C

CAPACITATED MINIMUM SPANNING TREES

The *capacitated minimum spanning tree problem* (CMST) or *terminal layout problem* is usually described as the problem of determining a rooted spanning tree of minimum cost in which each of the subtrees off the root node contains at most K nodes. That is, the CMST is a generalization of the well-known *minimum spanning tree problem* (MST) where the objective is to find a minimum cost tree spanning a given set of nodes such that some capacity constraints are observed.

As a graph theoretic problem we consider a connected graph $G = (V, A, b, c)$ with node set $V = \{0, \ldots, n\}$ and arc set A. Each node $i \in V$ has a nonnegative node weight b_i which may be interpreted as capacity requirement whereas a nonnegative arc weight c_{ij} represents the cost of using arc $(i, j) \in A$. Node 0 denoted as the *center node* will be the root of the tree (with $b_0 := 0$). We define a *subtree* or *component* C_i of a tree spanning V as its maximal subgraph uniquely connected to the center by arc $(0, i)$ (denoted as *central arc*). The demand of a subtree is the sum of the node weights of the included nodes. To satisfy the *capacity constraint* the demand of each subtree must not exceed a given capacity K. (Without loss of generality we may assume $b_i \leq K$ for all i.) By means of these definitions the CMST is the problem of finding a minimum cost tree spanning node set V where all subtrees satisfy the capacity constraint.

In spite of existing polynomial algorithms for the unconstrained MST the CMST has been shown to be *NP*-hard [32] even when all b_i-values are identical. This case of the CMST is referred to as *unit weight CMST* or *equal demand* CMST; otherwise it is called the *nonunit weight* case. Most references in the literature deal with the unit weight case with only a few exceptions treating the more general case. For a comprehensive survey of the (unit weight) CMST up to the mid-1990s see [4]. The CMST in undirected graphs requires a symmetric cost matrix. Otherwise, the direction of the arcs has to be considered, i.e., all subtrees are directed. This *capacitated minimum spanning arborescence problem* (CMDT) includes the CMST as a special case.

Motivated by the intractability of the problem both heuristic as well as exact algorithms have been developed. In the sequel various algorithmic concepts are reviewed mainly for the unit weight CMST (with special emphasis on progress made in the late nineties; for some older yet important references not given here see [4]). However, first we sketch some applications of the CMST some of which may also lead to important modifications of the problem.

Applications. The CMST has a great variety of applications especially in the field of telecommunications network design. For instance, in the design of minimum cost teleprocessing networks terminals (nodes) have to be connected to a central facility (the center node) by so-called multipoint lines (the subtrees) which have to be restricted with respect to the traffic transfered between the center and the included terminals or the number of terminals included in the line. The latter is sometimes called reliability constraint because it limits the maximal number of terminals disconnected from the central

facility in the case of a single link breakdown. Although different constraints may be referred to as capacity constraints (e.g. considering arc weights instead of node weights or even nonlinear weight functions depending on the distance of a node or arc from the center) most formulations in the literature consider only one of them.

Mathematical Programming Formulations. For the CMST a great variety of formulations may be found in the literature; see, e.g., [14], [19], [20], [21], [22]. Here we restrict ourselves to the presentation of a well-known flow-based formulation. As relaxations of directed formulations may be advantageous we consider the CMDT.

Assume $b_i = 1$ for all $i = 1, \ldots, n$, and $b_0 = 0$, then the CMDT can be described as a mixed integer linear programming formulation as follows. Define $x_{ij} = 1$, if arc (i, j) is included in the solution, and $x_{ij} = 0$, otherwise. Furthermore, let y_{ij} denote the flow on arc (i, j) for all i, j, i.e., $i = 0, \ldots, n$ and $j = 1, \ldots, n$. Ensure variables x_{ij} and y_{ij} with $(i, j) \notin A$ to be equal to zero by assigning prohibitively large weights to them. The following single-commodity flow formulation gives a minimum cost directed capacitated spanning tree with center node 0 as the root:

$$
\text{(P)} \quad \left\{ \begin{array}{ll}
\min & \displaystyle\sum_{i=0}^{n} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \\
\text{s.t.} & \displaystyle\sum_{i=0}^{n} x_{ij} = 1, \quad j = 1, \ldots, n, \\
& \displaystyle\sum_{i=0}^{n} y_{ij} - \sum_{i=1}^{n} y_{ji} = 1, \\
& \quad j = 1, \ldots, n, \\
& x_{ij} \leq y_{ij} \leq (K - b_i) \cdot x_{ij} \\
& \quad \text{for all } i, j, \\
& x_{ij} \in \{0, 1\}, \quad y_{ij} \geq 0 \quad \text{for all } i, j.
\end{array} \right.
$$

The first set of equalities ensures that exactly one arc is reaching each noncentral node. The coupling constraints in combination with the flow conservation ensure that no cycles are allowed and that the capacity constraint is satisfied in each subtree. For a formal proof of cycle prevention see [14], i.e., a tree spanning all nodes is guaranteed.

Exact Algorithms. Most exact algorithms for solving the CMST are based on the branch and bound or the branch and cut paradigm, while other approaches are usually not competitive due to time and space complexity (e.g. dynamic programming [23]).

When describing the concepts from the literature in most cases we do not report computational experiments as there is no fair comparison. When reporting problem sizes solved to optimality by a specific algorithm different authors have proposed various ways of conducting experiments (e.g. the way of data generation), i.e., comparability is not always guaranteed. Moreover, problem instances with a larger number of nodes might be easier to solve than those with a smaller number of nodes, depending on the respective values of K [20], [24]. Another aspect which seems to have considerable impact on the performance of most algorithms is the location of the root. For instance, instances with the root in the center of a rectangle in the Euclidean plane may be solved very easily compared to instances with the root in the corner of the rectangle. Recently, a set of problem instances with 40 and 80 nodes became used consistently (see, e.g., [20], [22]).

Branch and bound methods for the CMST can be divided into two classes. *Node oriented methods* branch by fixing nodes, *arc oriented methods* branch by including an arc (i, j) into the solution (i.e. fixing $x_{ij} = 1$) or excluding it from the solution ($x_{ij} = 0$). A node is called *established* if the path from the root to this node only consists of arcs fixed to 1 (these arcs are called established, too). Usually, only those arcs incident with exactly one established node are allowed to be fixed. If an arc is fixed to 1, it becomes established and both of its end nodes are established. Correspondingly, an arc is called *disallowed*, if $x_{ij} = 0$ is fixed.

A well-known relaxation of the CMST is the MST relaxation which can be easily solved to optimality. If the MST solution is feasible for the CMST, then it is also optimal and the respective problem can be fathomed.

In the early 1970s an arc oriented branch and bound algorithm based on the MST relaxation was very popular. Subproblems are, e.g., branched by

defining the first not yet established arc of an infeasible subtree (the first counted from the center) as established or disallowed, respectively [8]. This approach may be improved by using logical tests and tighter lower bounds [10]. Let node i be established, then in a subproblem disallowing arc (i, j) all arcs (i, k) with node k being an established node of the same subtree as j may be disallowed, too, without loosing optimality. If an optimal solution is lost by disallowing these arcs, the complementary subproblem with established arc (i, j) contains another optimal solution. A dominance criterion is used to fathom some subproblems. In addition, the lower bounds are improved using a special case of the degree constrained MST considering that the degree of the center node — and hence the number of subtrees — is greater or equal to the ratio of the total demand and the capacity K of each subtree.

A. Kershenbaum and R.R. Boorstyn [27] propose two branch and bound algorithms both using last-in first-out to choose the subproblem that is next to be considered. One of the algorithms is node oriented. It starts with n subtrees and each node being 'permissible' for each subtree. A subproblem is branched by including or excluding a node from a specific subtree. Lower bounds are obtained from a partitioning algorithm. The node weights used in this algorithm are originally derived from the MST solution and then, during the branch and bound, transformed in a weight exchange process. Theoretically, these bounds are at least as good as those from the MST relaxation, in practice they are much better. With the same partitioning technique an arc oriented branch and bound algorithm similar to the one of [8] is developed.

B. Gavish [14] compares several relaxations of the CMST with respect to lower and upper bounds. Best results are obtained with a Lagrangian relaxation with an additional degree constraint combined with a subgradient optimization procedure.

Outperforming his previous methods, Gavish [15] develops a new binary programming formulation for the CMST based on an extension of the subtour elimination constraints known from the traveling salesman problem (TSP). Because of the large number of these constraints involved in the formulation an augmented Lagrangian procedure is developed where a dual ascent algorithm is used to obtain initial multipliers and a subgradient procedure to optimize them.

L. Gouveia [20] presents a flow formulation with binary variables z_{ijq} being 1 if a flow of q units goes through arc (i, j). Instead of the $O(n^2)$ constraints of the above flow formulation (P) only $O(n)$ constraints are required. The linear relaxation of the new formulation yields lower bounds as good as those produced by the original formulation. With additional constraints different Lagrangian relaxation schemes are obtained that yield some improvements on the bounds of Gavish [15], especially for problem instances with small capacity K and the center in the 'corner' of a rectangle containing the nodes.

K. Malik and G. Yu [30] present another branch and bound algorithm with Lagrangian subgradient optimization. They give a formulation for the CMST (closely related to the one of [15]) and additional tightening constraints which are added to the problem during the optimization process. Based on a multicommodity flow formulation R. Kawatra [26] uses a Lagrangian approach, too.

L. Hall [24] reports on experience with a cutting plane algorithm for instances with up to 200 nodes making clever use of polyhedral methods. Gouveia and P. Martins [22] propose a hop-indexed generalization of formulation (P). Further improvements on the lower bounds for problem instances with the root in the corner of a rectangle are obtained.

P. Toth and D. Vigo [38] provide an exact algorithm for the CMDT and numerical results are also provided for problem instances with up to 200 nodes. Their approach uses an additive lower bounding procedure combining a Lagrangian lower bound and a lower bound based on solving minimum cost flow problems.

Heuristics. Before presenting heuristics for the CMST it is useful to consider the characteristics of feasible and infeasible solutions [4]. A solution consists of a set of components $C_i = (V_i, A_i)$ with node set V_i and arc set A_i where usually C_i is a spanning tree for V_i. Each component includes only one central arc so that two different node sets

V_i and V_j may have the center node as the only common node. Joining all node sets V_i would yield the entire node set V.

A component is called *feasible* if it does not violate the capacity constraint, and *infeasible*, otherwise. It is referred to as *central* if it includes the center node and *noncentral*, otherwise (i.e. a noncentral component results from a component by eliminating the central arc and the center node). Sets of components having both infeasible and noncentral components are not considered as a solution.

A solution is called *feasible*, if every component contained in the solution is central and feasible itself. It is *incomplete*, if every component is feasible but at least one is noncentral. If all components are central but at least one is not feasible then a solution is called *infeasible*.

The following special solutions of the CMST may be emphasized. The incomplete solution with $n+1$ components $C_i = (\{i\}, \emptyset)$, $i = 0, \ldots, n$, is called an *empty tree*. All components are feasible and all except C_0 are noncentral. The feasible solution with n components $C_i = (\{0, i\}, \{(0, i)\})$, $i = 1, \ldots, n$, is called a *star*. All components are central and feasible. In the case of a sparse graph with only a subset of nodes being directly connected to the center artificial arcs with high cost values should be introduced to complete the graph. The star then might be feasible only for the modified problem.

Finding Initial Feasible Solutions. Most procedures for determining initial feasible solutions (start procedures) for the CMST may be classified as *construction procedures*, *savings procedures* or *dual procedures*.

Construction Methods. Construction methods start with an incomplete solution, usually the empty tree, and successively enlarge it until the solution is feasible. Most procedures in this category replace two components and the chosen arc that connects them by a new component. We may distinguish between arc oriented and node oriented methods.

Arc oriented (or *best arc*) procedures choose in a greedy fashion arcs which are used to join its two incident components. The procedures stop when a feasible solution is obtained. The components of the final solution generally are not built one by one but simultaneously. It is not necessary to finish one component before starting another one.

As examples one may use the basic principle of well-known MST algorithms. The *modified Kruskal algorithm* [7] in each iteration chooses a feasible arc with lowest cost and joins the two corresponding components. All arcs that have become infeasible in this step are removed from consideration for the next iterations. Correspondingly, the *modified Prim algorithm* in each iteration chooses an arc with minimal cost which is incident to the center or a central component (with not yet exhausted capacity).

Node oriented (or *best node*) procedures choose in a greedy fashion a node or component and join it to its nearest neighbor component by the best possible arc incident to the chosen component, while preserving feasibility.

An obvious idea is to cluster the nodes into groups of no more than K nodes and then to choose the arc set according to an MST for the nodes of each group and the center. Assuming that coordinates of the nodes are given this approach may be referred to as *clustering* (or *sweep*) algorithm [36].

The *Martin algorithm* [25], [31] chooses the component which is most distant to the center and joins it to its nearest feasible neighbor component. If χ_i is the cost of connecting component C_i to the center, the component with maximal χ_i is chosen.

