative programming. Math Program 56:51–64
9. Konno H, Thach PT, Tuy H (1997) Optimization on low rank nonconvex structures. Kluwer, Dordrecht
10. Maling K, Mueller SH, Heller WR (1982) On finding most optimal rectangular package plans. Proc 19th Design Automation Conf, pp 663–670
11. Matsui T (1996) NP-hardness of linear multiplicative programming and related problems. J Global Optim 9:113–119
12. Pardalos PM (1990) Polynomial time algorithms for some classes of constrained nonconvex quadratic problems. Optim 21:843–853
13. Swarup K (1966) Programming with indefinite quadratic function with linear constraints. Cahiers CERO 8:132–136
14. Thoai NV (1991) A global optimization approach for solving the convex multiplicative programming problem. J Global Optim 1:341–357

Multi-Quadratic Integer Programming: Models and Applications

W. Art Chaovalitwongse[1], Xiaozheng He[2], Anthony Chen[3]

[1] Department of Industrial and Systems Engineering, Rutgers University, Piscataway, USA

[2] Department of Civil Engineering, University of Minnesota, Minneapolis, USA

[3] Department of Civil and Environmental Engineering, Utah State University, Logan, USA

MSC2000: 65K05, 90C11, 90C20

Article Outline

Keywords and Phrases

Quadratic programming; Quadratic constraints; Mixed-integer program; Linearization

Introduction

In this contribution, we consider Multi-Quadratic Programming (MQP) problems, where the objective function is a quadratic function and the feasible region is defined by a finite set of quadratic and linear constraints. They can be formulated as follows:

$$\begin{aligned} &\min x^T Qx + c^T x \\ &\text{s.t. } x^T A_j x + B_j x \le b_j\,, \quad j = 1, \dots, m \quad x \ge 0\,, \end{aligned} \tag{1}$$

where A_j is an $(n \times n)$ matrix corresponding to the mth quadratic constraint, and B_j is the jth row of the $(m \times n)$ matrix B.

MQP plays an important role in modeling many diverse problems. The MQP encompasses many other optimization problems since it provides a much improved model compared to the simpler linear relaxation of a problem. Indeed, linear mixed 0-1, fractional, bilinear, bilevel, generalized linear complementarity, and many more programming problems are or can easily be reformulated as special cases of MQP. However, there are theoretical and practical difficulties in the process of solving such problems. However, very large linear models can be solved efficiently; whereas MQP problems are in general $\mathcal{NP}$-hard and numerically intractable. The problem of finding a feasible solution is also $\mathcal{NP}$-hard. This is because MQP is a generalization of the linear complementarity problem [29]. The nonlinear constraints in MQP define a feasible region which is in general neither convex nor connected. Moreover, even if the feasible region is a polyhedron, optimizing the quadratic objective function is strongly $\mathcal{NP}$-hard as the resulting problem is considered to be the disjoint bilinear programming problem. Therefore, finding a finite and exact algorithm that solves large MQP problems is impractical. Even for the convex case (when Q and A_j are positive semidefinite), there are very few algorithms for solving MQP problems. However, the MQP constitutes an important part of mathematical programming problems, arising in various practical applications including facility location, production planning, VLSI chip design, optimal design of water distribution networks, and most problems in chemical engineering design.

The MQP was first introduced in the seminal paper of Kuhn and Tucker [31]. Later on, the case of MQP with a single quadratic constraint in the problem was discussed in [55,56]. The first general approach for solving MQP problems was proposed in [12], where the following two Lagrange functions for MQP are considered:

$$\begin{aligned} L_1(x,\mu) &= x^T Q x + c^T x \\ &\qquad + \sum_{j=1}^{m} \mu_j (x^T A_j x - B_j x - b_j)\,, \\ L_2(x,\mu,\lambda) &= L_1(x,\mu) - \lambda_i x_i\,, \end{aligned}$$

where μ and λ are the multipliers for the quadratic and bound constraints respectively. A cutting plane algorithm was applied to solve this problem; that is, the algorithm solves a sequence of linear master problems that minimize a piecewise linear function constructed from the Lagrange functions for constant x, and a primal problem with either an unconstrained quadratic function (using $L_2(x,\mu,\lambda)$) or a quadratic function over the nonnegative orthant (using $L_1(x,\mu)$) [21].

Multi-Quadratic Integer Program

In this contribution we consider a multi-quadratic integer programming (MQIP) problem with bilevel variables. This problem is a more specific case of MQP. Recently, multi-quadratic zero-one programming problems were proved equivalent to mixed-integer programming problems [16]. In that work, a quadratic zero-one programming was initially proved equivalent to a mixed integer programming problem. Then, the result was extended to the case multi-quadratic programming case.

Throughout this paper, we consider a multi-quadratic zero-one programming problem, which has following form:

$$\begin{aligned} &P_1 :: \min f(x) = x^T A x\,,\ \text{s.t. } Bx \ge b\,,\ x^T C x \ge \alpha\,, \\ &\quad x \in \{0,1\}^n\,,\ \alpha \text{ is a constant}\,. \end{aligned}$$

Notice $x^T C x \ge \alpha$ essentially represent the same the quadratic constraints as $x^T A_j x + B_j x \le b_j$ in problem (1), due to the binary variables' property $x_i x_i = x_i$.

Applications

Bilinear Problem

Each n-dimensional MQP problem can be easily transformed to a $2n$-dimensional bilinear problem. A strategy for reducing the necessary dimension of the resulting bilinear program is also proposed [7,28]. However, on the other hand, bilinear optimization problems are nothing else but a special instance of MQP. Pooling problems in petrochemistry, the modular design problem introduced in [17], in particular the multiple modular design problem [7,18] or the more general modularization of product sub-assemblies [46], and special classes of structured stochastic games [20] are only some examples of the wide range of applications of bilinear programming problems. Another large class of optimization problems are problems with linear or quadratic functions additionally involving Boolean variables (i. e., variables $x_i \in \mathbb{R}$ with the constraint $x_i \in \{0, 1\}$). Another widely explored problem is the problem of packing $n \in \mathbb{N}$ equal circles in a square, which can be transformed to a MQP problem. One looks for the maximum radius r of n non-overlapping circles contained in the unit square. This problem is equivalent to a MQP problem with a linear objective function and concave quadratic constraints.

Minimax Problem

A related class of global optimization problems are minimax location problems [42], which also lead to quadratic constraints. Production planning and portfolio optimization are examples where so-called chance constrained linear programs occur. These are problems, looking similar to linear programs. However, the matrix describing the linear constraints of such problems is not deterministic, it is a stochastic one. Under certain restrictive assumptions it is possible to transform these stochastic constraints to deterministic quadratic constraints [42], such that in general a problem of type MQP is obtained. In [8] it is shown that nonconvex MQP problems can be used for the examination of special instances of nonlinear bilevel programming problems. Other applications of MQP include the fuel mixture problem encountered in the oil industry [43] and also placement and layout problems in integrated circuit design [9,10].

Mixed Integer Problem

As described in the previous section, MQP problem can be easily linearized to a mixed integer zero-one problem with the same problem size. In theory and practice, the linearization technique proposed in [16] has been shown to be superior than other conventional linearization techniques. In medical applications, multi-quadratic zero-one problems were used to model epileptic brains for electrode selection problems. Basically, multi-quadratic zero-one problems were solved to identify the location (electrode) sites of the brain that can detect seizure pre-cursors (predict seizures) [30,34,36]. In order to operate in real time, multi-quadratic zero-one problems were linearized to a mixed integer zero-one problem, which is much faster to solve in practice.

Hence there are many applications of MQP. Whether the MQP is in practice applicable for solving, for example, problems resulting from integer programming problems, depends on the numerical efficiency of the solution method that is used. Up to now only few methods for solving the considered general case of MQP were proposed in the literature. Most of them result from methods being developed for other more general problem classes. In the next section we will discuss some of the solution techniques.

Solution Techniques

There are many different techniques proposed for solving this type of problems, most of them are of branch and bound type or some type of linearization techniques [4,25,26,27,37,38,39,57]. A disadvantage of the standard linearization technique is the additional variables for each product $x_i x_j$, in which the number of new variables is $O(n^2)$, where n is the number of initial 0-1 variables [4,25,26,57]. The method proposed in [16] needs only $O(kn)$ additional continuous variables, where k is the number of quadratic constraints, and the number of initial 0-1 variables remains the same. A branch-and-bound algorithm for solving MQP problems (and other more general problems), when the objective function is separable and the constraint set is linear, was introduced in [19]. The method evolves solving bounding convex envelope approximating problems over successive partitions of the feasible region. This method was later extended to deal with nonconvex constraints but it generates a number of infeasible solutions and does not, in general, converge in a finite number of iterations [53]. An algorithm for the solution to linear problems with an additional reverse convex constraint was proposed in [15]. The algorithm involves partitioning the feasible region into subsets contained in cones originating at an infeasible vertex of the polytope formed by the linear constraints while ensuring that an interior point of the feasible region is contained in each partition. Later on, an algorithm for the solution to problems with concave objective functions and separable quadratic constraint was proposed in [8]. The algorithm uses piecewise linear approximation for the quadratic constraints and solves a MQP problem as a mixed 0-1 linear problem. This algorithm is similar to the solution approaches for concave quadratic problems [40] and for indefinite quadratic problems [35]. During the last decade, several authors are interested in some special cases of MQP. Also, many extensions of MQP have been discussed in the literature. The problem of minimizing an indefinite quadratic objective subject to two-sided indefinite quadratic constraints was discussed in [54]. Under suitable assumptions, they derived necessary and sufficient optimality conditions and gave some conditions for the existence of solutions for this nonconvex program. While several methods have been suggested for solving MQP problems, numerical solutions of the general problem are still rarely available in the literature. By using a double duality argument, under suitable assumptions, the MQP is proved to be equivalent to a convex program [6]. In addition, a problem with a concave quadratic function is proved to be equivalent to a minimax convex problem, and thus can be solved in polynomial time via interior-points methods. The property is no longer true when Q is an indefinite quadratic function [6].