The *regret method* (or *Vogel approximation method*, VAM, [8]) computes for every component C_i a regret $r_i = a_2(i) - a_1(i)$ that has to be accepted if C_i is not joined to its nearest feasible neighbor component with cost $a_1(i)$ but to its second nearest feasible neighbor with cost $a_2(i)$. The component with maximal regret is chosen and joined to its nearest neighbor. The regrets are recomputed and the procedure continues until the solution is feasible.

Mixed procedures combine arc and node aspects. They assign a weight w_i to each node i and compute for every arc (i, j) the trade-off function value as $t_{ij} = w_i - c_{ij}$. The feasible arc with largest t_{ij} is

chosen and the respective components are joined. In general, the weights have to be updated after each iteration. With an appropriate definition of the weight function and an update rule all preceding heuristics except the clustering procedures can be incorporated in this concept [29]:

- The Kruskal algorithm is obtained for $w_i = 0$ for all i. Obviously, no update is needed.
- For the Prim algorithm assign weights $w_i = 0$ to all central components and $w_i = -\infty$ to all other components. If a noncentral component is joined with a central component, the weight of the new component is set to zero.
- The Martin algorithm requires $w_i = \chi_i + a_1(i)$, and with $w_i = a_2(i) = r_i + a_1(i)$ one obtains the VAM. The weights w_i have to be recomputed if the values of $a_1(i)$ or $a_2(i)$ have changed, respectively.

Mixed VAM is a combined regret-best arc procedure [18], [39]. The regret r_i is used as node weight w_i and thus, the trade-off function is $t_{ij} = r_i - c_{ij}$.

The *unified algorithm* [29] proposes a parameterization of the weight function and the trade-off function.

Savings Procedures. Savings procedures for the CMST usually start with the star. The best feasible change, i.e. the change which yields the largest savings, is performed. This is iteratively repeated until no savings can be obtained any more. The methods could easily be applied to other feasible solutions and so they could be classified as improvement procedures, too.

The *Esau-Williams algorithm* (EW, [11]) joins the two components which yield the maximal savings in cost. The savings s_{ij} of joining C_i and C_j is defined as $s_{ij} = \max\{\chi_i, \chi_j\} - c^*_{ij}$ if joining of C_i and C_j is feasible, and $s_{ij} = \infty$, otherwise, with χ_i again being the minimal cost of the connection from the center to the nodes of C_i and c^*_{ij} being the minimal cost of an arc connecting C_i and C_j. Then all savings concerning the new component have to be recomputed and again the maximal savings is chosen. The process is stopped if no more positive savings are available.

The EW is closely related to the above mentioned best node procedures. For instance, the Martin algorithm may be referred to as a less greedy version of the EW.

The EW can also be described as a special case of the unified algorithm starting with the empty tree and at each step adding a feasible arc with maximal trade-off $t_{ij} = \chi_i - c_{ij}$.

Whitney's savings heuristic [10], [39] modifies the EW by allowing noncentral arcs to be deleted as well as central arcs. This leads to a possible recombination of segments of the components. Here we see again that savings algorithms are closely related to the class of improvement procedures.

The *parallel savings algorithm* (PSA) [18] computes savings like the EW. However, one iteration does not only join one pair of components but a set of pairs with maximal total savings. This set is determined by solving a maximum weight matching (maximal with respect to the savings) in an adequate graph.

To avoid the parallel construction of nearly equal sized components which cannot be joined any longer if they exceed half of the capacity, Gavish [16] proposes consideration of dummy nodes which yield high savings for any component joined with them. Thus, in this *PSA with dummy nodes* the number of joins between original components in one iteration is reduced by half the number of dummy nodes.

Dual Procedures. Dual procedures start with an infeasible low cost solution, usually the MST solution. The *violation of the constraint(s)* is iteratively reduced at the expense of a total cost increase until the solution becomes feasible.

The start procedure of D. Elias and M.J. Ferguson [10] examines every arc (i, j) of any infeasible component. If (i, j) is deleted the resulting noncentral component C_k is connected to another central component by arc (k, l). This arc is chosen such that the total capacity overflow is reduced. Ties are broken such that the smallest cost increase is chosen. The procedure deletes that arc (i, j) which leads to minimal total cost increase $c_{kl} - c_{ij}$ and adds (k, l) to the solution. (As a modification the arc (i, j) with minimal ratio of cost increase and capacity overflow reduction could be chosen.) Given integer cost weights, the procedure terminates with a feasible solution after a finite

number of iterations, because in each iteration the total capacity overflow is reduced by at least one unit.

Given a feasible solution one can try to improve the solution by recombining segments of the components in a similar way [10]. To increase flexibility, consider a slight modification: Exchanging two arcs should be allowed even if it leads to an increase of total capacity overflow whenever the cost of the solution does not increase and the arc that is to be included never had been in the solution before.

Dual procedures may well be related to other concepts. For instance, a dual procedure may be seen as a constructive savings procedure starting within the infeasible region of the solution space. In that sense it might be related to metastrategies as, e.g., tabu search described below in the sense that it performs a recover phase within a strategic oscillation approach.

Additional Procedures. Besides classifying construction, savings or dual procedures, there are procedures using aggregation and decomposition techniques combined with dynamic programming. In addition, some heuristics which start with generating a TSP tour are not considered in that scheme.

Gouveia and J. Paixão [23] present two heuristics for the CMST which are based on problem size reduction by aggregation and decomposition techniques. In the *aggregation heuristic* the nodes are clustered using the EW — thus forming new nodes with higher and in general nonidentical weights — until the resulting aggregated problem is small enough to be solved to optimality in time limits deemed practical. The *decomposition heuristic* creates for each central arc of the MST solution a subproblem by considering only the nodes of the respective subtree. Subproblems which are small enough are solved to optimality. For the remaining subproblems the aggregation heuristic is used.

Note that the above mentioned sweep algorithm might be classified as aggregation procedure, too.

For the case of unit weights, K. Altinkemer and Gavish [2] provide a modified PSA with a worst-case error bound of $3 - 2/K$ and derive a bound of 4 for the case of nonunit weights. First, a TSP tour is constructed and then it is partitioned into feasible subtrees by adding some central arcs and removing respective noncentral arcs. Note that the (noncenter) nodes of the resulting subtrees are always connected in the same order as in the TSP tour.

In the case of unit weights a *K-iterated tour partitioning algorithm* is used: K solutions are constructed. In each solution the first subtree starts with the first (noncenter) node of the TSP-tour, the second subtree starts with node 2 (first solution), node 3 (second solution), ..., node $K + 1$ (Kth solution). Apart from the first and the last each subtree contains exactly K nodes. The best out of these K solutions is chosen.

For nonunit weights a *nearest insertion optimal partitioning algorithm* may be applied: In the nearest insertion tour the nodes are renumbered according to their position. Modified costs c'_{ij} are computed as the cost of a tree linking the center with node i, node i with node $i+1$, etc., and node $j-1$ with node j. If such a tree is infeasible (due to capacity), the respective cost is set to infinity. With these definitions, the shortest path with respect to c'_{ij} from the center to node n represents the optimal partitioning.

In each procedure a final step can be added: The solution is improved by computing MSTs for the derived components.

Improvement Procedures. Improvement procedures for the CMST can be classified as either *local exchange procedures* or *second order procedures*.

Neighborhood Definition. Local exchange procedures start with a feasible solution and seek to improve it by modifying the current solution in a prespecified way: Sets of arcs are included in or excluded from the solution. If more than one change of the solution is possible the best one (with respect to cost) is chosen. The procedure continues as long as improvements are possible.

Given a feasible solution, H. Frank et al. [13] examine for every node i the following exchange: Connect i to its nearest neighbor not yet connected to i and remove the arc with highest cost from the resulting cycle while still preserving feasibility. The exchange with greatest cost decrease is chosen as

long as improvements are positive. The authors describe this procedure for a network design problem with variable arc capacities and cost but it can be naturally applied to the special case with only one available capacity and fixed cost for each arc as in the CMST.

Elias and Ferguson [10] try to improve the solution by recombining segments of the components, i.e., deleting an arc and reconnecting the resulting noncentral component without loosing feasibility (cf. the Whitney savings heuristic above).

The previously reported improvement procedures alter a current solution by including or excluding arcs. In contrast, a node exchange procedure transforms one feasible solution to a neighbor solution by changing the assignment of the nodes to the subtrees. Such a transformation is called *move*. Subsequently a certain number of moves is performed thus trying to find improved solutions.

Starting from the EW solution, in their CMST procedure A. Amberg et al. [4] consider two types of moves: *Shift moves* choose one node and shift it from its actual component to another one. *Exchange moves* choose two nodes belonging to different subtrees and exchange them. Both types may be simultaneously used whereas only *feasible moves* are allowed, i.e. those leading again to feasible solutions.

A modified neighborhood definition involves cutting a subtree from a given solution and to paste it within another subtree or to connect it to the root node [35]. Additional neighborhood structures are given in [1]. Contrary to the previous neighborhood structures the authors do not restrict themselves to the consideration of two subtrees to be involved in one move but into a chain of moves performed simultaneously (called cyclic exchanges and path exchanges). That is, the number of exchanges grows exponentially with the problem size. Based on a shortest path algorithm some profitable exchanges may be determined in way which may be termed *Lin–Kernighan neighborhood* or *ejection chain*.

Second Order Algorithms. Second order algorithms iteratively apply a slave procedure to different start solutions (where some arcs are fixed to be included) and/or modified cost matrices (where inhibitively high cost has been assigned to some arcs) thus forcing arcs into or out of the solution. Savings procedures as the EW or the PSA are applied as slave procedures to complete the solution. In each iteration, all possible modifications according to a given rule are checked. The best one is realized and the respective modifications are made permanent for the remaining iterations. Two important second order algorithms are inhibit and join [25].

The *inhibit procedure* examines for every arc of the current solution the effect of excluding this arc by applying the EW to a modified graph where the cost of the respective arc has been made inhibitively high. The inhibition yielding the lowest cost solution is made permanent (the arc is inhibited for the remaining iterations) and the process is repeated until no further cost reduction can be obtained. At most $O(n^2)$ iterations, each with at most $O(n)$ inhibitions, have to be considered.

The *join procedure* determines for every node i its nearest neighbor i_1 as well as the closest neighbor i_2 closer to the center than i (if different from i_1). It computes the effect on the cost of the solution if node i is directly connected to node i_1 or alternatively to node i_2 (if this is not already done in the actual solution) by applying the start procedure on a modified graph. The joining which produces the best solution is made permanent and the procedure is repeated with this solution. In each of the $O(n)$ iterations $O(n)$ joins have to be considered.

It should be noted that both procedures, inhibit and join, are already look ahead procedures (trying to overcome a shortsighted myopic behavior). Both improvement procedures can be used alone or in combination with each other performing one iteration of join after an iteration of inhibit and vice versa. Combining the procedures restricts the number of iterations to $O(n)$ (from the join procedure) yielding a complexity of $O(n^2)$ times the EW complexity.

For the improvement procedure of [28] in a first step the MST solution and the EW solution have to be determined. Then the following iteration is performed. Define T as the set of arcs which are in the MST but not in the EW solution. For every nonempty subset S of T generate a (incom-

plete) solution including these arcs (if this is feasible), then exclude all arcs of the remaining subset $T\backslash S$ (by modifying the respective arc costs) and complete the solution by applying the heuristic. Choose the subset S^* which yields the largest improvement and permanently include these arcs into the solution. Repeat this iteration with modified $T := T\backslash S^*$.

The *min-exchange heuristic* outlined in [17] starts with any given feasible solution and determines for every pair of components C_p and C_q the cheapest arc (i, j) connecting the two components. All arcs incident to i or j are deleted. C_p and C_q are decomposed into two noncentral single node components C_i and C_j and some remaining components. Now the noncentral components are connected with the center; hereby the minimal cost arcs are chosen. The PSA completes this modified solution. The authors propose to split all components simultaneously.

Computational Results. In the early CMST literature the EW has been found to perform best on average when compared to procedures with similar computation times. Therefore, even nowadays EW is taken as a benchmark to check the performance of other procedures. Kershenbaum and W. Chou [29] report that the unified algorithm running with 3 to 10 different parameter combinations and correspondingly multiplied computation times yields 1–5% improvement over EW. Unfortunately, no specific parameter combination produces improvements in general.

Gouveia and Paixão [23] admit the nearest insertion optimal partitioning algorithm to perform much worse than EW on average with some significant exceptions. This shows that no general dominance of EW considering single problem instances can be derived.

Gavish and Altinkemer [18], [16] report for test problems with up to 400 nodes that the PSA yields improvements of 2–4% in the unit weight case, but performs poorly for nonidentical weights. In the latter case, the min-exchange heuristic applied to the PSA solution gives results comparable to those of EW [17]. Gavish [16] reports that the PSA with dummy nodes attains improvements over EW (up to 6% some cases). However, in the nonunit weight case EW performs still better. Here, the PSA with constant number of joins gives consistently better results than EW. Gouveia and Paixão [23] apply this variant of the PSA with the number of joins varying between 1 (which is in fact the EW) and 12 on unit weight test problems with up to 200 nodes: Significant improvements over EW with computation times raised by a factor of up to 250 are obtained. They also report that the (original) PSA performs best when the capacity is a power of 2 (in the unit weight case). Their aggregation heuristic on average yields a slight improvement over EW (up to 3% in some cases). In test problems, that have the center in the 'middle' of the rectangle containing the nodes, the decomposition procedure has larger computation times than the aggregation algorithm (factors of slightly more than 1 up to 3 are found) and better results, whereas in cases with the center on the 'corner' of the rectangle both methods in almost all instances have similar running times and solutions. Apart from a few cases the PSA with constant number of joins (and varied parameters) on average performs better than both procedures.

M. Karnaugh [25] tests inhibit and join on problems with up to 150 nodes. The combination of the procedures gives 2–3% improvement over EW while the running time is increased by a factor 100 (derived for the 150-node problems). Applying only inhibit performs slightly worse.

Kershenbaum et al. [28] found that inhibit, join and their own procedure yield improvements of around 2% over EW. Thus, their own procedure requiring only 2 to 3 times more computation time than EW, outperforms join.

Metaheuristics. Given a local search mechanism, a metastrategy like tabu search or simulated annealing as a *guiding process* decides which of the possible moves is chosen and forwards its decision to the *application process* which then executes the chosen move. In addition, it provides some information for the guiding process (depending on the requirements of the respective metastrategy) like the recomputed set of possible moves.

Contrary to the improvement procedures reported in the last section, the cost of a new solution may exceed the cost of the previous one.

Moves leading to a cost increase are allowed in order to overcome local optima. Which of the available feasible moves should be chosen to transform the current solution? The answer to this question is not clear and various approaches may lead to good solutions. The guiding process may use, e.g., the two metastrategies simulated annealing and tabu search.

Simulated annealing (SA) randomly chooses one of the feasible moves and its change in cost is computed. If the change is a cost decrease the move is performed. Otherwise, the new solution is accepted with a certain probability. The probability function usually is logarithmic and — intending to favor good solutions — decreases with raising amount of cost increase. It decreases with the number of iterations already performed thus intensifying the search in the current area of the solution space when the execution time is growing. A parameter called *start temperature* has to be specified to adapt the probability function to the actual problem. SA does not require any additional information. If the new solution is rejected the current solution remains unchanged in this step. The next iteration tries again to alter the same solution. Simulated annealing implementations for the CMST are given in [4], [6].

Tabu search (TS) examines all feasible moves. The best move — leading to the highest cost decrease or the lowest cost increase, respectively — is chosen and performed. Now suppose that a local optimum is reached. Without further instructions the procedure could permanently alternate between this local optimum and its best neighbor. For that reason a so-called *tabu list* is created: To prevent that a yet explored solution is examined again, all moves that (could) lead to such a solution are stored in the tabu list. Which moves have to be set tabu is derived from the *running list* (RL) containing all performed moves in their sequence of execution. Both lists have to be updated after each iteration.

In the literature there are several distinct ways of deriving the tabu list. They are referred to, e.g., as static tabu search STS, reverse elimination method REM and cancellation sequence method CSM (see [4] for the CMST). For STS and CSM some parameters have to be specified to adopt the methods to a specific problem and problem instances (especially problem size and scaling of cost).

The storage complexity of the application process is $O(n^2)$ and the time complexity $O(K^2)$ per iteration because of the recomputation of MSTs in the changed components. Using simulated annealing we have a time complexity of the guiding process of $O(K^2)$: To compute the probability of acceptance for the new solution two new subtrees have to be computed. This is part of the application process and need not be performed twice. Thus, additional effort only arises if a solution is rejected which does not influence the overall complexity. The storage complexity also is not raised if simulated annealing is used.

The complexity of the guiding process depends on the special tabu search method. Different tabu search implementations are described in [4], [35]. Whereas [4] seem to provide better results for the benchmark instances with up to 80 nodes than [35], both seem to be outperformed by the more recent (as of 2000) algorithm in [1] based on their more powerful neighborhood structures.

Besides TS and SA additional modern heuristic search concepts have been investigated for the CMST. A neural network approach is investigated in [33]. A GRASP implementation is provided in [34]. The results for both approaches seem to be behind some of those described in the previous paragraphs.

Problem Modifications and Related Problems. Additionally to considering arc costs one may take into account unreliable arcs and node outage costs which are incurred by the user whenever a terminal node is unable to communicate with the central node, i.e., costs associated with link failures [9].

An interesting modification of the CMDT is the resource-constrained minimum spanning tree problem in directed graphs [12]. Here each node, say i, has a certain amount of scarce resources available (a capacity) which may be used to fulfill capacity requirements of all arcs leaving i. Instead of measuring capacity requirements for subgraphs off the root node here the consideration restricts to the set of incident arcs leaving a node. The cur-

rent state of the art for solving this problem circumvents a branch and cut approach [12].

In practice we might be faced with the problem that a solution of the design phase need not be a tree but a forest with more than one root node. Most of the approaches developed for the CMST might be applied in a slightly modified way to this so-called multicenter CMST (see e.g. [3] for an extension of the partitioning heuristics with corresponding worst-case bounds).

When multiple centers are considered in arc oriented vehicle routing then the capacitated arc routing problem (CARP) may be transformed in a way that subproblems are successively solved as CMST. Amberg et al. [5] develop this transformation and apply their TS and SA approaches to this multiple center CARP.

Besides solving the CMST as a pure combinatorial optimization problem it may also be embedded into a problem of users with traffic requirements who have to build contracts with, e.g., a telephone company for the provision of service. This may lead to the consideration of some game-theoretic concepts associated with a cost allocation problem arising from the CMST or more general capacitated network design problems [37].

Conclusions. In this paper we have provided a survey on existing methods for solving the CMST.

With respect to considered algorithmic concepts it might be interesting to incorporate some sort of either exact or heuristic reduction techniques.

See also: **Shortest path tree algorithms**; **Directed tree networks**; **Bottleneck Steiner tree problems**; **Minimax game tree searching**.

References

[1] AHUJA, R.K., ORLIN, J.B., AND SHARMA, D.: 'New neighborhood search structures for the capacitated minimum spanning tree problem', *Techn. Report Sloan School Management, MIT* (1998).

[2] ALTINKEMER, K., AND GAVISH, B.: 'Heuristics with constant error guarantees for the design of tree networks', *Managem. Sci.* **32** (1988), 331–341.

[3] ALTINKEMER, K., AND PIRKUL, H.: 'Heuristics with constant error guarantees for the multi center capacitated minimum spanning tree problem', *J. Inform. Optim. Sci.* **13** (1992), 49–71.

[4] AMBERG, A., DOMSCHKE, W., AND VOSS, S.: 'Capacitated minimum spanning trees: Algorithms using intelligent search', *Combin. Optim.: Theory and Practice* **1** (1996), 9–39.

[5] AMBERG, A., DOMSCHKE, W., AND VOSS, S.: 'Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees', *Europ. J. Oper. Res.* **124** (2000), 360–376.

[6] ANDERSEN, K., VIDAL, R.V.V., AND IVERSEN, V.B.: 'Design of a teleprocessing communication network using simulated annealing', in R.V.V. VIDAL (ed.): *Applied simulated annealing*, Vol. 396 of *Lecture Notes Economics and Math. Systems*, Springer, 1993, pp. 201–215.

[7] BOORSTYN, R.R., AND FRANK, H.: 'Large-scale network topological optimization', *IEEE Trans. Communications* **25** (1977), 29–47.

[8] CHANDY, K.M., AND RUSSELL, R.A.: 'The design of multipoint linkages in a teleprocessing tree network', *IEEE Trans. Computers* **21** (1972), 1062–1066.

[9] DUTTA, A., AND KAWATRA, R.: 'Topological design of a centralized communication network with unreliable links and node outage costs', *Europ. J. Oper. Res.* **77** (1994), 344–356.

[10] ELIAS, D., AND FERGUSON, M.J.: 'Topological design of multipoint teleprocessing networks', *IEEE Trans. Communications* **22** (1974), 1753–1762.

[11] ESAU, L.R., AND WILLIAMS, K.C.: 'On teleprocessing system design', *IBM Systems J.* **5** (1966), 142–147.

[12] FISCHETTI, M., AND VIGO, D.: 'A branch-and-cut algorithm for the resource-constrained minimum-weight arborescence problem', *Networks* **29** (1997), 55–67.

[13] FRANK, H., FRISCH, I.T., SLYKE, R. VAN, AND CHOU, W.S.: 'Optimal design of centralized computer design network', *Networks* **1** (1971), 43–57.

[14] GAVISH, B.: 'Formulations and algorithms for the capacitated minimal directed tree problem', *J. ACM* **30** (1983), 118–132.

[15] GAVISH, B.: 'Augmented Lagrangean based algorithms for centralized network design', *IEEE Trans. Communications* **33** (1985), 1247–1257.

[16] GAVISH, B.: 'Topological design of telecommunication networks – local access design methods', *Ann. Oper. Res.* **33** (1991), 17–71.

[17] GAVISH, B., AND ALTINKEMER, K.: 'A parallel savings heuristic for the topological design of local access tree networks', *Techn. Report Graduate School Management Univ. Rochester, NY* (1986).

[18] GAVISH, B., AND ALTINKEMER, K.: 'Parallel savings heuristics for the topological design of local access tree networks': *Proc. IEEE INFOCOM 86 Conf.*, 1986, pp. 130–139.

[19] GOUVEIA, L.: 'A comparison of directed formulations for the capacitated minimal spanning tree problem', *Telecommunication Systems* **1** (1993), 51–76.

[20] GOUVEIA, L.: 'A $2n$ constraint formulation for the ca-

pacitated minimal spanning tree problem', *Oper. Res.* **43** (1995), 130–141.

[21] GOUVEIA, L., AND HALL, L.: 'A comparative study of network flow formulations for the capacitated spanning tree problem', *Working Paper Fac. Ciencias Univ. Lisboa* **10** (1998).

[22] GOUVEIA, L., AND MARTINS, P.: 'The capacitated minimal spanning tree problem: An experiment with a hop-indexed model', *Ann. Oper. Res.* **86** (1999), 271–294.

[23] GOUVEIA, L., AND PAIXAO, J.: 'Dynamic programming based heuristics for the topological design of local access networks', *Ann. Oper. Res.* **33** (1991), 305–327.

[24] HALL, L.: 'Experience with a cutting plane approach for the capacitated spanning tree problem', *INFORMS J. Comput.* **8** (1996), 219–234.

[25] KARNAUGH, M.: 'A new class of algorithms for multipoint network optimization', *IEEE Trans. Communications* **24** (1976), 500–505.

[26] KAWATRA, R.: 'A multicommodity network flow application for the capacitated minimal spanning tree problem', *Opsearch* **31** (1994), 296–308.

[27] KERSHENBAUM, A., AND BOORSTYN, R.R.: 'Centralized teleprocessing network design', *Networks* **13** (1983), 279–293.

[28] KERSHENBAUM, A., BOORSTYN, R.R., AND OPPENHEIM, R.: 'Second-order greedy algorithms for centralized teleprocessing network design', *IEEE Trans. Communications* **28** (1980), 1835–1838.

[29] KERSHENBAUM, A., AND CHOU, W.: 'A unified algorithm for designing multidrop teleprocessing networks', *IEEE Trans. Communications* **22** (1974), 1762–1772.

[30] MALIK, K., AND YU, G.: 'A branch and bound algorithm for the capacitated minimum spanning tree problem', *Networks* **23** (1993), 525–532.

[31] MARTIN, J.: *Design of real-time computer systems*, Prentice-Hall, 1967.

[32] PAPADIMITRIOU, C.H.: 'The complexity of the capacitated tree problem', *Networks* **8** (1978), 217–230.

[33] PATTERSON, R.A.: 'Hybrid neural networks and network design', *PhD Thesis Ohio State Univ.* (1995).

[34] ROLLAND, E., PATTERSON, R.A., AND PIRKUL, H.: 'Memory adaptive reasoning and greedy assignment techniques for the capacitated minimum spanning tree problem', in S. VOSS, S. MARTELLO, I.H. OSMAN, AND C. ROUCAIROL (eds.): *Meta-heuristics: Advances and trends in local search paradigms for optimization*, Kluwer Acad. Publ., 1999, pp. 487–498.

[35] SHARAIHA, Y.M., GENDREAU, M., LAPORTE, G., AND OSMAN, I.H.: 'A tabu search algorithm for the capacitated shortest spanning tree problem', *Networks* **29** (1997), 161–171.

[36] SHARMA, R.L., AND EL-BARDAI, M.T.: 'Suboptimal communications network synthesis': *Proc. 1970 Internat. Conf. Comm. (19.11-19.16.)*, 1970.