Linear Forms of MQIP

As aforementioned, MQP problems have a close relationship with mixed integer zero-one problem by applying linearization schemes, which have been explored for decades. Although the existing linearization schemes originally were developed for QP instead of MQP, they could be easily applied to MQP, since the quadratic constraints in MQP could be reformulated by using the same technique in linearization considered for the quadratic objective. This section will provide a brief view of major linearization schemes and their applications on MQP problems.

No matter what specific reformulation of the linearization schemes, the ideas are the same as replacing the quadratic product $x_i x_j$ by additional variables. Currently existing linearization schemes were developed in four phases.

The prototype of linearization technique arose in 1960s, proposed by Watters [57] and Zangwill [58] (see also Fortet [22,23]). This approach introduces additional binary variables w_{ij} for replacement of the products $x_i x_j$ and additional constraints, $x_i + x_j - w_{ij} \le 1$ and $x_i + x_j \ge 2w_{ij}, \forall i, j$, for a guarantee of correct replacement. Taken this approach, the MQP P_1 is transformed as following form:

$$\begin{aligned} MIP_1 :: \min f(x,w) = &\sum_i a_{ii}x_i \\ &+ \sum_i \sum_{j>i} (a_{ij} + a_{ji}) w_{ij}, \text{ s.t. } Bx \ge b, \\ &\sum_i c_{ii}x_i + \sum_i \sum_{j>i} (c_{ij} + c_{ji}) w_{ij} \ge \alpha, \\ &x_i + x_j - w_{ij} \le 1, \forall i, j,\ x_i + x_j \ge 2w_{ij}, \\ &\forall i, j,\ x \in \{0,1\}^n, w_{ij} \in \{0,1\}\ \alpha \text{ is a constant} \\ &\text{and } A = (a_{ij}), C = (c_{ij}) . \end{aligned}$$

In this formulation, the quadratic products $x_i x_j$ in objective and constraints, $x^T Ax$ and $x^T Cx$, have been similarly replaced by additional binary variables w_{ij}, and the formulation is consistent with original P_1 by additional constraints, $x_i + x_j - w_{ij} \le 1$ and $x_i + x_j \ge 2w_{ij}, \forall i, j$. Following this seminal work, Glover and Woolsey [25] provided more concise zero-one linear programming formulations, where reformulation rules are given under difference conditions to reduce the numbers of additional constraints and additional variables.

In the second phase development of linearization techniques, researchers recognized the additional binary variables w_{ij} in MIP_1 could be relaxed by continuous ones. Such linearization schemes include the models developed in Glover [24], Glover and Woolsey [26], and Rhys [45]. One scheme with close relationship of the linearization prototype was provided in Glover and Woolsey [26], which introduces additional cut constraints $x_i \ge w_{ij}$ and $x_j \ge w_{ij}, \forall i, j$ enforcing the additional continuous variables w_{ij} to be binary. However, this technique doubles the number of additional constraints added and thereafter enlarges the size of original MQP problems. A straightforward generalization of $x_i \ge x_i x_j, \forall j$ generated their further improvement of this technique in [26] that used alternative concise constraints $(n-i)x_j \ge \sum_{(j>i)} w_{ij}, \forall i$ to enforce the additional variables to be binary, with somewhat fewer constraints. Applying such linearization technique, the MQP P_1 has the following representation:

$$\begin{aligned} MIP_2 :: \min f(x,w) = &\sum_i a_{ii}x_i \\ &+ \sum_i \sum_{j>i} (a_{ij} + a_{ji}) w_{ij}, \text{ s.t. } Bx \ge b, \end{aligned}$$

$$\sum_i c_{ii}x_i + \sum_i \sum_{j>i}(c_{ij}+c_{ji})w_{ij} \geq \alpha,$$
$$x_i + x_j - w_{ij} \leq 1, \forall i, j,\ (n-i)x_j$$
$$\geq \sum_{(j>i)} w_{ij}, \forall i,\ x \in \{0,1\}^n, w_{ij} \geq 0$$
$$\alpha \text{ is a constant and } A = (a_{ij}), C = (c_{ij})\,.$$

The main difference between MIP_1 and MIP_2 is the continuity of w_{ij} and the smaller number of additional constraints.

Beyond the linearization technique in [26], Glover first noticed Petersen's work [41], where the cross products in the model are considered by their upper and lower bounds. Following this idea, Glover, in [24], firstly proposed a linearization technique introducing different continuous variables w_i to the pioneer research. In his linearization scheme, the additional continuous variables are defined by $w_i = x_i \sum_j a_{ij}x_j$, where x_i are binary variables in original model and a_{ij} are quadratic coefficients in the objective function. Further define the lower and upper bounds of $\sum_j a_{ij}x_j$ by $A_i^- = \sum_j\{a_{ij}|a_{ij}<0\}$ and $A_i^+ = \sum_j\{a_{ij}|a_{ij}>0\}$, respectively. Taking the cross products and binary variables into consideration, the additional inequalities $A_i^+x_i \geq w_i \geq A_i^-x_i$ and $\sum_j a_{ij}x_j - A_i^-(1-x_i) \geq w_i \geq \sum_j a_{ij}x_j - A_i^+(1-x_i)$ provide the equivalence of original QP model. Applied such linearization technique, the MQP P_1 has a different structure as follows:

$$MIP_3 :: \min f(w) = \sum_i w_i,\ \text{s.t. } Bx \geq b,$$
$$A_i^+x_i \geq w_i \geq A_i^-x_i, \forall i, \sum_j a_{ij}x_j - A_i^-(1-x_i)$$
$$\geq w_i \geq \sum_j a_{ij}x_j - A_i^+(1-x_i)\forall i \sum_i v_i \geq \alpha,$$
$$C_i^+x_i \geq v_i \geq C_i^-x_i, \forall i, \sum_j c_{ij}x_j - C_i^-(1-x_i)$$
$$\geq v_i \geq \sum_j c_{ij}x_j - C_i^+(1-x_i)\forall i,\ x \in \{0,1\}^n,$$
$$\alpha \text{ is a constant}\,.$$

Notice, in MIP_3, the quadratic constraint $x^TCx \geq \alpha$ is replaced by a series of inequalities $\sum_i v_i \geq \alpha, C_i^+x_i \geq v_i \geq C_i^-x_i, \forall i,\ \sum_j c_{ij}x_j - C_i^-(1-x_i) \geq v_i \geq \sum_j c_{ij}x_j - C_i^+(1-x_i)\forall i$, which follow the definitions in [24] as: $v_i = x_i\sum_j c_{ij}x_j,\ C_i^- = \sum_j\{c_{ij}|c_{ij}<0\}$, and $C_i^+ = \sum_j\{c_{ij}|c_{ij}>0\}$. Compared MIP_3 with MIP_1 and MIP_2, the most important improvement of this linearization technique is that the numbers of additional variables and constraints reduce from $O(n^2)$ to $O(n)$. Some recent papers [1,2] proposed further-improved linearization techniques based on the strategy of Glover's technique, either providing concise formulation or generating tighter upper/lower bounds.

The linearization techniques in the third phase development considered the transformation from the direction of tightness instead of problem size. One typical technique included in this category is the famous Sherali–Adams Reformulation-Linearization Technique (as RLT in short) [3,4,5,50,51,52] which provides wide-range applications. The development milestones of RLT can be found in a recent memorial paper written by Sherali [47]. Interested readers could follow this paper to find the development details of the linearization scheme.

Some practical applications of linearization technique in early 1980s (e. g. [13] considered for solving notorious quadratic assignment problem) generated the experiences that the linearization techniques are practically inefficient although they may have small problem size. Such experience intrigued some researchers to provide better LP structures with tighter bounds, which offer better computational efficiencies, rather than to pursue smaller problem size. The linearization technique shown in [3] provides a structure having tighter bounds for zero-one QP. The transformation happens not only replacing the cross products x_ix_j in the model but also reconstructing the constraints to obtain the tightness. The example given in [3] not only includes the additional constraints and continuous variables, but reconstructs the linear constraints by multiplying x_j and $1-x_j$, respectively. Applying this linearization technique to MQP, P_1 is transformed as follows:

$$MIP_4 :: \min f(x,w) = \sum_i a_{ii}x_i$$
$$+ \sum_i \sum_{j>i}(a_{ij}+a_{ji})w_{ij},\ \text{s.t. } \sum_i B_{ki}x_i - \beta_k$$
$$\geq \sum_{i<j} B_{ki}w_{ij} + \sum_{i>j} B_{ki}w_{ji} + (B_{kj}-\beta_k)x_j$$
$$\geq 0,$$

$$\begin{aligned}&\forall j,k,\ \sum_i c_{ii}x_i + \sum_i \sum_{j>i}(c_{ij}+c_{ji})w_{ij} \ge \alpha,\\ &x_i + x_j - w_{ij} \le 1, \forall i,j,\ x_i \ge w_{ij}, x_j \ge w_{ij}\\ &\forall i,j,\ x \in \{0,1\}^n, w_{ij} \ge 0,\\ &\text{where } \alpha \text{ is a constant and } A = (a_{ij}),\\ &C = (c_{ij}),\\ &B = (B_{ij}), b = (\beta_k)\,.\end{aligned}$$

Notice the linear constraints $Bx \ge b$ are reconstructed as by multiplying x_j and $1 - x_j$ and then have much more complicated but tighter representations. The authors also provided the rigorous proof that the construction is tighter than the linearizaiton provided by Glover [24]. Other than that, this formulation uses the inequalities $x_i \ge w_{ij}$ and $x_j \ge w_{ij}$ instead of $(n-i)x_j \ge \sum_{(j>i)} w_{ij}$ which will weaken the model's tightness as pointed out by the authors.