[37] SKORIN-KAPOV, D., AND BELTRAN, H.F.: 'An efficient characterization of some cost allocation solutions associated with capacitated network design problems', *Telecommunication Systems* **3** (1994), 91–107.

[38] TOTH, P., AND VIGO, D.: 'An exact algorithm for the capacitated shortest spanning arborescence', *Ann. Oper. Res.* **61** (1995), 121–141.

[39] WHITNEY, V.K.M.: 'A study of optimal file assignment and communication network configuration in remote access computer message processing and communication systems', *PhD Thesis Univ. Michigan, Ann Arbor* (1970).

Stefan Voß
Inst. Wirtschaftswissenschaften Techn. Univ. Braunschweig
Abt-Jerusalem-Straße 7
D-38106 Braunschweig, Germany
E-mail address: stefan.voss@tu-bs.de

MSC2000: 90C27, 68T99
Key words and phrases: telecommunication, combinatorial optimization, spanning tree, capacitated minimum spanning tree problem, terminal layout problem, resource-constrained minimum spanning tree problem.

CARATHÉODORY, CONSTANTINE

Constantin Carathéodory, a mathematician of Greek origin, was born in Berlin on September 13, 1873 and died on February 2, 1950, in Munich, Germany. He made important contributions to the theory of real functions, to the *calculus of variations*, and to *measure theory*.

He first studied in the Brussels' Military School, where he received a solid mathematical background. After two years as an assistant engineer with the British Asyut Dam project in Egypt, Carathéodory began his study of mathematics at the Univ. of Berlin in 1900, where he attended the courses of L. Fuchs, G. Frobenius and H. Schwarz. He was particularly influenced by Schwarz' lectures with whom he became a close friend. In 1902 he entered the Univ. of Göttingen, where he received his PhD [1] under the German mathematician H. Minkowski. In 1909 he became a full Professor in the Univ. of Hannover. In 1913 he obtained the chair held previously by F. Klein in Göttingen and in 1918 he succeeded Frobenius in the Univ. of Berlin. Then, in 1920, he accepted to help the Greek Government in creating the Univ. of Smyrna, Asia Minor, which then belonged to the Greeks. When the Turks razed Smyrna in 1922, Carathéodory managed to save the university li-

brary, which he moved to the Univ. of Athens, where he taught until 1924. He then was appointed professor of mathematics at the Univ. of Munich.

Carathéodory made important contributions to various branches of mathematics. In the calculus of variations, besides a comprehensive study of discontinuous solutions, which was contained in his PhD thesis, he also added important results linking the theory with first order partial differential equations. His work on the problems of variation of m-dimensional surfaces in an n-dimensional space marked the first far-reaching results for the general case. He also applied the calculus of variations to specific problems of mechanics and physics. He contributed important findings in his book [6]. The theory of functions and measure theory are two additional areas where the work of Carathéodory is very important. His book [3] is a classic of the field. In the theory of functions of several variables he simplified the proof of the main theorem of conformal representation of simply connected regions on the unit-radius circle. His investigations of the geometrical-set theoretic properties of boundaries resulted in his theory of boundary correspondence. Already in 1909 he published a far-reaching paper on the foundations of *thermodynamics* [2]. The paper remained unnoticed by the physicists, because it was published in a mathematical journal. Only in 1921 M. Born brought the paper to the attention of the physics community, and since then the paper and the *Carathéodory principle* became classics. He also contributed to Einstein's special theory of relativity. His published works include [9], [4], [5], [7], [8].

See also: **History of optimization**; **Carathéodory theorem**.

References

[1] CARATHÉODORY, C.: 'On the discontinuous solutions in the calculus of variations', *PhD Thesis* (1904).

[2] CARATHÉODORY, C.: 'Untersuchungen über die Grundlage der Thermodynamik', *Math. Ann.* **67** (1909), 355–386.

[3] CARATHÉODORY, C.: *Vorlesungen über reelle Funktionen*, Teubner, 1918, second ed. 1928. Also: Chelsea Publ., 1948.

[4] CARATHÉODORY, C.: *Conformal representation*, Vol. 28 of *Tracts in Math. and Math. Phys.*, Cambridge Univ. Press, 1932.

[5] CARATHÉODORY, C.: *Variationsrechnung und partielle Differentialgleichungen erster Ordnung*, Teubner, 1935, 2nd ed., 159.

[6] CARATHÉODORY, C.: *Geometrische Optik*, Ergebnisse der Math. und ihre Grenzgeb. Springer, 1937.

[7] CARATHÉODORY, C.: *Reelle Funktionen*, Vol. I, Teubner, 1939.

[8] CARATHÉODORY, C.: *Funktionentheorie*, Vol. 1–2, Birkhäuser, 1950.

[9] CARATHÉODORY, C.: *Mass und Integral und ihre Algebraisierung*, Birkhäuser, 1956.

Nicolas Hadjisavvas
Dept. Math. Univ. Aegean
Karlovassi, Samos, Greece
E-mail address: nhad@aegean.gr
Panos M. Pardalos
Center for Applied Optim.
Dept. Industrial and Systems Engin. Univ. Florida
Gainesville, FL 32611, USA
E-mail address: pardalos@ufl.edu

MSC2000: 01A99
Key words and phrases: Carathéodory, calculus of variations, measure theory.

CARATHÉODORY THEOREM

One of the basic results ([3]) in convexity, with many applications in different fields. In principle it states that every point in the convex hull of a set $S \subset \mathbf{R}^n$ can be represented as a convex combination of a finite number $(n+1)$ of points in the set S. See for example [7], [9], [4], [1], [6], [10]. Generalizations of the theorem can be found in [2] and [5].

THEOREM 1 Let S be any subset of $\mathbf{R}^n$.

For every $x \in \text{conv}(S)$ (the convex hull of S), there exist $n+1$ points $x_0, \ldots, x_n \in S$ such that $x \in \text{conv}(x_0, \ldots, x_n)$. □

PROOF. Since $x \in \text{conv}(S)$, there exists a representation $x = \sum_{i=0}^{k} \alpha_i x_i$, $x_i \in S$ for $i = 0, \ldots, k$ and $\sum_{i=0}^{k} \alpha_i = 1$.

If $k \leq n$, we are finished.

Now suppose $k > n$. Note that then $x_1 - x_0, \ldots, x_k - x_0$ are linearly dependent. There then exist scalars $\lambda_1, \ldots, \lambda_k$, not all zero, such that $\sum_{i=1}^{k} \lambda_i (x_i - x_0) = 0$.

Let now $\lambda_0 = -\sum_{i=1}^{k} \lambda_i$; it then follows that $\sum_{i=0}^{k} \lambda_i x_i = 0$ and we can find at least one $\lambda_i > 0$. So we have,

$$x = \sum_{i=0}^{k} \alpha_i x_i - \gamma \cdot 0$$

$$= \sum_{i=0}^{k} \alpha_i x_i - \gamma \sum_{i=0}^{k} \lambda_i x_i = \sum_{i=0}^{k} (\alpha_i - \gamma\lambda_i) x_i$$

for any $\gamma \in \mathbf{R}$.

Choose γ in the following way:

$$\gamma = \min_{0 \le i \le k} \left\{ \frac{\alpha_i}{\lambda_i} : \lambda_i > 0 \right\} = \frac{\alpha_j}{\lambda_j}$$

for some $j \in \{0, \ldots, k\}$; so, $\alpha_i - \gamma\lambda_i \ge 0$ for all $i = 0, \ldots, k$.

Then we obtain $x = \sum_{i=0}^{k} (\alpha_i - \gamma\lambda_i) x_i$ with $\alpha_i - \gamma\lambda_i \ge 0$ for $i = 0, \ldots, k$, $\sum_{i=0}^{k} (\alpha_i - \gamma\lambda_i) = 1$ and $\alpha_j - \gamma\lambda_j = 0$.

And so x is represented as a convex combination of at most k points in S. We can now repeat these steps until $k = n$. □

See also: **Krein–Milman theorem**; **Linear programming**; **Carathéodory, Constantine**.

References

[1] BAZARAA, M.S., SHERALI, H.D., AND SHETTY, C.M.: *Nonlinear programming*, Wiley, 1993.

[2] BONNICE, W., AND KLEE, V.L.: 'The generation of convex hulls', *Math. Ann.* **152** (1963), 1–29.

[3] CARATHÉODORY, C.: 'Ueber den Variabilitätsbereich der Fourierschen Konstanten von positiven harmonischen Funktionen', *Rend. Circ. Mat. Palermo* **32** (1911), 193–217.

[4] GRÜNBAUM, B.: *Convex polytopes*, Interscience, 1967.

[5] REAY, J.R.: 'Generalizations of a theorem of Carathéodory', *Memoirs Amer. Math. Soc.* **54** (1965).

[6] ROCKAFELLAR, R.T.: *Convex analysis*, Princeton Univ. Press, 1970.

[7] STOER, J., AND WITZGALL, CH.: *Convexity and optimization in finite dimensions I*, Springer, 1970.

[8] VALENTINE, F.A.: *Convex sets*, McGraw–Hill, 1964.

[9] WETS, R.J.-B.: *Grundlagen Konvexer Optimierung*, Vol. 137 of *Lecture Notes Economics and Math. Systems*, Springer, 1976.

[10] ZIEGLER, G.M.: *Lectures on polytopes*, Springer, 1995.

Gabriele E. Danninger-Uchida
Univ. Vienna
Vienna, Austria
E-mail address: gabriele.uchida@univie.ac.at

MSC2000: 90C05

Key words and phrases: polytope, convex hull, representation.

CHECKLIST PARADIGM SEMANTICS FOR FUZZY LOGICS

Why the Checklist Paradigm? The classical logic deals with two logical values — 'truth' and 'falsity'. It can be characterised algebraically and semantically by a *Boolean algebra*. Not all issues of logic can, however, be settled by a system of classical two-valued logic. For example some modalities, such as necessity and possibility cannot, in general, be expressed in any system that admits only a finite number of logical values. Also some temporal logics [33] characterising time require an infinite number of logical values in their semantics.

Fuzzy Logics. Many-valued logic algebras are needed for developing the mathematics of *fuzzy relations* [28] and sets [18]. For example, in order to compute the degree δ to which two fuzzy sets intersect, we use the formula $\delta_{A \cap B}(x) = \delta_A(x) \wedge \delta_B(x)$, where $\wedge$ is a many-valued 'AND' connective and $\delta_A(x)$, $\delta_B(x)$ are some logical values: either truth-values, possibilities, probabilities, etc. Depending on the *epistemological interpretation* of the logical values, we read the statement $\delta(A)(x)$ 'The degree to which it is true that $x \in A$', 'The degree to which it is possible that $x \in A$', 'The degree to which it is probable that $x \in A$', etc.

Computing the *degree of inclusion* of two sets [2] is done by the formula $\delta(A \subseteq B) = (\forall x)\delta_A(x) \to \delta_B(x)$, where x ranges over elements of the universe U from which the elements of A and B are drawn. Here $\to$ is a many-valued *implication operator*.

Approximate Reasoning. Many-valued logic systems are also required for algebraic characterization of logics of *approximate reasoning*. The *premises of an inference* (i.e. the antecedent formulas that form the arguments of the rules of approximate inference) are used by the rules to generate the succedent formulas — the *conclusion*(s).

If each of these logic formulas attains as its logic value a single value from some *lattice*, we speak of a *point-based logic* system of approximate reasoning. If the logic value is a whole interval $[\delta_k, \delta_l]$ such that $\delta_k \le \delta_l$ it is an *interval logic*.

Hence, many-valued logics play a key role in

all the areas of mathematics and logic discussed above. There is not one many-valued logic, there is an infinite number of families of logic systems of various kinds. Hence, according to the purpose of its use, one has to choose an appropriate many-valued system. But even after the choice is made, the two key questions still remain:

- Where the logic values come from?
- Is there any basic epistemic or semantic procedure by which the basic logic connectives can be meaningfully derived?

These questions are answered by the *checklist paradigm*.

Many-Valued Logics in Fuzzy Sets. The theory of fuzzy sets and relations requires a many-valued logic in which to manipulate the degrees of truth which attach to fuzzy statements. As in classical two-valued logic (in which the statements are judged to be either utterly true or utterly false), one wishes a *truth-functional* connection between the truth values assigned to 'p' and to 'q' and those to be assigned to 'p or q' and 'p and q' and 'if p then q', as well as to 'not-p' and 'not-q', that is, one wishes the evaluation of the derived formulas to depend solely on the evaluation of the original formulas, without further reference to their contents.

There are a number of such many-valued logical systems, with truth values in the closed real interval $[0,1]$. Everyone agrees that the values assigned in the crisp 'corners', where the values $|p|$ of p and $|q|$ of q are zero (false) or one (true), must accord with the classical Boolean logic. Most agree in setting

$$|\text{not} - p| = |\neg p| = 1 - |p|$$

and the most usual 'or' and 'and' connectives are given by

$$|p \text{ or } q| = |p \vee q| = \max(p, q),$$
$$|p \text{ and } q| = |p \wedge q| = \min(p, q),$$

although other have been proposed and have something to be said for them.

Selecting max and min as the functions for computing the logical values of the *connectives* $\vee$ and $\wedge$ does not yet determine the system of many-valued logic fully. Indeed, a number of different systems employ these. Third determining factor is the choice the *implication operator* $\rightarrow$. Some frequently used $\rightarrow$ are listed below.

No.	Opr.	Definition
2.	S	Standard Strict $a \overset{2}{\rightarrow} b = \begin{cases} 1, & a \le b \\ 0, & \text{otherwise} \end{cases}$
3.	S^*	Gödel $a \overset{3}{\rightarrow} b = \begin{cases} 1, & a \le b \\ b, & \text{otherwise} \end{cases}$
4.	G43	product ply (also: Goguen–Gaines) $a \overset{4}{\rightarrow} b = \min\left(1, \frac{b}{a}\right)$
4'.	G43'	Modified G43 $a \overset{4'}{\rightarrow} b = \min\left(1, \frac{b}{a}, \frac{1-a}{1-b}\right)$
5.	L	Lukasiewicz $a \overset{5}{\rightarrow} b = \min(1, 1 - a + b)$
5.5.	KDL	Reichenbach $a \overset{5.5}{\rightarrow} b = \min(1, 1 - a + ab)$
6.	KD	Kleene–Dienes $a \overset{6}{\rightarrow} b = (1 - a) \vee b$
7.	EZ	Early Zadeh $a \overset{7}{\rightarrow} b = (a \wedge b) \vee (1 - a)$ $= (a \overset{6}{\rightarrow} b) \wedge ka$ where $ka = (1 - a) \vee a$
8.	W	Willmott $a \overset{8}{\rightarrow} b = (a \overset{7}{\rightarrow} b) \wedge kb$

Table 1: Some important many-valued implication operators.

Not only properties of the many-valued logic systems but also of the systems of *fuzzy sets* crucially depend on the choice of the implication operator. For example both the definition of a *fuzzy power set* (i.e. the set of all subsets) and of the *fuzzy set-inclusion operator* depend on its choice. The very first paper on fuzzy sets by L.A. Zadeh [44], [45] uses max and min connectives to define the intersection $\cap$ and the union $\cup$ of two fuzzy sets. The set inclusion operator Zadeh defines by the formula

$$\mu(A \subseteq B) = 1 \quad \Leftrightarrow \quad (\forall x)\mu_A \le \mu_B(x).$$

Using the 'Standard Strict' $\rightarrow_2$ in the formula given of the first section above we obtain

$$\begin{aligned} &\delta(A \subseteq B) \\ &= (\forall x)\delta_A(x) \rightarrow \delta_B(x) \\ &= (\forall x)\mu_A(x) \overset{2}{\rightarrow} \mu_B(x) \end{aligned}$$

$$= \min_{\{x \in U\}} (\mu_A(x) \overset{2}{\rightarrow} \mu_B(x)).$$

This formula is equivalent to Zadeh's early definition of fuzzy set inclusion which in fact is crisp (nonfuzzy). Power set theories with proper fuzzy set inclusion have been first investigated in [2], [43] (using the *implication operators* listed in the table above).

Since 1965, when the first paper on fuzzy sets was written by Zadeh, not only max and min but also other many-valued logic connectives were used to define the union $\cup$ and the intersection $\cap$ of fuzzy sets. An important pair are the so called 'bold connectives': '$a \wedge_5 b = \max(0, a + b - 1)$' and '$a \vee_5 b = \min(1, a + b)$'. As the subscript indicates, these are related to the *Łukasiewicz implication* operator. These represent the so-called *MV algebras* which play an important role in application of fuzzy sets in quantum logics [35] and elsewhere. Both types of connectives, the pairs 'max-min' and the 'bold' connectives are special instances of the so-called triangular norms (*t-norms*) and conorms (*t-conorms*) [36],[17]. These associative operations with special properties, defined on $[0, 1]$, algebraically characterise the whole infinite family of OR-AND pairs of many-valued logic connectives and play a crucial role in the theory and applications of fuzzy sets.

The Checklist Paradigm. The *checklist paradigm* provides the mechanism by which several types of very different families of many-valued logic connectives *emerge* from some more basic considerations.

- It provides the *semantics* of systems that use single value as its logic value.
- It provides the justification for *interval logics*.
- It provides a link of *many-valued logics* connectives with *generalized quantifiers*.

Mathematics of the Checklist Paradigm. A *checklist template* Q is a finite family of properties $\langle P_1, \ldots, P_n \rangle$. With a template Q, and a given *proposition* A, one can associate a specific checklist $Q_A = \langle Q, A \rangle$. A *valuation* f_A of a checklist Q_A is a function from Q to $\{0, 1\}$.

The value a_Q of the proposition A with respect to a template Q (which is the summarised value of the valuation f_A) is given by the formula

$$a_Q = \sum_{i=1}^{n} p_i^A$$

where $n = \operatorname{card} Q$ and $p_i^A = f_A(P_i)$.

A *fine valuation structure*, a pair of propositions A, B with respect to the template Q, is a function $f_{A,B}^Q$ from Q into $\{0, 1\}$ assigning to each attribute P_i the ordered pair of its values $\langle p^A, p^B \rangle$.

Let $\alpha_{j,k}$ be the cardinality of the set of all attributes P_i such that $f_{A,B}^Q(P_i) = \langle j, k \rangle$.

Obviously we have the following constraint on the values: $\alpha_{00} + \alpha_{01} + \alpha_{10} + \alpha_{11} = n$. Further, we define $r_0 = \alpha_{00} + \alpha_{01}$, $r_1 = \alpha_{10} + \alpha_{11}$, $c_0 = \alpha_{00} + \alpha_{10}$, $c_1 = \alpha_{01} + \alpha_{11}$.

These entities can be displayed systematically in a *contingency table*. In such a table, the inner fine-summarization structure consists of the four $\alpha_{j,k}$ appropriately arranged, and of margins c_0, c_1, r_0, r_1 (see Fig. 1).

	No for B	Yes for B	Row total
No for A	α_{00}	α_{01}	r_0
Yes for A	α_{10}	α_{11}	r_1
Column Total	c_0	c_1	n

Fig. 1: Checklist paradigm of the assignment of fuzzy values. Define: $a = r_1/n$; $b = c_1/n$.

Now let F be any logical propositional function of propositions A and B. For $i, j \in \{0, 1\}$, let $f(i, j)$ be the classical truth value of F for the pair i, j of truth values; let $u(i, j) = \alpha_{i,j}/n$, the ratio of the number in the ij-cell of the constraint table, to the grand total. Then we define the (nontruth-functional) *fuzzy assessment of the truth* of the proposition $F(A, B)$ to be

$$m(F(A, B)) = \sum_{i,j} f(i, j) \cdot u_{ij}.$$

This assessment operator will be called the *contraction/approximation measure*.

The four interior cells $\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11}$ of the constraint table constitute its *fine structure*; the margins r_0, r_1, c_0, c_1 constitute its *coarse structure* (see Fig. 1).

The fine structure gives us the appropriate fuzzy assessments for all propositional functions of A

and B; the coarse structure gives us only the fuzzy assessments of A and B themselves. Our central question is:

> to what extent can the fine structure be reconstructed from the coarse?

As shown elsewhere [3], [5], [6], [8] the coarse structure imposes bounds upon the fine structure, without determining it completely. Hence, associated with the various logical connectives between propositions are their extreme values.

There are four extremes that the fine structure of the contingency table (see Fig. 1) can attain [6], [8]:

i) the two *mindiag fine structures* with the diagonal values minimized ($\alpha_{00} = 0$ or $\alpha_{11} = 0$); and

ii) the two *maxdiag fine structures* with the diagonal values maximized ($\alpha_{01} = 0$ or $\alpha_{10} = 0$).

Thus we obtain the inequality restricting the possible values of $m(F)$:

$$\text{con top} \geq m(F) \geq \text{con bot},$$

where 'con' is the name of connective represented by $f(i,j)$. Choosing for the logical type of the connective 'con' the *implication* and making the assessment of the fuzzy value of the truth of a proposition by the formula $m_1(F) = 1 - u_{10}$ we obtain:

$$\min(1, 1-a+b) \geq m_1(A \to B) \geq \max(1-a, b).$$

We can see that the checklist paradigm generated the *Łukasiewicz implication* operator, and the *Kleene–Dienes implication* operator.

We have already noted that choosing for 'F' the connective type 'AND' [5], [8]) and m_1, we obtain the bounds

$$\min(a,b) \geq m_1(\text{AND}) \geq \max(0, a+b-1).$$

These bounds are formally identical with those of B. Schweitzer and A. Sklar [36] giving the bounds on *copulas* which play an important role in their theory of *t-norms* and *t-conorms*. Surprisingly, these checklist paradigm bounds also coincide with Novák's recent (1991) derivation [31] of bounds on fuzzy sets approximating classes of Vopěnka's alternative set theory [41], [42]. E. Hisdal derives the same inequalities as the bounds on some connectives of her TEE model and comments on a possible link (cf. [16, Appendix A2]). In the context of modalities in fuzzy logics, checklist paradigm-like inequalities for $F = \{\text{AND}, \text{OR}\}$ were recently (1992) also independently discovered in [34]. Yet all these models are neither formally nor epistemologically identical. This indicates the need for a more precise meta- and metametalogical formulation of many-valued based mathematical systems, that would include in their full definition a part formulating their 'mathematical epistemology'.

Interval Inference and The Checklist Paradigm. The checklist paradigm puts ordering on the pairs of distinct implication operators and other pairs of connectives. Hence it provides a theoretical justification of *interval-valued approximate inference*. For the m_1 contraction/approximation measure, there are 16 inequalities linking the *TOP and BOT types of connectives* [5], [8], thus yielding 16 logical types of TOP-BOT pairs of connectives. Ten of these interval pairs generated by m_1 are listed in Table 2.

Logical Type of Connective	Valuation BOTTOM≤TOP
AND $a\&b$	$\max(0, a+b-1) \leq \min(a,b)$
Nicod $a \downarrow b$	$\max(0, 1-a-b) \leq \min(1-a, 1-b)$
Sheffer $a\|b$	$\max(1-a, 1-b) \leq \min(1, 2-a-b)$
OR $a \vee b$	$\max(a,b) \leq \min(1, a+b)$
Nonimplication $a \not\leftarrow b$	$\max(0, b-a) \leq \min(1-a, b)$
Nonimplication $a \not\rightarrow b$	$\max(0, a-b) \leq \min(a, 1-b)$
Implication $a \leftarrow b$	$\max(a, 1-b) \leq \min(1, 1+a-b)$
Implication $a \to b$	$\max(1-a, b) \leq \min(1, 1-a+b)$
Equivalence $a \equiv b$	$\max(1-a-b, a+b-1) \leq \min(1-a+b, 1+a-b)$
Exclusive OR $a \oplus b$	$\max(a-b, b-a) \leq \min(2-a-b, a+b)$

Table 2: Two-argument interval pairs of connectives generated by m_1.

Other Systems of Fuzzy Logic Connectives for Interval Inference. In Boolean (crisp) logic, the values of a logical formula written in the *disjunctive*

normal form (*DNF*) are equal to the values the formula expressed in the *conjunctive normal form* (*CNF*). This does not hold for every system of many-valued connectives.

I.B. Türksen [40], [38], [39] has shown that for $\max(a,b)$, $\min(a,b)$ and some other t-norm and t-conorm based CNFs and DNFs the inequality $\mathrm{DNF}(a\,\mathrm{CON}\,b) \leq \mathrm{CNF}(a\,\mathrm{CON}\,b)$ holds for all 16 basic many-valued connectives CON.

Taking for example the max-min based CNF and DNF, the corresponding implications are given by

$$\mathrm{CNF}(a \to b) = (\neg a \vee b),$$
$$\mathrm{DNF}(a \to b) = (a \wedge b) \vee (\neg p \wedge q) \vee (\neg a \wedge \neg b).$$

For further information on other systems of connectives for *fuzzy interval inference* see [11], [27].

Optimization of Interval Inference. Formulas that are equivalent logically [6], [27] may not be equivalent when compared by their formula complexity. This is well-known phenomenon when expressing logical formulas in DNF or CNF normal forms [19]. The same logical function expressed in one of these forms may have more complicated expression in the other normal form. Similarly, this can be observed with other logic connectives [40]. So, transformations between logically equivalent formulas expressed by different connectives may have different formula complexity. Hence, the knowledge of such transformations is useful in optimization of interval inference.