Using this idea of multiplying x_j and $1 - x_j$ to the feasible set Adams and Sherali [4] provided a linearization strategy to more general MIP with cross products between continuous and binary variables. Comparisons were also provided between the RLT strategy and the linearization techniques in Watters [57], Zangwill [58], Petersen [41], Glover and Woolsey [25,26], and Glover [24]. Along with this direction, Sherali and Adams [49,50,51] generated a hierarchy of relaxations for zero-one polynomial problems. This relaxation strategy generalizes the idea in [3] by introducing a select set of d-degree polynomial terms or factors, where d is an integer less than the number of binary variables. Multiplying the feasible set by d-degree polynomial terms, as the authors showed, obtains an equivalent reformulation, for each $d = 1, \dots, n$, which can enforce the binary restrictions on the original x variables. And these papers also proved that, when $d = n$, the resulting linear system characterizes the convex hull of feasible solutions, and therefore is tighter than any other linearization techniques.

The most recent development, as the final phase, of linearization technique is proposed by Chaovalitwongse et al. [16]. The authors took the dual variables into account, and proposed a new linearization technique based on KKT optimality conditions. Their approach was originally considered for MQP, and is not hard to be utilized for zero-one QP problems. The transformation of MQP P_1 using this linearization strategy can be shown as follows.

$$\begin{aligned}MIP_5 :: \min g(s,x) &= e^T s - M e^T x,\ \text{s.t.}\\ &Ax - y - s + Me = 0,\ Bx \ge b,\\ &y \le 2M(e - x), Cx - z + M'e \ge 0,\\ &e^T z - M' e^T x \ge \alpha,\ z \le 2M'x,\\ &x \in \{0,1\}^n,\ y_i,\ s_i,\ z_i \ge 0,\\ &\text{where } M' = \|C\|_\infty \text{ and } M = \|A\|_\infty\,.\end{aligned}$$

Notice the additional variables including s, y and z, which are introduced from the Lagrangian function of MQP.

Theorem 1 *P_1 has an optimal solution x^0 iff there exist y^0, s^0, z^0 such that (x^0, y^0, s^0, z^0) is an optimal solution of MIP_5.*

Proof 1 See [16]. □

To conclude all the linearization schemes shown herein, we provide a table aggregating the numbers of additional variables and constraints for these techniques as a brief comparison. Assuming we have k linear constraints $\sum_i B_{ji}x_i \ge b_j, j = 1, \dots, k$ and m quadratic constraints $x^T C_j x \ge \alpha_j, j = 1, \dots, m$, in an MQP. Also assume that the number of binary variables $n \gg k$ and $n \gg m$. Then the number of additional variables and constraints of the linearized forms applying different techniques can be shown in the table as follows:

Models	P_1	MIP_1	MIP_2	MIP_3	MIP_4	MIP_5
Additional constraints	0	$O(n^2)$	$O(n^2)$	$O(nm)$	$O(n^2)$	$O(nm)$
Addiontal variables	0	$O(n^2)$	$O(n^2)$	$O(nm)$	$O(n^2)$	$O(nm)$
Total constraints	$O(m+k)$	$O(n^2)$	$O(n^2)$	$O(nm)$	$O(n^2)$	$O(nm)$
Total variables	$O(n)$	$O(n^2)$	$O(n^2)$	$O(nm)$	$O(n^2)$	$O(nm)$

Notice that the problem size is not the only reason of computational efficiencies. The tightness of linearization schemes, as pointed out by [47], may significantly change the effectiveness of the techniques for MQP.

In terms of solution methods, there are many studies in the literature dealing with the MQP. Most of them apply a technique called semidefinite programming to solve the problem. Specifically, these approaches include special branch-and-bound [9,10,32,44], branch-and-cut [11], lift-and-project [11], and the state-of-the-art Interior Point method [14,33]. Some of them have been applied in the commercial software package, e. g., the solvers BARON and CPLEX in GAMS.

References

1. Adams WP, Forrester RJ (2005) A simple recipe for concise mixed 0-1 linearizations. Oper Res Lett 33:55–61
2. Adams WP, Forrester RJ, Glover FW (2004) Comparison and enhancement strategies for linearizing mixed 0-1 quadratic programs. Discret Optim 1:99–120
3. Adams WP, Sherali HD (1986) A tight linearization and an algorithm for zero-one quadratic programming problems. Manag Sci 32:1274–1290
4. Adams WP, Sherali HD (1990) Linearization strategies for a class of zero-one mixed integer programming problems. Oper Res 38(2):217–226
5. Adams WP, Sherali HD (1993) Mixed-integer bilinear programming problems. Math Program 59(3):279–305
6. Aharon B-T, Teboulle M (1996) Hidden convexity in some nonconvex quadratically constrained quadratic programming. Math Program 72:51-639
7. Al-Khayyal FA (1992) Generalized bilinear programming: Part I. Models, applications and linear programming relaxation. Eur J Oper Res 60:306–314
8. Al-Khayyal FA, Horst R, Pardalos PM (1992) Global optimization of concave functions subject to separable quadratic constraints: An application to bilevel programming. Ann Oper Res 34:125–147
9. Al-Khayyal FA, Larsen C, van Voorhis T (1995) A relaxation method for nonconvex quadratically constrained quadratic programs. J Glob Optim 6:215–230
10. Al-Khayyal FA, van Voorhis T (1996) Accelerating convergence of branch-and-bound algorithms for quadratically constrained optimization problems. In: Floudas CA (ed) State of the art in global optimization: computational methods and applications. Kluwer, Dordrecht
11. Audet C, Hansen P, Jaumard B, Savard G (2000) A branch and cut algorithms for nonconvex quadratically constrained program. Math Program 87:131–152
12. Baron DP (1972) Quadratic programming with quadratic constraints. Nav Res Logist Q 19:253–260
13. Bazaraa MS, Sherali HD (1980) Benders' partitioning applied to a new formulation of the quadratic assignment problem. Nav Res Logist Q 27(1):28–42
14. Ben-Tal A, Zibulevsky M (1997) Penalty/Barrier multiplier methods for convex programming problems. SIAM J Optim 7(2):347–366
15. Ben-Saad S (1989) An algorithm for a class of nonlinear convex optimization problems. PhD thesis. University of California, Los Angeles
16. Chaovalitwongse WA, Pardalos PM, Prokoyev OA (2004) Reduction of Multi-Quadratic 0-1 Programming Problems to Linear Mixed 0-1 Programming Problems. Oper Res Lett 32(6):517–522
17. Evans DH (1963) Modular design – A special case in nonlinear programming. Oper Res 11:637–647
18. Evans DH (1970) A note on modular design – A special case in nonlinear programming. Oper Res 18:562–564
19. Falk JE, Soland RM (1969) An algorithm for separable nonvox programming problems. Manag Sci 15(9):550–569
20. Filar JA, Schultz TA (1987) Bilinear programming and structured stochastic games. J Optim Theor Appl 53(1):85–104
21. Floudas CA, Visweswaran V (1995) Quadratic optimization. In: Horst R, Pardalos PM (eds) Handbook of Global Optimization. Kluwer, Dordrecht, pp 217–269
22. Fortet R (1959) L'algèbre de Boole et ses Applications en Recherche Opérationnelle. Cah Cent Etudes Rech Oper 1:5–36
23. Fortet R (1960) Applications de l'algèbre de Boole et Recherche Opérationnelle. Rev Fr Informat Rech Oper 4:17–26
24. Glover F (1975) Improved linear integer programming formulation of nonlinear integer programs. Manag Sci 22:455–460
25. Glover F, Woolsey E (1973) Futher reduction of zero-one polynomial programs to zero-one linear programming. Oper Res 21(1):156–161
26. Glover F, Woolsey E (1974) Converting the 0-1 Polynomial Programming Program to a 0-1 Linear Program. Oper Res 22(1):180-182
27. Hansen P (1979) Methods of nonlinear 0-1 programming. Ann Discret Math 5:53–70
28. Hansen P, Jaumard B (1992) Reduction of indefinite quadratic programs to bilinear programs. J Glob Optim 2:41–60
29. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht
30. Iasemidis LD, Pardalos PM, Shiau D-S, Chaovalitwongse WA, Narayanan K, Kumar S, Carney PR, Sackellares JC (2003) Prediction of Human Epileptic Seizures based on Optimization and Phase Changes of Brain Electrical Activity. Optim Methods Softw 18(1):81–104
31. Kuhn HW, Tucker AW (1951) Nonlinear Programming. In: Nayman J (ed) Proceedings of the Second Berkeley Symposium on Math. Stat. and Prob. University of California Press, Berkeley, pp 481–492
32. Linderoth J (2005) A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs. Math Program 103(2):251–282
33. Nesterov YE, Nemirovskii AA (1994) Interior Point Polynomial Methods in Convex Programming. In: SIAM Series in Applied Mathematics. SIAM, Philadelphia
34. Pardalos PM, Chaovalitwongse WA, Iasemidis LD, Sackellares JC, Shiau D-S, Carney PR, Prokopyev OA, Yatsenko VA (2004) Seizure Warning Algorithm Based on Spa-