For example, *exclusive OR*, or the *eor operator*, is conveniently defined in two ways: as 'a without b' or 'b without a', or else as 'a or b but not both', thus

$$a \text{ eor } b = (a \text{ and } \neg b) \text{ or } (\neg a \text{ and } b)$$
$$= (a \text{ or } b) \text{ and } (\neg a \text{ and } b).$$

Using these definitions together with the definitions of the previous section easy calculations bring the results of the following Table [6]:

$a \equiv_{\mathrm{top}} b$	$= (a \to_{\mathrm{top}} b) \wedge_{\mathrm{top}} (b \to_{\mathrm{top}} a)$
	$= (a \wedge_{\mathrm{top}} b) \vee_{\mathrm{top}} (\neg a \wedge_{\mathrm{top}} \neg b)$
$a \equiv_{\mathrm{bot}} b$	$= (a \to_{\mathrm{bot}} b) \wedge_{\mathrm{bot}} (b \to_{\mathrm{bot}} a)$
	$= (a \wedge_{\mathrm{bot}} b) \vee_{\mathrm{bot}} (\neg a \wedge_{\mathrm{bot}} \neg b)$
$a \oplus_{\mathrm{top}} b$	$= (a \wedge_{\mathrm{top}} \neg b) \vee_{\mathrm{top}} (\neg a \wedge_{\mathrm{top}} b)$
	$= (a \vee_{\mathrm{top}} b) \wedge_{\mathrm{top}} \neg(a \wedge_{\mathrm{bot}} b)$
$a \oplus_{\mathrm{bot}} b$	$= (a \wedge_{\mathrm{bot}} \neg b) \vee_{\mathrm{bot}} (\neg a \wedge_{\mathrm{bot}} b)$
	$= (a \vee_{\mathrm{bot}} b) \wedge_{\mathrm{bot}} \neg(a \wedge_{\mathrm{top}} b)$

Formulas for equivalence (IFF) and exclusive-OR (EOR).

Other useful formulas are those that give universal bounds on classes of *fuzzy interval pairs* of formulas. If we define the *unnormalized fuzziness* of x [1] as $\phi x = \min(x, 1-x)$, then for x in the range $[0,1]$, ϕx is in the range $[0, 0.5]$, with value 0 if and only if x is crisp , and value .5 if and only if x is .5. The following *gap theorem* holds [8]:

THEOREM 1

$$(a \wedge_{\mathrm{top}} b) - (a \wedge_{\mathrm{bot}} b)$$
$$= (a \vee_{\mathrm{top}} b) - (a \vee_{\mathrm{bot}} b)$$
$$= (a \to_{\mathrm{top}} b) - (a \to_{\mathrm{bot}} b)$$
$$= \min(\phi a, \phi b).$$
$$(a \equiv_{\mathrm{top}} b) - (a \equiv_{\mathrm{bot}} b)$$
$$= (a \oplus_{\mathrm{top}} b) - (a \oplus_{\mathrm{bot}} b)$$
$$= 2\min(\phi a, \phi b).$$

□

The width of the interval produced by an application of a pair of associated connectives (i.e. TOP and BOT connectives) characterises the margins of imprecision of an interval logic expression. Because the interval between the TOP connective and the BOT connective is directly linked to the concept of fuzziness ϕ, the margins of imprecision can be directly measured by the degree of ϕ.

Other Systems of Checklist Paradigm Connectives. Several measures other than m_1 that yield interesting results are also important. For implication again, but only the evaluation 'by performance' (that is, we are only concerned with the cases in which the evaluation of A is 1; see Fig. 1), we use $m_2 = u_{11}/(u_{10} + u_{11})$ and obtain the inequality

$$\min\left(1, \frac{b}{a}\right) \geq m_2(F) \geq \max\left(0, \frac{a+b-1}{a}\right),$$

in which the left-hand side is the well-known *Goguen–Gaines implication* (cf. e.g. [3]). Still another *contracting measure* which distinguishes the proportion of satisfactions 'by performance', $u(1,1)$, and 'by default', $u(0,0)+u(0,1)$. This measure given by the formula $m_3 = u_{11} \vee (u_{00} + u_{01})$ yields [3]

$$\max[\min(a,b), 1-a] \geq m_3(F) \geq \max(a+b-1, 1-a).$$

Two variations on measure m_3 have turned out to be of interest [3]. One is its lower *contrapositivization* given by the formula

$$m_4 = (u_{11} \vee (u_{00} + u_{01})) \vee (u_{00} \vee (u_{01} + u_{11}))$$

which gives the following inequality:

$$\min[\max(a+b-1, 1-a), \max(b, 1-a-b)] \leq m_4 \leq \min[\max(1-a, b), \kappa a, \kappa b],$$

where $\kappa a = a \vee (1-a)$.

The other arises by taking for the 'performance' part the less conservative m_2 thus obtaining the formula for $m_5 = m_2 \vee (u_{00} + u_{11})$. This yields

$$\max\left[\min\left(1, \frac{b}{a}\right), 1-a\right] \geq m_5 \geq \max\left[\frac{a+b-1}{a}, 1-a\right].$$

For the proofs of the results presented in this subsection and further explanation see [3, Sect. 5; 6] (this is the first paper on the checklist paradigm, published in 1980).

Collapse of Intervals into Points Under the Additional Probabilistic Constraints. When only the row and column totals r_i, c_j of the fine structure are known (see Fig. 1), one can ask what are the expected values for the α_{ij} ([3, Sect. 7]).

Suppose the ways in which numbers can be distributed within the cells of the fine structure (so as to give the fixed coarse totals) constitute a hypergeometric distribution. Then the means of the distribution for each cell, give the *expected* configuration of the fine structure. The inequalities determining the interval BOTCON $\leq$ TOPCON now turn into equalities:

$$\frac{\alpha_{ij}}{\alpha_{ik}} = \frac{c_j}{c_k}; \qquad \frac{\alpha_{ij}}{\alpha_{hk}} = \frac{r_i}{r_h}.$$

Surprisingly, introducing the expected value (a probabilistic notion) this way causes the fuzzy interval to collapse into a single point: the expected value thus generating the values of the *mid connective* [3], [5]. For example, the interval pair $(\rightarrow_5, \rightarrow_6)$ generated by m_1 consisting of the Łukasiewicz implication and Kleene–Dienes implication operators collapses into the *Reichenbach implication* operator.

For further details on the *'probabilistic collapse'* of the interval pairs generated by other measures see [3], [20], [5], [8], [26].

Checklist Paradigm and Generalized Quantifiers. Some of the notions and results of the checklist paradigm are in a remarkable relation to the theory of observational generalized quantifiers, as studied by P. Hájek and T. Havránek [14] in connection with the method of *automated hypothesis formation* [12], [13]. Namely, a particular type of implication operator and a particular type of implicational quantifier are mutually definable. The link is given by the contingency tables of the checklist paradigm and the statistics of the observational quantifiers [15].

Checklist Paradigm and Four Modes of Reasoning. Classical two-valued logic has presented certain modes of reasoning, of which only two concern us: *modus ponens* and *modus tollens*, respectively.

The first of these derives from the two premises 'if a then b' and 'a', the conclusion 'b'; the second derives from 'if a then b' and 'not b', the conclusion 'not a'. The validity of these modes is trivial.

On the other hand, there are two modes of reasoning which are classically illegitimate, although in the daily life we all use something very much like them all the time. These, so-called *plausible rules* [32], [4], are shown as the central pair in [26, Fig. 1]. *Denial* derives from 'if a then b' and 'not a', the assertion 'not b', while *confirmation* derives from 'if a then b' and 'b', the assertion 'a' .

The reason why these errors in classical reasoning retain a strong intuitive attraction is that most human reasoning does not deal with crisp or two-valued or Boolean truth-versus-falsity, but with graded *degrees of credence*, or belief-worthiness, or

whatever you like to call it. Because in multiple-valued logics the plausible modes *gain legitimacy* this intuition about human reasoning gains mathematical legitimacy. Indeed, human reasoning, 'good' human reasoning, is best modeled in multiple-valued logic which admits in addition to the modus ponens and modus tollens also the two modes of plausible reasoning.

In classical logic, an *evaluation* takes each of a given set of propositions into one of the extreme truth-values 0 (false) or 1 (true), subject to some semantic consistency rules. In multiple-valued logic, an *evaluation* is a mapping of the set of propositions into a somewhat richer set, which for present purposes may be taken to be the closed interval $[0, 1]$ from 0 to 1, again subject to certain semantic consistency rules.

Hence, in multiple-valued logic, for any fixed choice among the distinguished implication operators, to the classically valid modes of modus ponens and modus tollens are to be added fuzzily valid modes of denial and confirmation (*modus negans* and *modus confirmans*) [4], [7]. Although the out-of-bounds constraints were addressed elsewhere [4], one may wonder, what does the checklist paradigm have to offer when applied to the four plausible modes of inference.

Checklist paradigm is applicable not only to the components of the object language, such as logical operators and connectives, but also at the metalevel, thus providing an interval logic based semantics for various rules of inference. As shown below and in [8], it also provides a justification and the proofs of validity of nonclassical interval-based rules (plausible modes) of reasoning called denial and confirmation (modus negans and modus confirmans) [4], [7], [8], [24].

As already mentioned, these do not have a nontrivial analogy in Boolean crisp logic. Thus we have the following theorems.

THEOREM 2 (*checklist modus ponens*) Given $r = m(A \to B)$ and $a = m(A)$ satisfying the consistency condition $r \geq 1 - a$, the values of $b = m(B)$ are subject to

$$r - (1 - a) \leq b \leq r. \tag{1}$$

□

THEOREM 3 (*checklist confirmation*) Given $r = m(A \to B)$ and $b = m(B)$ subject to the consistency condition $b \leq r$, the values of $a = m(A)$ are subject to

$$1 - r \leq a \leq 1 - (r - b). \tag{2}$$

□

THEOREM 4 (*checklist modus tollens*) Given $r = m(A \to B)$ and $\neg b = m(\text{not} - B)$ satisfying the consistency condition $r \geq b$, the values of $\neg a = m(\text{not} - A)$ are subject to

$$r - b \leq \neg a \leq r. \tag{3}$$

□

THEOREM 5 (*checklist denial*) $r = m(A \to B)$ and $\neg a = m(\text{not} - A)$ subject to the consistency condition $1 - a \leq r$, the values of $\neg b = m(\text{not} - B)$ are subject to

$$1 - r \leq \neg b \leq 2 - (r + a). \tag{4}$$

□

Group Transformations of Logic Connectives and the Checklist Paradigm. Let us recall that a *realization of an abstract group* is any group of concretely realizable operations which has the same algebraic structure as the given abstract group. It is well known that any abstract group can be concretely realized by a family of permutations. So a specific abstract group provides a global structural characterization of a specific family of permutations that concretely represent this abstract group. This idea can be used for global characterization of logic connectives.

The Piaget Group of Transformations. Such a global characterization of two-valued connectives of logic was first given by Piaget in the context of studies of human cognitive development. J. Piaget and his collaborators have shown that an important role in child's mental development is transition from more concrete to more abstract thinking. This transition plays a role in development of intelligence, which is viewed in the Piagetian setup as a transition from totally ambiguous and vague notions to crisp propositions in two-valued logic.

Given a family of logical connectives one can apply to them various transformations. Individual

logic connectives are 2-argument logic functions. Transformations are functors that, taking one connective as the argument will produce another connective.

Let 4 transformations on basic propositional functions $f(x, y)$ of 2 arguments be given as follows:

$$I(f) = f(x, y), \quad D(f) = \neg f(\neg x, \neg y),$$
$$C(f) = f(\neg x, \neg y), \quad N(f) = \neg f(x, y).$$

In 1940, Piaget discovered experimentally a specific concrete form of such transformations. In the set of the above transformations $T_p = \{I, D, C, N\}$ these individual transformations are called *identity*, *dual*, *contradual*, *negation transformation*, respectively.

It has been shown that the *Piaget group of transformation* is satisfied by some many-valued logics (cf. [37], [10], [5], [6]).

The system of connectives

$$\{\equiv_{\mathrm{TOP}}, \oplus_{\mathrm{BOT}}, \equiv_{\mathrm{BOT}}, \oplus_{\mathrm{TOP}}\}$$

obeys the Piaget group of transformations. Hence it possesses the abstract structure of the *Klein 4-element group*.

An 8-Element Group of Logic Transformations. Adding new nonsymmetrical transformations to those defined by Piaget enriches the algebraic structure of logic transformations. In 1979 L.J. Kohout and W. Bandler added the following nonsymmetric operations [22], [23]:

$$LC(f) = f(\neg x, y), \quad RC(f) = f(x, \neg y)$$
$$LD(f) = \neg f(\neg x, y), \quad RD(f) = \neg f(x, \neg y)$$

to the above defined four symmetrical transformations. This yields a new 8-element group of transformations.

The abstract 8-element group $\{T, *\}$ that captures the structure of the above defined logic transformations is also commutative and is called the *symmetric* $\mathcal{S}_{2\times2\times2}$ *group* in the standard terminology of group theory. The interval logic system based on m_1 can be characterized by such groups of transformations.

Given a set of connectives CON and a set of transformations $\mathcal{T}$, we say that $T_{CON,\mathcal{T}} = \mathcal{T}(\mathrm{CON})$ is the set of connectives generated by the application of $\mathcal{T}$ to the set CON. For example, $a \to_5 b$ will generate such a set of connectives. This generated set is a realization of $\mathcal{S}_{2\times2\times2}$.

A 16-Element $\mathcal{S}_{2\times2\times2\times2}$ Group of Logic Transformations. The implication operators $a \to_5 b$ and $a \to_6 b$ yielding the measure m_1 are contrapositive. This means that their valuations satisfy the semantic equality $a \to b = \neg b \to \neg a$.

If we are interested in extending the interval logics into the domain of noncontrapositive $\to$ then the corresponding $\wedge$ is not commutative. In order to distinguish the contrapositive cases from noncontrapositive ones in a syntactically correct formal way, an additional operator is introduced.

This operator called, *commutator* K, satisfies the equality $a * b = K(b * a)$. The commutativity as well as the contrapositivity involves restrictions on transformations of connectives. In the abstract group, these restrictions are expressed abstractly as congruences [30]. It is convenient to express such restrictions equationally. For any contrapositive $\to$, the following equalities hold: $C[K(a \to b)] = K[C(a \to b)] = a \to b$. For a noncontrapositive $\to$, (1) fails, but the following equality holds:

$$(K(C(K(C(a \to b))))) = a \to b.$$

The following holds [29]:

THEOREM 6 The closed set of connectives generated by $\{\to_4, \leftarrow_4, K\}$ is a representation of the symmetric 16 element abstract group $S_{2\times2\times2\times2}$. □

Conclusion. The checklist paradigm clearly demonstrates the following general meta-principle: a system of logic connectives is formed by a specific family of connectives together with some common process/structure/principles that involve the said family of connectives in some unifying way, causing these to interact.

In the checklist paradigm semantic model we use two basic unifying principles:

i) approximation (contraction) measures;

ii) transformations of logical types of connectives leading to a global characterization of logics by their groups of transformations [23].

The methods of the checklist paradigm surveyed here give the theoretical bounds on the performance of particular many-valued implication operators and other connectives by deriving these from deeper epistemological and formal assumptions. Hence, it provides a theoretical justification of interval-valued approximate inference. The checklist paradigm, together with fuzzy questionnaires and square and triangle relational products also plays an important role in the experimental identification of fuzzy membership functions and structures [25], [21] (see also **Boolean and fuzzy relations**). The results can be extended to the groupoid-based many-valued *Pinkava algebras* (see **Finite complete systems of many-valued logic algebras**) that are used in the design of knowledge-based and other systems [19]. This theoretical work is supplemented by empirical studies of the adequacy of various logical connectives in practical applications of fuzzy sets and relations [9].

See also: **Boolean and fuzzy relations**; **Alternative set theory**; **Finite complete systems of many-valued logic algebras**; **Optimization in Boolean classification problems**; **Optimization in classifying text documents**; **Inference of monotone Boolean functions**.

References

[1] BANDLER, W., AND KOHOUT, L.J.: 'Fuzzy relational products and fuzzy implication operators': *Internat. Workshop on Fuzzy Reasoning Theory and Appl.*, Queen Mary College Univ. London, 1978.

[2] BANDLER, W., AND KOHOUT, L.J.: 'Fuzzy power sets and fuzzy implication operators', *Fuzzy Sets and Systems* **4** (1980), 13–30, Reprinted in: Readings in Fuzzy Sets for Intelligent Systems, D. Dubois, H. Prade and R. Yager (eds.), Morgan Kaufmann 1993, pp. 88-96.

[3] BANDLER, W., AND KOHOUT, L.J.: 'Semantics of implication operators and fuzzy relational products', *Internat. J. Man-Machine Studies* **12** (1980), 89–116, Reprinted in: Fuzzy Reasoning and its Applications, E.H. Mamdani and B.R. Gaines (eds.), Acad. Press, 1981, pp. 219-246.

[4] BANDLER, W., AND KOHOUT, L.J.: 'The four modes of inference in fuzzy expert systems', in R. TRAPPL (ed.): *Cybernetics and Systems Res.*, Vol. 2, North-Holland, 1984, pp. 581–586.

[5] BANDLER, W., AND KOHOUT, L.J.: 'Unified theory of multiple-valued logical operators in the light of the checklist paradigm': *Proc. 1984 IEEE Conf. Systems, Man Cybern.*, IEEE, 1984, pp. 356–364.

[6] BANDLER, W., AND KOHOUT, L.J.: 'The interrelations of the principal fuzzy logical operators', in M.M. GUPTA, A. KANDEL, W. BANDLER, AND J.B. KISZKA (eds.): *Approximate Reasoning in Expert Systems*, North-Holland, 1985, pp. 767–780.

[7] BANDLER, W., AND KOHOUT, L.J.: 'Probabilistic vs. fuzzy production rules in expert systems', *Internat. J. Man-Machine Studies* **22** (1985), 347–353.

[8] BANDLER, W., AND KOHOUT, L.J.: 'The use of checklist paradigm in inference systems', in C.V. NEGOITA AND H. PRADE (eds.): *Fuzzy Logic in Knowledge Engineering*, TÜV Rheinland, Köln, 1986, pp. 95–111.

[9] BEN-AHMEIDA, B., KOHOUT, L.J., AND BANDLER, W.: 'The use of fuzzy relational products in comparison and verification of correctness of knowledge structures', in L.J. KOHOUT, J. ANDERSON, AND W. BANDLER (eds.): *Knowledge-Based Systems for Multiple Environments*, Ashgate Publ. (Gower), 1992.

[10] DUBOIS, D., AND PRADE, H.: *Fuzzy sets and systems: Theory and applications*, Acad. Press, 1980.

[11] DUBOIS, D., AND PRADE, H.: 'Fuzzy sets in approximate reasoning, Part I: Inference with possibility distributions', *Fuzzy Sets and Systems* **40**, no. 1 (1991), 143–202.

[12] HÁJEK, P.: 'Special issue on GUHA method', *Internat. J. Man-Machine Studies* **10**, no. 1 (1978), 1–93, (guest editor).

[13] HÁJEK, P.: 'Special issue on GUHA method', *Internat. J. Man-Machine Studies* **15** (1981), (guest editor).

[14] HÁJEK, P., AND HAVRÁNEK, T.: *Mechanizing hypothesis formation: Mathematical foundations of a general theory*, Springer, 1978.

[15] HÁJEK, P., AND KOHOUT, L.J.: 'Fuzzy implications and generalized quantifiers', *Internat. J. Uncertainty, Fuzziness and Knowledge Based Systems* **4**, no. 3 (1996), 225–233.

[16] HISDAL, E.: *Infinite-valued logic based of two-valued logic and probability. Pt. 1.4 The TEE model for grades of membership*, Inst. Informatics Univ. Oslo, 1990.

[17] KLEMENT, E.P., AND MESIAR, R.: 'Triangular norms', in R. MESIAR AND B. RIEČAN (eds.): *Tatra Mountains, Math. Publ. (Special Issue: Fuzzy Structures - Current Trends)*, Vol. 13, Math. Inst. Slovak Acad. Sci. Bratislava, 1997, pp. 169–194.

[18] KLIR, G.J., AND YUAN, B.: *Fuzzy sets and fuzzy logic: Theory and applications*, Prentice-Hall, 1995.

[19] KOHOUT, L.J.: *A perspective on intelligent systems: A framework for analysis and design*, Chapman and Hall and v. Nostrand, 1990.

[20] KOHOUT, L.J.: 'Epistemological aspects of many-valued logics and fuzzy structures', in U. HÖHLE AND E.P. KLEMENT (eds.): *Non-Classical Logics and their Applications to Fuzzy Subsets: A Handbook of Mathematical Foundations of Fuzzy Set Theory*, Kluwer Acad. Publ., 1995, pp. 291–339.

[21] KOHOUT, L.J., ANDERSON, J., BANDLER, W., ET AL.:

Knowledge-based systems for multiple environments, Ashgate Publ. (Gower), 1992.

[22] KOHOUT, L.J., AND BANDLER, W.: 'Checklist paradigm and group transformations', *Techn. Note Dept. Electrical Engin. Univ. Essex.*, no. EES-MMS-ckl91.2 (1979).

[23] KOHOUT, L.J., AND BANDLER, W.: 'How the checklist paradigm elucidates the semantics of fuzzy inference': *Proc. IEEE Internat. Conf. Fuzzy Systems (1992)*, IEEE, 1992, pp. 571–578.

[24] KOHOUT, L.J., AND BANDLER, W.: 'Modes of plausible reasoning viewed via the checklist paradigm': *IPMU'92: Proc. Internat. Conf. Information Processing and the Management of Uncertainty in Knowlege-Based Systems, Mallorca, July 6-10 (1992)*, Univ. Illes Balears, Palma, Mallorca, Spain, 1992, pp. 411–414.

[25] KOHOUT, L.J., AND BANDLER, W.: 'Use of fuzzy relations in knowledge representation, acquisition and processing', in L.A. ZADEH AND J. KACPRZYK (eds.): *Fuzzy Logic for the Management of Uncertainty*, Wiley, 1992, pp. 415–435.

[26] KOHOUT, L.J., AND BANDLER, W.: 'Interval-valued systems for approximate reasoning based on the checklist paradigm', in P. WANG (ed.): *Advances in Fuzzy Theory and Technology*, Vol. 1, Bookwrights Press, 1993, pp. 167–193.

[27] KOHOUT, L.J., AND BANDLER, W.: 'Fuzzy interval inference utilizing the checklist paradigm and BK-relational products', in R.B. KEARFOTT AND V. KREINOVICH (eds.): *Applications of Interval Computations*, Kluwer Acad. Publ., 1996, pp. 291–335.

[28] KOHOUT, L.J., AND BANDLER, W.: *A survey of fuzzy and crisp relations*, Lecture Notes Fuzzy Math. and Computer Sci. Creighton Univ., 2001.

[29] KOHOUT, L.J., AND KIM, E.: 'Global characterization of fuzzy logic systems with para-consistent and grey set features', in P. WANG (ed.): *Proc. 3rd Joint Conf. Inform. Sci. JCIS'97 (5th Internat. Conf. on Fuzzy Theory and Techn.)*, Vol. 1, Duke Univ., 1997, pp. 238–241.

[30] KOHOUT, L.J., AND KIM, E.: 'Group transformations of systems of logic connectives': *Proc. IEEE-FUZ'97*, Vol. 1, IEEE, 1997, pp. 157–162.

[31] NOVÁK, V.: 'On the position of fuzzy sets in modelling of vague phenomena', in', in R. LOWEN AND M. ROUBENS (eds.): *IFSA '91 Brussels: Artificial Intelligence*, Internat. Fuzzy Systems Assoc., 1991, pp. 165–167.

[32] POLYA, G.: *Mathematics and plausible reasoning*, Princeton Univ. Press, 1954.

[33] PRIOR, A.: *Formal logic*, Oxford Univ. Press, 1962.

[34] RESCONI, G., KLIR, G.J., AND U.ST.CLAIR: 'Hierarchical uncetrainty metatheory based upon modal logic', *Internat. J. General Syst.* (1992).

[35] RIEČAN, B., AND NEUBRUNN, T.: *Integral, measure, and ordering*, Kluwer Acad. Publ., 1997.

[36] SCHWEIZER, B., AND SKLAR, A.: *Probabilistic metric spaces*, North-Holland, 1983.

[37] TURKSEN, I.B.: 'Containment and Klein groups of fuzzy propositions', *Working Paper Dept. Industr. Engin. Univ. Toronto* **79-010** (1979).

[38] TURKSEN, I.B.: 'Interval-valued fuzzy sets based on normal forms', *Fuzzy Sets and Systems* **20** (1986), 191–210.

[39] TURKSEN, I.B.: 'Four methods of approximate reasoning with interval-valued fuzzy sets', *Internat. J. Approximate Reasoning* **3** (1989), 121–142.

[40] TÜRKSEN, I.B.: 'Type I and interval-valued type II fuzzy sets and logics', in P.P. WANG (ed.): *Advances in Fuzzy Theory and Techn.*, Vol. 3, Duke Univ., 1995, pp. 31–81.

[41] VOPĚNKA, P.: *Mathematics in the alternative set theory*, Teubner, 1979.

[42] VOPĚNKA, P.: 'The philosophical foundations of alternative set theory', *Internat. J. General Syst. (Special Issue of Fuzzy Sets and Systems in Czechoslovakia)* **20**, no. 1 (1991), 115–126.

[43] WILLMOTT, R.: 'Two fuzzier implication operators in the theory of fuzzy power sets', *Fuzzy Sets and Systems* **4**, no. 31-36 (1980), 31–36.

[44] ZADEH, L.A.: 'Fuzzy sets', *Inform. and Control* **8** (1965), 338–353.

[45] ZADEH, L.A.: *Fuzzy sets: Selected papers II*, World Sci., 1996, Edited by G. Klir and B. Yuan.