tiotemporal Dynamics of Intracranial EEG. Math Program 101(2):365–385
35. Pardalos PM, Glick JH, Rosen JB (1987) Global minimization of indefinite quadratic problems. Computing 39:281–291
36. Pardalos PM, Iasemidis LD, Shiau D-S, Sackellares JC (2001) Quadratic binary programming and dynamic system approach to determine the predictability of epileptic seizures. J Comb Optim 5(1):9–26
37. Pardalos PM, Jha S (1991) Graph separation techniques for quadratic zero-one programming. Comput Math Appl 21(6/7):107–113
38. Pardalos PM, Rodgers GP (1990) Computational aspects of a branch and bound algorithm for quadratic 0-1 programming. Computing 45:131–144
39. Pardalos PM, Rodgers GP (1990) Parallel branch and bound algorithm for quadratic zero-one on a hypercube architecture. Ann Oper Res 22:271–292
40. Pardalos PM, Rosen JB (1986) Methods for global concave minimization: A bibliographic survey. SIAM Rev 28(3):367–379
41. Petersen CC (1971) A note on transforming the product of variables to linear form in linear programs. Working Paper, Purdue University
42. Phan Huy Hao E (1982) Quadratically constrained quadratic programming: Some applications and a method for solution. Z Oper Res 26:105–119
43. Phing TQ, Tao PD, Hoai An LT (1994) A method for solving D.C. programming problems, application to fuel mixture nonconvex optimization problems. J Glob Optim 6:87–105
44. Raber U (1998) A simplicial branch-and-bound method for solving nonconvex all-quadratic programs. J Glob Optim 13:417–432
45. Rhys JMW (1970) A selection problem of shared fixed costs and network flows. Manag Sci 17:200–207
46. Rutenberg DP, Shaftel TL (1971) Product design: Subassemblies for multiple markets. Manag Sci 18(4):B220–B231
47. Sherali HD (2007) RLT: A unified approach for discrete and continuous nonconvex optimization. Ann Oper Res 147:185–193
48. Sherali HD, Adams WP (1984) A decomposition algorithm for a discrete location-allocation problem. Oper Res 32(4):878–900
49. Sherali HD, Adams WP (1990) A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM J Discret Math 3(3):411–430
50. Sherali HD, Adams WP (1994) A hierarchy of relaxations and convex hull Characterizations for mixed-integer zero-one programming problems. Discret Appl Math 52:83–106
51. Sherali HD, Adams WP (1999) A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems. Kluwer, Dordrecht
52. Sherali HD, Adams WP, Driscoll PJ (1998) Exploiting special structures in constructing a hierarchy of relaxations for 0-1 mixed integer problems. Oper Res 46(3):396–405
53. Soland RM (1971) An algorithm for separable nonvox programming problems II: Nonconvex constraints. Manag Sci 17(11):159
54. Stern RJ, Wolkowicz H (1995) Indefinite trust region subproblems and nonsymmetric eigenvalue perturbations. SIAM J Optim 5:286–313
55. Swarup K (1966) Indefinite quadratic programming with a quadratic constraint. Ekonom Obz 4:69–75
56. Van De Panne C (1966) Programming with a quadratic constraint. Manag Sci 12:709–815
57. Watters LJ (1967) Reduction of integer polynomial programming to zero-one linear programming problems. Oper Res 15:1171–117
58. Zangwill WI (1965) Media selection by decision programming. J Advert Res 5(3):30–36

Multi-Scale Global Optimization Using Terrain/Funneling Methods

ANGELO LUCIA
Dept. of Chemical Engineering,
University of Rhode Island, Kingston, USA

MSC2000: 65H20

Article Outline

Introduction

Many problems in optimization involve multiple length and time scales. Perhaps the most commonly studied problems in this area are configurational molecular modeling problems such as phase transitions in wax formation, the structure of Lennard-Jones (LJ) clusters, and protein folding. This class of optimization problems is generally characterized by the presence of many, many stationary points (i. e., minima, first-order saddles, second-order saddles, etc.) that give the appearance of roughness at the small length scale and quite different geometric structure at the large length scale. A good example of the disparity in different length scales is described in Onuchic et al. [28] who illustrate the small-scale geometry of the protein free energy landscape showing many stationary points (or roughness or frustration) at the small length scale and a funnel shaped geometry for the large length scale. This is the multi-scale description we adopt in this expose. It is often acknowledged that finding all stationary points on these multi-scale objective function surfaces is all but impossible [8] for many problems of practical interest and in most cases it is irrelevant. What is of primary interest from a computational chemistry perspective is finding the relatively few important stationary points that describe important physical phenomena – without finding everything else. These important stationary points include global minima, strong local minima and important transitions states that describe rate limiting behavior.

There are many deterministic and/or stochastic methods that can be used to solve multi-scale global optimization problems. See, for example, [1,2,3,4,5,6,7,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,29,30,31,32,33,34,35,36]. These methodologies run the gambit from branch and bound methods, to homotopy-continuation and interval methods, to simulated annealing and genetic algorithms, to terrain and funneling methods, to specialized techniques.

In this chapter, a deterministic terrain/funneling approach for multi-scale global optimization is described. This approach considers two distinct length scales and assumes that the geometry of the larger length scale is funnel shaped. Terrain methods are used to explore the objective function landscape at small length scales and gather point-wise and average gradient and curvature information while funneling methods are used to make large-scale, monotonically decreasing moves at the large length scale and 'funnel' iterates to the global minimizer.

Formulation

The formulation of the problem under consideration is straightforward. It is to find the

$$\text{global min } f(z): \quad \text{subject to } z \le c(z) \tag{1}$$

where $f = f(z)$ is a C^3 objective function defined on R^n subject to bounds on variables, $c(z)$, and where z are the optimization variables. It is assumed that f has two distinct length scales – a small length scale of considerable roughness and a large length scale where f has non-quadratic behavior. For the discussions that follow, it is convenient to denote the gradient of f by $g = g(z)$ and the Hessian matrix of f by $h = h(z)$.

Generally, formulations based on second-order Taylor series expansion are adequate to describe behavior at the small length scale, and methods based on quadratic approximations of f are well known. However, since it is assumed that the behavior of the objective function, f, is non-quadratic at the large length scale, funneling methods are used to build approximations to the large-scale geometry of f using the funnel function given by

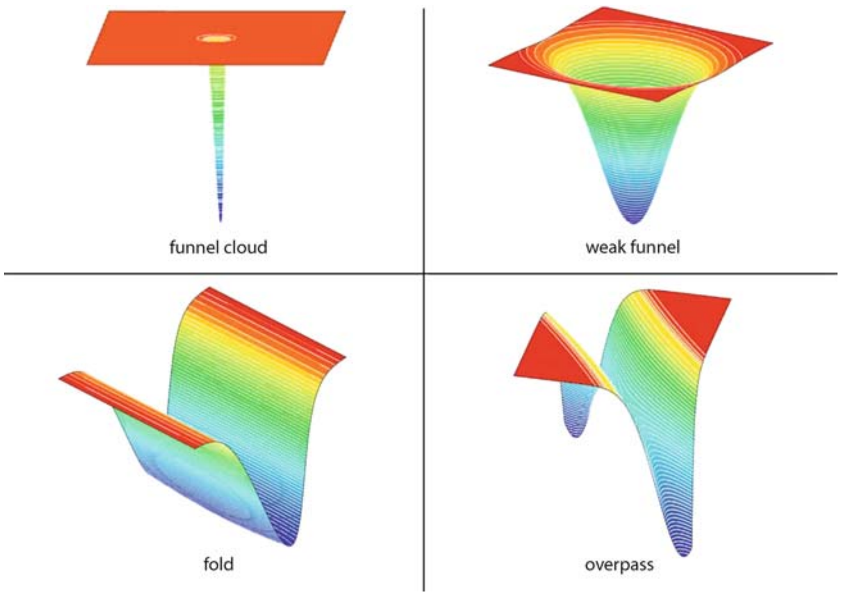
$$\mathrm{F(z)} = \mathrm{F}_0 - \Gamma \exp[-q(z)] \tag{2}$$

where $q(z) = \frac{1}{2} z^{\mathrm{T}} A z + b^{\mathrm{T}} z + c$, and where $\Gamma > 0$, F_0 and c are scalar parameters, b is an n-dimensional vector, and A is an n x n symmetric matrix. The functional form of Eq. (2) is interesting because it is non-convex, has a unique global minimum when A is positive definite, and contains certain inherent self-scaling characteristics. Figure 1 gives an illustration of the funnel geometries that can be represented by the functionality of Eq. (2).

The funnels at the top and the bottom left in Fig. 1 each have unique minimum because A is positive definite while the one at the bottom right has two minima since A is indefinite.

Methods

In this sub-section, the global terrain method, a funneling algorithm, and a multi-scale global optimization method are described.



Multi-Scale Global Optimization Using Terrain/Funneling Methods, Figure 1
Various Funnel Geometries

A Global Terrain Method for Optimization at the Small Length Scale

Terrain methods [18,19,20] have described in detail in a separate chapter in this encyclopedia and are briefly summarized in this chapter for the purpose of continuity of presentation. Terrain methods are used to locate sets of stationary and singular points of the objective function. They do this by following valleys and moving up and down the landscapes of $g^T g$ and f. Key among the equations used in terrain methods is the characterization of valleys as solutions, V, to a sequence of general nonlinearly constrained optimization problems

$$\mathrm{V} = \{\min h^T g^T hg \text{ such that } g^T g = L, \text{ for all } L \varepsilon\ \Lambda\} \tag{3}$$

where g and h are defined as before, where L is any given value (or level) of the least-squares objective function, and where Λ is some collection of contours. Terrain methods require

1) Reliable downhill equation solving
2) Reliable and efficient computation of singular points
3) Efficient uphill movement comprised of predictor-corrector calculations
4) Reliable and efficient eigenvalue-eigenvector computations
5) Effective bookkeeping
6) A termination criterion to decide when the computations have finished.

In this chapter, a terrain methodology is used to find sets of stationary and singular points and to determine *average* gradient and *average* curvature (or Hessian matrix) information along a given terrain path. Average gradient and curvature information is calculated from the mean value theorem using the following equations.

$$\langle g \rangle = (1/\alpha) \quad \int g[z(\alpha)]d\alpha \tag{4}$$

$$\langle h \rangle = (1/\alpha) \quad \int h[z(\alpha)]d\alpha \tag{5}$$

where α is some relevant length of the smooth terrain path connecting any set of stationary and singular points. It is important for the reader to understand that it is the set of stationary and singular points as well as average gradient and Hessian matrix information that are communicated from the small length scale to the large length scale. This will be discussed again a little later in this chapter.

A Funneling Method for Optimization at the Large Length Scale

To build iterative global funnel approximations of the objective function we match function, gradient, and second derivative information of the true objective function, f, g and h, with the function, gradient, and Hessian matrix information, F, G and H respectively, of the funnel function at various points, where $G = G(z)$ and $H = H(z)$ are given by

$$G(z) = \Gamma \exp[-q(z)][Az + b] \quad (6)$$

$$H(z) = \Gamma \exp[-q(z)][A - (Az + b)(Az + b)^{\mathrm{T}}] \quad (7)$$

Note that if $f(z)$ is used in place of $F(z)$ in Eq. (2), then it follows that

$$\gamma(z) = F_0 - f(z) = \Gamma \exp[-q(z)] \quad (8)$$

where $\gamma > 0$ is a positive scaling factor that depends on a single numerical measurement, $f(z)$, and the scalar parameter, F_0. Moreover, replacing $G(z)$ with $g(z)$ and $H(z)$ with $h(z)$ in Eqs. (6) and (7) respectively give the equations

$$[Az + b] = g(z)/\gamma \quad (9)$$

$$A = [\gamma h + gg^{\mathrm{T}}]/\gamma^2 \quad (10)$$

Equations (8), (9), and (10) show that it is possible to estimate A and b from values of $f(z)$, $g(z)$, and $h(z)$ using interpolation formula at two or more iterates.

Interpolating Formulae Let z_k be any value of the unknown optimization variables with corresponding objective function, gradient and Hessian matrix values f_k, g_k and h_k respectively. Also let z_{k+1} be some other arbitrary but not necessarily nearby or successive iterate with corresponding function, gradient and Hessian matrix values f_{k+1}, g_{k+1} and h_{k+1}. Writing Eq. (8) for z_k and z_{k+1} and then subtracting the latter from the former, eliminates F_0 and gives

$$\gamma_{k+1} - \gamma_k = f_k - f_{k+1} \quad (11)$$

Repeating the same algebra using Eq. (10) yields

$$\gamma_k^2[\gamma_{k+1}h_{k+1} + g_{k+1}g_{k+1}^{\mathrm{T}}] - \gamma_{k+1}^2[\gamma_k h_k + g_k g_k^{\mathrm{T}}] = 0 \quad (12)$$

Equations (11) and (12) form a set of $[1 + n(n + 1)/2]$ nonlinear equations in the two unknowns γ_k and γ_{k+1} when the symmetry of the associated matrices is taken into account. This together with Eq. (11) gives a total of $[1 + n(n + 1)/2]$ nonlinear equations. For $n = 1$, there are two equations and two unknowns. When $n > 1$ there are more equations than unknown variables. However, irrespective of this, two equations for which γ_k and $\gamma_{k+1} > 0$ can be determined using the Routh criterion.

Estimating Funnel Parameters Calculated values of γ_k and γ_{k+1} can be used to determine the matrix A from Eq. (10) – using gradient and Hessian matrix information either at z_k or z_{k+1}. Following this, the parameter b can be computed by simply rearranging Eq. (9) to give

$$b = g(z)/\gamma - Az \quad (13)$$

while F_0 can be calculated from $F_0 = f(z) + \gamma$. Like A, the values of $\boldsymbol{b}$ and F_0 can be determined using function and gradient values at either z_k or z_{k+1}.

Finding the Funnel Minimum It is then straightforward to estimate the unique global minimizer of the *funnel approximation*, say y, by simply solving

$$Ay = -b \quad (14)$$

Note that Eq. (10) shows that the matrix A is generated from a rank-one, positive semi-definite correction to $h(z)$, that the sign definiteness of A can be controlled by the parameter γ, and that γ appears to also play the role of a self-scaling factor.

Communication Between Length Scales One of the keys to success in any multi-scale global optimization methodology is the communication between length scales. In the terrain/funneling approach to multi-scale global optimization, small-scale calculations communicate *average* gradient and Hessian matrix information at two distinct points to the large length scale. The large length scale optimizations, on the other hand, communicate an estimate of the values of the optimization variables at a converged minimum of the funnel approximation to the small length scale and identify the next region on the objective function surface on which small-scale optimizations should be conducted.

A Multi-Scale Global Optimization Method

The details of a multi-scale global optimization methodology based on the terrain and funneling methods is as follows.

1 Perform two sets of small-scale optimization calculations using the terrain methodology starting from two different points on the objective function surface. Calculate *average* gradient, and *average* Hessian information along the resulting terrain paths. Thus at the kth funnel iteration the following information is available – z_k, f_k, g_k and h_k and $z_{k+1}, f_{k+1}, g_{k+1}$ and h_{k+1} such that $f_{k+1} < f_k$.

2 Conduct iterative large-scale optimization calculations with the funneling methodology initialized using the objective function, average gradient and average Hessian information from the small-scale optimization calculations to find a funnel minimum that also corresponds to a stationary point on the true objective function surface. To do this,

   a) Solve Eqs. (11) and (12) for γ_k and γ_{k+1}.
   b) Using γ_{k+1}, calculate A and b from Eqs. (10) and (13) respectively.
   c) Determine an estimate of funnel minimum, y, from Eq. (14).
   d) Evaluate $f(y)$, $g(y)$ and $h(y)$.
   e) Test $f(y)$ against f_{k+1}. If $f(y) < f_{k+1}$, then go to step 2f for the next funnel iteration. Else set $\gamma_k = \gamma_k/2$ and return to step 2a.
   f) Set $z_{k+1} = y$, $f_{k+1} = f(y)$, $g_{k+1} = g(y)$, and $h_{k+1} = h(y)$.
   g) If $||g(z_{k+1})|| < \varepsilon$, set $y = y^*$, and go to step 3; else go to step 2a.

3 Conduct a new set of small-scale terrain calculations using the funnel minimum from step 2. Calculate *average* gradient and *average* Hessian information along the resulting terrain path such that new values of z_{k+1}, f_{k+1}, g_{k+1} and h_{k+1} satisfy the condition $f_{k+1} < f_k$.

4 Repeat step 2 using the new small-scale information and z_k, f_k, g_k and h_k from step 1.

5 Repeat steps 2 and 3 until there is no further decrease in the objective function.

Here we describe step 2 of the multi-scale algorithm. The most effective way to determine γ_{k+1} in step 2a is to rearrange Eq. (11) for γ_{k+1} in terms of γ_k and then substitute the resulting expression into Eq. (12) This gives a cubic polynomial equation in γ_k and shows that there are *three* possible values of γ_k and thus three possible sets of scaling factors (γ_k, γ_{k+1}). Using an equation solver like Newton's method, it is easy to find one solution for γ_k. The other two values of γ_k can be determined by deflation of the cubic equation to a quadratic equation and by using the quadratic formula. The correct value of γ_k is the smallest *real* valued $\gamma_k > 0$ such that $\gamma_{k+1} > \gamma_k$, where $\gamma_{k+1} = f_k - f_{k+1} + \gamma_k$. Step 2b is straightforward and step 2c requires the solution of a system of linear equations. Step 2d evaluates the *actual* function, gradient, and Hessian matrix at the funnel iterate y. Step 2e is used to ensure monotonic decreasing objective function values by halving γ_{k+1} until $f(y) < f_{k+1}$ while step 2f replaces the information associated with z_{k+1} with that for the funnel iterate y. Finally, step 2g checks the norm of the gradient of the objective function and terminates the funnel iterations once that norm of the gradient falls below the specified tolerance. Note that any point, y^*, that satisfies the convergence condition in step 2g is simultaneously a stationary point of $f(z)$ and a minimizer of the funnel function $F(z)$.

The proposed multi-scale optimization algorithm is very robust. The reason for this is because if the funneling algorithm gets trapped at a local minimum, the methodology returns to the small-scale terrain calculations to get average gradient and average Hessian information around that local minimum. Replacing point-wise zero-valued gradient information at a local minimum with averaged non-zero valued gradient information forces the optimizer to look for a minimizer that is deeper in funnel. By forcing movement further down the funnel in this way, the multi-scale algorithm will continue to improve the value of the objective function in a monotonic fashion until it reaches the global minimum at the bottom of the funnel.

Application

In this sub-section, a simple illustration of the multi-scale global optimization methodology is given using the classical thirteen-particle Lennard-Jones (LJ_{13}) problem. This example was selected because it is the first in the series of Lennard-Jones clusters that truly has a single funnel based on the Mackay icosahedron structure with the global minimum lying at the bottom of the funnel [8]. The LJ_{13} cluster has 33 unknown Cartesian coordinates, 1509 minima, 116,835 first-order and second-order transition states, and many higher order transition states [9].

Small Scale Terrain Optimization

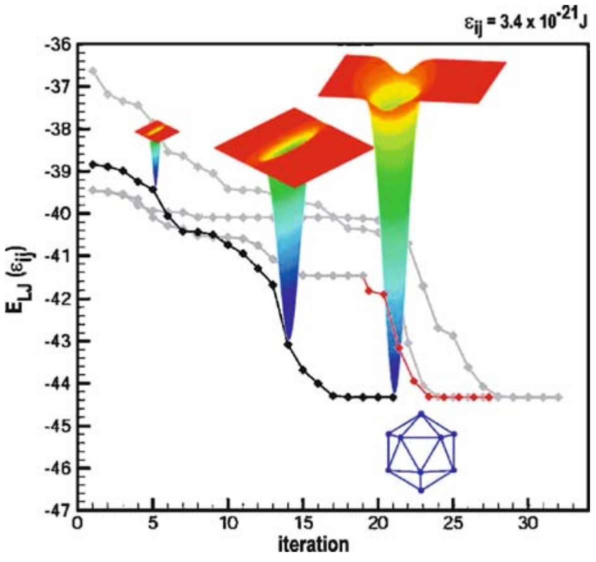
Small-scale optimization calculations were performed using the terrain methodology from two different starting points. From the first starting point, which corresponds to an energy of $E = -38.8572$, six (6) stationary and singular points along a terrain path were calculated requiring 4832 function and gradient evaluations and 616.68 s of computer time. Average gradient and Hessian information was accumulated along the terrain path defined the path integrals given by Eqs. (4) and (5). From a second and quite different starting point on the objective function surface another set of small-scale optimization calculations was performed using the terrain methodology. The energy at this second starting point was $E = -38.4246$, and five (5) stationary and singular points were calculated requiring 1415 function and gradient evaluations and 214.25 s of computer time. Again, average gradient and Hessian information was accumulated along the terrain path using the path integrals using Eqs. (4) and (5).

Large Scale Funneling Optimization

To build an approximation of the large-scale geometry, two singular points – one from each of the two sets of terrain calculations – were selected. Using the function values at these singular points as well as average gradient and average Hessian information, funnel parameters were determined and a first estimate of the funnel minimum was calculated. We then replaced one of the singular points with this estimate of the funnel minimum and repeated the funnel optimization calculations until $||g|| < \varepsilon = 1\,x\,10^{-4}$. Figure 2 summarizes these results, which are shown by the curve of dark or black diamonds. The resulting structure for LJ_{13} is also shown in Fig. 2. Twenty (20) funneling iterations were required for convergence to the global minimum at $E = -44.3268$ on the LJ_{13} energy surface.

Robustness

Terrain/funnel calculations were performed using many other arbitrary starting points on the potential energy surface for the LJ_{13} cluster to illustrate the robustness of this multi-scale optimization method. Results for some of these calculations are also shown in Fig. 2 by the two light gray curves with diamonds. Note that these multi-scale optimization calculations using the terrain/funneling approach also converge easily and monotonically to the global minimum in 32 and 27 funnel iterations respectively from these starting points. In all cases, the funneling portion of the overall multi-scale algorithm finds the global minimum in less than 0.35 s.



Multi-Scale Global Optimization Using Terrain/Funneling Methods, Figure 2
Multi-Scale Terrain/Funneling Calculations for the LJ_{13} Cluster

Reliability – Avoiding Traps at Local Minima

To show that the proposed optimization approach does not get trapped at local minima, terrain/funneling calculations were repeated from a different set of initial terrain calculations. In particular, small-scale terrain calculations starting from points on the energy surface corresponding to energy values of $E_1 = -39.1597$ and $E_2 = -38.4246$ were performed and average gradient and average Hessian information gathered along the resulting terrain paths. Using this information, the funneling algorithm converged to a *local* minimum on the potential energy surface at $E_3 = -41.4445$ in 18 funnel iterations. The results of these calculations correspond to the gray curve that terminates at $E = -41.4445$ in

Fig. 2. This local minimum was then used as a starting point to perform a third set of small-scale terrain calculations and to gather average gradient and average Hessian information in the valley around the local minimum. Using this average information around the local minimum, E_3, together with average information around E_1, a second set of iterative funneling calculations was performed. In this case, the funneling calculations located the global minimum at $E = -44.3268$ on the energy surface in 16 funnel iterations. This second set of funnel iterations is shown by the red curve in Fig. 2.

Those readers interested in the numerical details of the LJ_{13} illustration are encouraged to contact the author, who is quite willing to provide all computer-generated numerical results for this example.

References

1. Aluffi-Pentini F, Parisi V, Zirilli F (1985) Global optimization and stochastic differential equations. J Opt Theory Appl 47:1–16
2. Bahren J, Protopopescu V (1996) Generalized TRUST algorithms for global optimization. In: Floudas CA, Pardalos PM (eds) State of the Art in Global Optimization. Kluwer, Dordrecht, pp 163–180
3. Baker J (1986) An algorithm for the location of transition states. J Comput Chem 7:385–395
4. Bilbro GL (1994) Fast stochastic global optimization. IEEE Trans Sys Man Cyber 4:684–689
5. Cerjan CJ, Miller WH (1981) On finding transition states. J Chem Phys 75:2800–2806
6. Deaven DM, Tit N, Morris JR, Ho KM (1996) Structural optimization of Lennard-Jones clusters by a genetic algorithm. Chem Phys Lett 256:195–200
7. DeJong K (1975) An analysis of the behavior of a class of genetic adaptive systems. Ph.D. Thesis, Univ. of Michigan, Ann Arbor, Princeton
8. Doye JPK, Miller MA, Wales DJ (1999) Evolution of the potential energy surface with size for Lennard-Jones clusters. J Chem Phys 111:8417–8428
9. Doye JPK, Wales DJ (2002) Saddle points and dynamics of Lennard-Jones clusters, solids and supercooled liquids. J Chem Phys 116:3777–3788
10. Gibson KD, Scheraga HA (1987) Revised algorithm for the build-up procedure for predicting protein conformation by energy minimization. J Comput Chem 8:826–834
11. Gregurick SK, Alexander MH, Hartke B (1996) Global geometry optimization of $(Ar)_n$ and $B(Ar)_n$ clusters using a modified genetic algorithm. J Chem Phys 104:2684–2691
12. Hansen ER (1980) Global optimization using interval analysis – The multidimensional case. Numer Math 34:247–270
13. Henkelman G, Johannesson G, Jonsson H (2000) Methods for finding saddle points and minimum energy paths. In: Schwartz SD (ed) Progress in Theoretical Chemistry and Physics. Kluwer, Dordrecht, 5:269–300
14. Holland JH (1992) Genetic algorithms. Sci Am 267:66
15. Jones DT, Taylor WR, Thornton JM (1992) A new approach to protein folding recognition. Nature 358:86–89
16. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
17. Levy AV, Montalvo A (1985) The tunneling algorithm for the global minimization of functions. SIAM J Sci Stat Comp 6:15–29
18. Lucia A, Yang F (2002) Global terrain methods. Comput Chem Eng 26:529–546
19. Lucia A, Yang F (2003) Multivariable terrain methods. AIChE J 49:2553–2563
20. Lucia A, DiMaggio PA, Depa P (2004) A geometric terrain methodology for global optimization. J Global Optim 29:297–314
21. Lucia A, DiMaggio PA, Depa P (2004) Funneling algorithms for multi-scale optimization on rugged terrains. Ind Eng Chem Res 43:3770–3781
22. Maranas CD, Floudas CA (1992) A global optimization approach to Lennard-Jones microclusters. J Chem Phys 97:7667–7678
23. Maranas CD, Floudas CA (1995) Finding all solutions to nonlinearly constrained systems of equations. J Global Optim 7:143–182
24. Matro A, Freeman DL, Doll JD (1994) Locating transition states using double-ended classical trajectories. J Chem Phys 101:10458–10463
25. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21:1087–1092
26. Muller K, Brown LD (1979) Location of saddle points and minimum energy paths by a constrained simplex optimization procedure. Theoret Chim Acta 53:75–93
27. Niesse JA, Mayne HR (1996) Global geometry optimization of atomic clusters using a modified genetic algorithm in space-fixed coordinates. J Chem Phys 105:4700–4706
28. Onuchic JN, Luthey-Schulten Z, Wolynes PG (1997) Theory of protein folding: the energy landscape perspective. Annu Rev Phys Chem 48:545–600
29. Piela L, Kostrowicki J, Scheraga HA (1989) The multiple minima problem in conformational analysis of molecules. Deformation of the potential energy hypersurface by the diffusion equation method. J Phys Chem 93:3339–3346
30. Pincus MR, Klausner RD, Scheraga HA (1982) Calculation of the three dimensional structure of the membrane-bound portion of melittin from its amino acid sequence. Proc Natl Acad Sci USA 79:5107–5110
31. Scheraga HA (1974) Prediction of protein conformation. In: Anfinsen CB, Schechter AN (eds) Current Topics in Biochemistry. Acad Press, New York, p 1

32. Schnepper CA, Stadtherr MA (1996) Robust process simulation using interval methods. Comput Chem Eng 20:187–199
33. Sevick EM, Bell AT, Theodorou DN (1993) A chain of states method for investigating infrequent events in processes occurring in multistate, multidimensional systems. J Chem Phys 98:3196–3212
34. Sun AC, Seider WD (1992) Homotopy-continuation algorithm for global optimization. In: Floudas CA, Pardalos PM (eds) Recent Advances in Global Optimization. Princeton Univ Press, Princeton, pp 561–592
35. Wales DJ, Doye JPK (1997) Global optimization by basin hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms. J Phys Chem A 101:5111–5116
36. Westerberg KM, Floudas CA (1999) Locating all transition states and studying the reaction pathways of potential energy surfaces. J Chem Phys 110:9259–9295

Multistage Stochastic Programming: Barycentric Approximation

KARL FRAUENDORFER, MICHAEL SCHÜRLE
Institute Operations Res., University St. Gallen,
St. Gallen, Switzerland

MSC2000: 90C15

Article Outline

Keywords

Stochastic programming; Approximation

Many problems in finance, economics and other applications require that decisions $x_t \in \mathbf{R}_t^n$ are made periodically over time, depending on observations of uncertain data (η_t, ξ_t) in future periods $t = 1, \dots, T$. Here, it is distinguished between random data $\eta_t \in \Theta_t \subset \mathbf{R}_t^K$ that influence prices in the objective function and random data $\xi_t \in \Xi_t \subset \mathbf{R}_t^L$ that affect the demand on the right-hand side of constraints in an optimization problem.

Once an observation (η_t, ξ_t) becomes available, the decision maker has to determine a policy x_t that minimizes the costs $\rho_t(x^{t-1}, x_t, \eta^t)$ in t plus the expected costs in the subsequent periods $t+1, \dots, T$, subject to a set of constraints $f_t(x^{t-1}, x_t) \le h(\xi^t)$. Both the objective function and the constraints may depend on the sequences of observations $\eta^t = (\eta_1, \dots, \eta_t)$, $\xi^t = (\xi_1, \dots, \xi_t)$ up to t and earlier decisions $x^{t-1} = (x_0, \dots, x_{t-1})$. Obviously, an action x_t must be selected after (η_t, ξ_t) is observed but before the future outcomes $\eta_{t+1}, \dots, \eta_T$ and $\xi_{t+1}, \dots, \xi_T$ are known, i. e. the decision is based only on information available at time t. Hence, one obtains a sequence of decisions with the property $x_0, x_1(\eta^1, \xi^1), \dots, x_T(\eta^T, \xi^T)$, called *nonanticipativity*. This results in a *multistage stochastic program*, which may be written in its dynamic representation as a series of nested two-stage programs (with $\phi_{T+1}(\cdot) := 0$, see [4]):

$$\phi_t(x^{t-1}, \eta^t, \xi^t) := \min\left\{\rho_t(x^{t-1}, x_t, \eta^t) + \int \phi_{t+1}(x^{t-1}, x_t, \eta^t, \eta_{t+1}, \xi^t, \xi_{t+1})\, dP_{t+1}\right\}, \quad t = 0, \dots, T, \qquad (1)$$

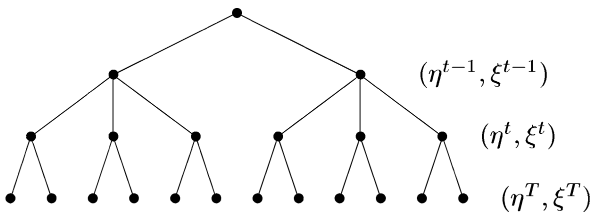
where the expectation is taken w.r.t. the probability measure $P_{t+1}(\eta_{t+1}, \xi_{t+1} | \eta^t, \xi^t)$ of the joint distribution of (η_{t+1}, ξ_{t+1}), subject to

$$f_t(x^{t-1}, x_t) \le h(\xi^t), \quad x_t \ge 0. \qquad (2)$$

In case of discrete distributions, it is well known that one can immediately transform the stochastic multistage program given by (1) and (2) into a (large) deterministic equivalent problem which can be solved by standard optimization tools, possibly combined with *decomposition techniques* to exploit the special structure of the problem (see e. g. [1,9,10]). However, if the distribution is continuous with some density function, it is in general impossible to do the integration in (1) exactly. One way to overcome this difficulty is to approximate the (continuous) probability measure P_t by a discrete one Q_t. In MSP, this is usually done by constructing a *scenario tree* which can be illustrated as follows:

Together with the associated scenario probabilities, this tree is defined formally as

$$\mathcal{A} := \left\{(\eta^T, \xi^T) \colon \begin{array}{c}(\eta_t, \xi_t) \in \mathcal{A}_t(\eta^{t-1}, \xi^{t-1}) \\ \forall t > 0\end{array}\right\}, \qquad (3)$$
$$q(\eta^T, \xi^T) := \prod_{t=1}^{T} q_t(\eta_t, \xi_t | \eta^{t-1}, \xi^{t-1}).$$



Multistage Stochastic Programming: Barycentric Approximation, Figure 1

The scenario tree represents an approximation of the discrete-time process (η_t, ξ_t; t = 1, …, T), and $\mathcal{A}_t(\cdot)$ denotes the set of finitely many outcomes for (η_t, ξ_t) conditioned on the history (η^{t-1}, ξ^{t-1}). Again, this results in a sparse large scale program. Naturally, the question arises how good the accuracy of the associated (deterministic) optimization problem is, and if the set of scenarios can be improved w.r.t. the accuracy.

For convex optimization problems where the random data are decomposable in two groups, one that determines the cost function and the second one affecting the demand, it can be shown (see [4] for details) that the *value function* (1) is a saddle function for all t = 1, …, T under the following conditions:

i) $\rho_t(\cdot)$ is concave in η_t,
ii) the left-hand sides of the constraints are deterministic, and
iii) the distribution function of $P_t(\cdot|\ \eta^{t-1}, \xi^{t-1})$ depends linearly on the past.

Then, (1) is concave in η_t and convex in (x_t, ξ_t). The situation where assumptions i)–iii) are fulfilled is called the entire convex case.

This underlying saddle property of the value function motivates the application of *barycentric approximation* which derives two scenario trees $\mathcal{A}^u$ and $\mathcal{A}^l$. The associated approximate deterministic programs provide upper and lower bounds to the original problem. In this sense, barycentric approximation is a generalization of the inequalities due to H.P. Edmundson [2] and A. Madansky [8] (see e. g. ► Stochastic Programs with Recourse: Upper Bounds) and J.L. Jensen [6] that is applicable to saddle functions of correlated random data. Here, it is assumed that $\Theta_t \subset R_t^K$ and $\Xi_t \subset R_t^L$ are regular simplices whose vertices are denoted by u_{ν_t}, ν_t = 0, …, K_t, and $v_{\mu t}$, μ_t = 0, …, L_t. Both Θ_t and Ξ_t may depend on prior observations (η^{t-1}, ξ^{t-1}) although this is not stressed in the notation for simplicity.

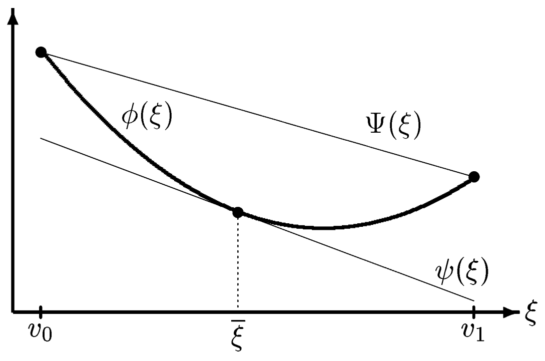
To illustrate the way the discretization is performed, assume that a two-stage problem is given (the time index is omitted here) with deterministic objective, i. e. only the right-hand side coefficients $h(\xi)$ are random (see e. g. [7]). For any $\xi \in \Xi$, the *barycentric weights* $\tau_0(\xi), \dots, \tau_L(\xi)$ w.r.t. the simplex Ξ are given by

$$\begin{aligned}&\tau_0 + \cdots + \tau_L = 1,\\ &\tau_0 v_0 + \cdots + \tau_L v_L = \xi.\end{aligned} \tag{4}$$

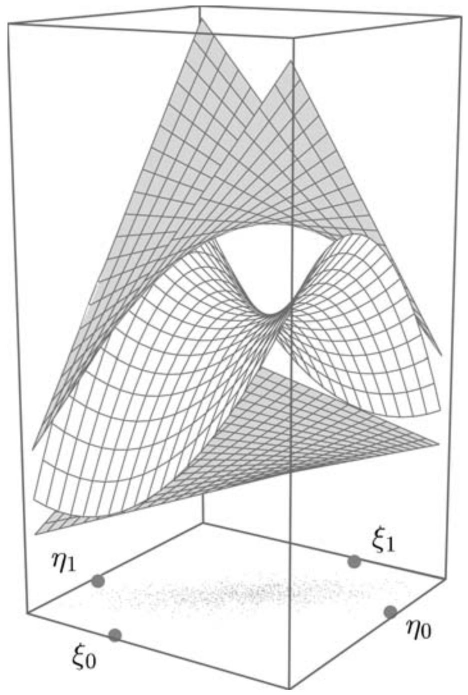
Since $\phi(x, \xi)$ is convex in ξ for all x, $\varphi(\xi) := \phi(\widehat{x}, \xi)$ is a convex function for any fixed first-stage decision $\widehat{x}$. Due to convexity, $\varphi(\xi)$ is bounded from above for all $\xi \in \Xi$ by a linear function $\Psi(\xi) = \sum_{\mu=0}^{L} \tau_\mu(\xi)$ v_μ. To construct the 'classical' *Edmundson–Madansky upper bound* (EM) for $\int \varphi(\xi)\, dP$ over the simplex Ξ, ξ is replaced by a discrete random variable with the same expectation, attaining values $v_0, \dots, v_L$. To obtain the corresponding probabilities, ξ has to be replaced by $overline\xi = \int \xi\, dP$ in (4), and the system must be solved for $\tau_0, \dots, \tau_L$. Then, $\int \varphi(\xi)\, dP \leq \sum_{\mu=0}^{L} \tau_\mu(\overline{\xi}) v_\mu$, and the weights may be interpreted as the probabilities of the discrete outcomes.

On the other hand, a lower bound can be found using *Jensen's inequality*: $\varphi(\overline{\xi}) \leq \int \varphi(\xi)\, dP$, i. e. by evaluation of the function for the expectation of ξ, and the tangent $\psi(\xi)$ to $\varphi(\xi)$ at $\overline{\xi}$ is a lower bound to the original function. Both linear approximations $\psi(\xi)$ and $\Psi(\xi)$ to the convex value function for a given policy are shown in Fig. 2.

From a computational viewpoint, the original function $\varphi(\xi)$ is replaced by two linear affine functions. Clearly, $\psi(\xi)$ and $\Psi(\xi)$ can be integrated easily over the support of ξ. If there is only randomness in the objec-



Multistage Stochastic Programming: Barycentric Approximation, Figure 2



Multistage Stochastic Programming: Barycentric Approximation, Figure 3

tive with deterministic right-hand sides, a lower and an upper bound can be constructed by applying the same procedure to the dual concave (maximization) problem, deriving an upper bound from Jensen's inequality and a lower approximation with the EM-rule.

Barycentric approximation combines these concepts for stochastic objective and right-hand sides [3] and extends them to the multistage case [4,5]. It derives distinguished points, so-called *generalized barycenters*, where the value function (1) must be supported by two bilinear functions to minimize the error induced by the approximation. This is shown in Fig. 3 for $K_t = L_t = 1$, where the minorant is supported at ξ_0 and ξ_1 and the majorant at η_0 and η_1.

Let $\lambda_{t,0}(\eta_t), \dots, \lambda_t, K_t(\eta_t), \tau_t, 0(\xi_t), \dots, \tau_t, L_t(\xi_t)$ be the barycentric weights w.r.t. Θ_t and Ξ_t defined analogously to (4). For both simplices, the generalized barycenters and their probabilities are given by

$$\eta_{\mu_t} = \left[q(\eta_{\mu_t})\right]^{-1} \cdot \sum_{\nu_t=0}^{K_t} u_{\nu_t} \int \lambda_{\nu_t}(\eta_t)\tau_{\mu_t}(\xi_t)\,\mathrm{d}P_t,$$

$$q(\eta_{\mu_t}) = \int \tau_{\mu_t}(\xi_t)\,\mathrm{d}P_t, \quad \mu_t = 0,\dots,L_t,$$

$$\xi_{\nu_t} = \left[q(\xi_{\nu_t})\right]^{-1} \cdot \sum_{\mu_t=0}^{L_t} v_{\mu_t} \int \lambda_{\nu_t}(\eta_t)\tau_{\mu_t}(\xi_t)\,\mathrm{d}P_t,$$

$$q(\xi_{\nu_t}) = \int \lambda_{\nu_t}(\eta_t)\,\mathrm{d}P_t, \quad \nu_t = 0,\dots,K_t.$$

Note that the integrand $\lambda_{\mu t}(\eta_t)\cdot\tau_{\nu_t}(\xi_t)$ is a bilinear function in (η_t, ξ_t) since the barycentric weights $\lambda_{\mu t}$ and τ_{ν_t} are linear in their components. Obviously, a bilinear function is easy to integrate which was the intention of the approximation.

The generalized barycenters ξ_{ν_t}, $\nu_t = 0, \dots, K_t$, are supporting points of the minorant. They are combined with the vertices u_{ν_t} and weighted with the corresponding probabilities $q(\xi_{\nu_t})$ to obtain discrete outcomes for the lower approximation of the original measure P_t. This way, one derives a discrete probability measure Q_t^l with support

$$\operatorname{supp} Q_t^l = \{(u_{\nu_t}, \xi_{\nu_t}):\ \nu_t = 0,\dots,K_t\}.$$

Analogously, $\eta_{\mu t}$, $\mu_t = 0, \dots, L_t$, are supporting points for the majorant with assigned probabilities $q(\eta_{\mu t})$. This induces a discrete measure Q_t^u for the upper approximation with

$$\operatorname{supp} Q_t^u = \left\{(\eta_{\mu_t}, v_{\mu_t}):\ \mu_t = 0,\dots,L_t\right\}.$$

Both measures represent the solutions of two corresponding moment problems. The advantageous feature from a computational viewpoint is that the generalized barycenters and their probabilities are completely determined by the first moments of η_t and ξ_t, and by the bilinear cross moments $E(\eta_{\nu_t}\cdot\xi_{\mu t})$, $\nu_t = 0, \dots, K_t$, $\mu_t = 0, \dots, L_t$. Note that the covariance of two random variables is derived from the first moments and the corresponding cross moments. Therefore, the measures Q_t^u and Q_t^l incorporate implicitly a correlation between η_t and ξ_t. However, cross moments (or covariances, respectively) between different elements of η_t are not taken into account (the same holds for the components of ξ_t). Hence, the formulae given above are applicable without the assumption of independent random variables.

Applying the approximation scheme dynamically over time, one obtains two *barycentric scenario trees* $\mathcal{A}^l$ and $\mathcal{A}^u$ with their path probabilities of type (3). The set of outcomes at stage $t = 1, \ldots, T$ is given by $\mathcal{A}^l(\eta^{t-1}, \xi^{t-1}) = \text{supp}\ Q_t^l(\cdot|\eta^{t-1}, \xi^{t-1})$ and $\mathcal{A}^u(\eta^{t-1}, \xi^{t-1}) = \text{supp}\ Q_t^u(\cdot|\eta^{t-1}, \xi^{t-1})$. Substituting P_t in (1) by the discrete measures Q_t^l and Q_t^u yields two value functions

$$\psi_t(x^{t-1}, \eta^t, \xi^t) := \min \Big\{ \rho_t(x^{t-1}, x_t, \eta^t) + \int \psi_{t+1}(x^{t-1}, x_t, \eta^t, \eta_{t+1}, \xi^t, \xi_{t+1})\, \mathrm{d}Q_{t+1}^l \Big\},$$

$$\Psi_t(x^{t-1}, \eta^t, \xi^t) := \min \Big\{ \rho_t(x^{t-1}, x_t, \eta^t) + \int \Psi_{t+1}(x^{t-1}, x_t, \eta^t, \eta_{t+1}, \xi^t, \xi_{t+1})\, \mathrm{d}Q_{t+1}^u \Big\}$$

for $t = 0, \ldots, T$ with $\psi^{T+1}(\cdot) = \Psi^{T+1}(\cdot) := 0$. According to [4], these are lower and upper bounds to the original value function, i. e.

$$\psi_t(x^{t-1}, \eta^t, \xi^t) \le \phi_t(x^{t-1}, \eta^t, \xi^t) \le \Psi_t(x^{t-1}, \eta^t, \xi^t).$$

In the entire convex case, the accuracy of the approximation is quantifiable by the difference between the upper and lower bound. If required, the approximation can be improved by partitioning the simplices Θ_t and Ξ_t. In case that the subsimplices become arbitrarily small, the extremal measures converge to P_t, and the convergence of the upper and lower bounds to the expectation of the value function is guaranteed (see [5] for details).

See also

- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Two-stage Stochastic Programs with Recourse
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ General Moment Optimization Problems
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Minimax Approach
- ▶ Simple Recourse Problem: Primal Method
- ▶ Simple Recourse Problem: Dual Method
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Integer Programs
- ▶ L-shaped Method for Two-stage Stochastic Programs with Recourse
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Two-stage Stochastic Programming: Quasigradient Method
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Preprocessing in Stochastic Programming
- ▶ Stochastic Network Problems: Massively Parallel Solution

References

1. Birge JR, Donohue CJ, Holmes DF, Svintsitski OG (1996) A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs. Math Program 75:327–352
2. Edmundson HP (1957) Bounds on the expectation of a convex function of a random variable. Techn Report RAND Corp, 982
3. Frauendorfer K (1992) Stochastic two-stage programming. Springer, Berlin
4. Frauendorfer K (1994) Multistage stochastic programming: Error analysis for the convex case. Math Meth Oper Res 39:93–122

5. Frauendorfer K (1996) Barycentric scenario trees in convex multistage stochastic programming. Math Program 75:277–293
6. Jensen JL (1906) Sur les fonctions convexes et les inégalitées entre les valeurs moyennes. Acta Math 30:175–193
7. Kall P (1998) Bounds for and approximations to stochastic linear programs with recourse. In: Marti K, Kall P (eds) Stochastic Programming Methods and Technical Applications. Springer, Berlin, pp 1–21
8. Madansky A (1959) Bounds on the expectation of a convex function of a multivariate random variable. Ann Math Statist 30:743–746
9. Mulvey JM, Ruszczyński A (1995) A new scenario decomposition method for large-scale stochastic optimization. Oper Res 41:477–490
10. Rosa CH, Ruszczyński A (1996) On augmented Lagrangian decomposition methods for multistage stochastic programs. Ann Oper Res 64:289–309