Ladislav J. Kohout
Dept. Computer Sci. Florida State Univ.
Florida 32308-4530, USA
E-mail address: kohout@cs.fsu.edu

MSC2000: 03B52, 03B50, 03C80, 62F30, 62Gxx, 68T27
Key words and phrases: checklist paradigm, many-valued logics, GUHA, exploratory statistical analysis, semantics of MVL connectives, generalized quantifier, observational quantifiers, fuzzy sets, logic of approximation, interval computing, approximate reasoning.

CHEMICAL PROCESS PLANNING

As companies are increasingly concerned about long term stability and profitability, recent years have witnessed growing demand for long range planning tools in all sectors. The chemical process industries are no exception. New environmental regulations, rising competition, new technology, uncertainty of demand, and fluctuation of prices have all led to an increasing need for decision policies that will be 'best' over a long time horizon. Quantitative techniques have long established their importance in such decision making problems. It is therefore no surprise that there is a

considerable number of papers in the optimization literature devoted to the problem of long range planning in the processing industries. The purpose of this article is to review recent advances in this area. We will describe the main modeling issues, and discuss the computational complexity, formulations and solution algorithms for this problem.

The Long Range Planning Problem. Consider a plant comprising of several processes to produce a set of chemicals for sale. Each process intakes a number of raw materials and produces a main product along with some by-products. Any of these main or by-products could be the raw materials for another process. Considering the ingredients and final product of all the processes, we have a list of chemicals consisting of all raw materials we consider purchasing from the market, all products we consider offering for sale on the market, and all possible intermediates. The plant can then be represented as a network comprising of nodes representing processes and the chemicals in the list, interconnected by arcs representing the different alternatives that are possible for processing, and purchases to and sales from different markets.

The *process planning problem* then consists of choosing among the various alternatives in such way as to maximize profit. Once we know the prices of chemicals in the various markets and the operating costs of processes, the problem is then to decide the operating level of each process and amount of each chemical to be purchased and sold to the various markets. The problem in itself grows combinatorially with the number of chemicals and processes and is further complicated once we start planning over multiple time periods.

Let us now consider the operation of the plant over a number of time periods. It is reasonable to expect that prices and demands of chemicals in various markets would fluctuate over the planning horizon. These fluctuations along with other factors, such as new environmental regulations or technology obsolescence, might necessitate the decrease or complete elimination of the production of some chemicals while requiring increase or introduction of others. Thus, we have some additional new decision variables: capacity expansion of existing processes, installation of new processes and shut down of existing processes. Moreover, due to the broadening of the planning horizon, the effect of discount factors and interest rates will become prominent in the cost and price functions. Thus, the planning objective should be to maximize the net present value instead of short term profit or revenue. This is the problem that we shall devote our attention to. The problem can be stated as follows: assuming a given network of processes and chemicals, and characterization of future demands and prices of the chemicals and operating and installation costs of the existing as well as potential new processes, we want to find an operational and capacity planning policy that would maximize the net present value.

Computational Complexity. The number of possible alternatives, regarding which processes to expand and when, increases with the number of processes and the number of time periods. Even though this increase is clearly exponential in the number of processes and time periods, it was not until recently that a formal computational complexity characterization was provided for this problem. In particular, the general long range process planning problem has been shown by S. Ahmed and N.V. Sahinidis [3] to be *NP*-hard by identifying two known *NP*-hard problems as special cases.

Consider first a single-process, multiperiod problem where the decisions consist of determining the expansion sequence to satisfy given demands over a number of time periods at a minimum cost. It can be shown that this problem is equivalent to the *NP*-hard *capacitated lot-sizing problem*, where one has to determine production lot sizes to satisfy demands at a minimum cost. Similarly, a multiple-process, single-time period problem, where the decisions are to determine which processes to install to satisfy demand at a minimum cost, can be shown to be equivalent to the *NP*-hard *knapsack problem*, where one has to select items from a set to place into a knapsack such that weight restrictions are not violated and utility is maximized.

Solution Strategies. Some of the early approaches for the long range planning problem were based on dynamic programming as described by

S.M. Roberts [18]. A.S. Manne [5], [15] used integer programming approaches to account for economies of scale. D.M. Himmelblau and T.C. Bickel [7] presented a nonlinear programming formulation for a hydrodesulfurization process, and I.E. Grossmann and J. Santibez [6] developed a multi period mixed integer linear programming formulation. Y. Shimizu and T. Takamatsu [22] discussed a goal programming approach where in addition to cost minimization, minimizing the number of expansions is also suggested. M. Santiago, O.A. Iglesias and C.N. Pamiagua [21] developed a method to handle nonlinear concave cost functions arising in planning models. A.G. Jimenez and D.F. Rudd [9] presented a recursive mixed integer linear programming technique and applied it to the Mexican petrochemical industry. We next describe some of the more contemporary approaches to these problems.

Integer Programming Approach. Under the assumption of linear mass balances in the processes and fixed charge cost models, Sahinidis et al. [20] developed a mixed integer linear programming (MILP) formulation of the long range process planning problem as described below.

Indices

i For the set of NP processes

j For the set of NC chemicals

l For the set of NM markets

t For the set of NT time periods.

Parameters

X_{it}^L, X_{it}^U Lower and upper bounds on the expansion of process i in period t.

a_{jlt}^L, a_{jlt}^U Lower and upper bounds on the availability of chemical j in market l in period t.

d_{jlt}^L, d_{jlt}^U Lower and upper bounds on the demand of chemical j in market l in period t.

Γ_{jlt}, γ_{jlt} Forecasted buying and selling prices of chemical j in market l in period t.

μ_{ij}, η_{ij} Input and output proportionality constants for chemical j in process i.

α_{it}, β_{it} Variable and fixed cost for the expansion of process i at the beginning of period t.

Variables

X_{it} Capacity expansion of process i at the beginning of period t.

P_{jlt} Amount of chemical j purchased from market l at the beginning of period t.

Q_{it} Total capacity of process i in period t. The capacity of a process is expressed in terms of its main product.

S_{jlt} Units of chemical j sold to market l at the end of period t.

W_{it} Operating level of process i in period t expressed in terms of output of its main product.

y_{it} A 0–1 integer variable. If process i is expanded during period t then $y_{it} = 1$, else $y_{it} = 0$.

Formulation

$$\max NPV \tag{1}$$

$$= \sum_{t=1}^{NT} \left\{ -\sum_{i=1}^{NP} (\alpha_{it} X_{it} + \beta_{it} y_{it} + \delta_{it} W_{it}) + \sum_{j=1}^{NC} \sum_{l=1}^{NM} (\gamma_{jlt} S_{jlt} - \Gamma_{jlt} P_{jlt}) \right\}$$

subject to

$$y_{it} X_{it}^L \le X_{it} \le y_{it} X_{it}^U, \quad \forall i,t \tag{2}$$

$$Q_{it} = Q_{it-1} + X_{it}, \quad \forall i,t \tag{3}$$

$$W_{it} \le Q_{it}, \quad \forall i,t \tag{4}$$

$$\sum_{i=1}^{NP} (\eta_{ij} - \mu_{ij}) W_{it} = \sum_{l=1}^{NM} S_{jlt} - \sum_{l=1}^{NM} P_{jlt}, \quad \forall j,t \tag{5}$$

$$a_{jlt}^L \le P_{jlt} \le a_{jlt}^U, \quad \forall j,l,t \tag{6}$$

$$d_{jlt}^L \le S_{jlt} \le d_{jlt}^U, \quad \forall j,l,t \tag{7}$$

$$X_{it}, Q_{it}, W_{it} \ge 0, \quad \forall i,t \tag{8}$$

$$y_{it} \in \{0,1\}, \quad \forall i,t. \tag{9}$$

The objective (1) in the above formulation is to maximize the difference between the sales revenues of the final products and the investment, operating, and raw material costs. Equation (2) is a constraint that bounds capacity expansions. A zero value of y_{it} forces the capacity expansion of process i at period t to zero. If the binary variable equals 1, then the capacity expansion is performed within prescribed bounds. Constraint (3) in the above for-

mulation defines the total capacity available at period t as a sum of capacity available in period $t-1$ and the capacity expansion at the beginning of period t. The condition that the operating level of any process cannot exceed the installed capacity is modeled by constraint (4). Equation (5) expresses mass balances for chemicals across processes and markets. Constraints (6) and (7) are bounds on the purchase and sales quantities. The nonnegativity and binary restrictions are imposed through constraints (8) and (9). Various extensions of this general model are discussed in the recent survey article [2].

Sahinidis et al. [20] developed strong bounding techniques and cutting planes to be used within a *branch and bound* framework to solve the above problem. The fact that the problem is decomposable in the number of time periods can also be exploited by using *Benders decomposition*. Further improvement of the bounding schemes are suggested by reformulating the problem to exploit lot sizing substructure in [19]. The reformulated problem results in a large number of constraints and variables. In [10], the reformulated problem is projected onto a lower-dimensional space to reduce the number of variables, and is solved using a cutting plane strategy along with branch and bound. Computational results in [10], [19] and [20] suggest the following:

- Branch and bound with strong bounding techniques performs much better than Benders decomposition for large problems.
- For small sized problems, the reformulation and projection approach do not provide appreciable gains.
- For large problems, the best approach is to use a cutting plane method based on the projected model.

Continuous Global Optimization. In the MILP model, economies of scale in the investment cost functions were modeled by the introduction of a set of binary decision variables (y_{it}) to impose a fixed charge on the decision to expand in addition to the linear term for variable costs. In reality, variable costs are not directly proportional to expansion quantity. Rather, the investment cost is a concave function because of the presence of quantity discounts. Thus, a more realistic model for the investment cost would be:

$$f(X_{it}) = \begin{cases} 0 & \text{when} \quad X_{it} = 0, \\ \beta_{it} + a_{it} X_{it}^{b_{it}} & \text{when} \quad X_{it} > 0, \end{cases}$$

where $a_{it} > 0$ and $0 < b_{it} < 1$. In this formulation, the integer variables have been discarded and the linear variable cost function has been replaced by a concave function in X_{it} with coefficient a_{it} and exponent b_{it}. Note that this function is discontinuous at $X_{it} = 0$. M.L. Liu, Sahinidis and J.P. Shectman [14] present two formulations using these concave cost functions. In the fixed charge concave programming model (FCP), the linear cost relation is retained but the discrete variables are eliminated by using the following concave function:

$$f(X_{it}) = \begin{cases} 0 & \text{when} \quad X_{it} = 0, \\ \beta_{it} + \alpha_{it} X_{it} & \text{when} \quad X_{it} > 0. \end{cases}$$

In the continuous concave programming model (CCP), the discontinuity at $X_{it} = 0$ is avoided by using the following function:

$$f(X_{it}) = a_{it} X_{it}^{b_{it}}.$$

Both (FCP) and (CCP) are problems with concave objective functions to be minimized over a set of linear constraints. These can be solved by a *concave programming* method based on the branch and bound procedure. Computational experience with these models suggests that the algorithm for (FCP) outperforms the straightforward branch and bound for the MILP formulation.

Approximation Schemes. Despite the success of optimization models and algorithms in solving problems of industrial relevance, the majority of approaches in current industrial-level planning practice are still based on *heuristics* rather than integer programming techniques. However, the performance characterization of these approximate methods is based on empirical evidence and little has been done in the way of analytical investigations. Liu and Sahinidis [12] developed a simple heuristic for the process planning problem. The method is based upon solving the LP relaxation of the MILP, and then shifting capacity expansions from latter periods to earlier periods while

maintaining feasibility. Worst-case bounds on the performance of this heuristic have also been developed and *probabilistic analysis* of the heuristic has shown that, under standard assumptions on the problem data, the heuristic solution converges to the optimal solution *almost surely* as the problem size increases. A modification of this heuristic for process planning problems with a restriction on the number of allowed expansions has been presented in [3]. The modified heuristic has been proven to be asymptotically optimal *in expectation.*

Dealing with Uncertainty. Uncertainty is an integral part of the long range process planning problem. In the deterministic models discussed above, it is assumed that all uncertainty has been accounted for in the estimation of the problem parameters. Stochastic models, on the other hand, provide explicit means of handling parameter uncertainties.

In process planning problems under uncertainty, the decision maker is interested in a plan that optimizes some sort of a stochastic objective. Two most common such objective functions in the literature are the expected cost/profit of the plan and the plan's flexibility.

Problems with the expected cost objective have been formulated as *two-stage stochastic linear programs* (2S-SLP). In such problems, the uncertain parameters are treated as random variables with known distributions. The desired degree of flexibility of the plan is pre-specified by identifying the probability space over which the plan is required to be feasible. The decision variables of the problem are partitioned into two sets. The *first stage* variables, which are often known as 'design' variables, have to be decided before the actual realization of the random parameters. Subsequently, once the values of the design variables have been decided and the random events presented themselves, further policy improvements can be made by deciding the values of the *second stage* variables, also known as 'control' or 'operating' variables. The choice of the design variables should be such that the first stage costs and the second stage expected costs are minimized. These problems have been solved using decomposition schemes, where the expectation functional over the uncertain parameter space has been approximated using Monte-Carlo sampling [11], successive disaggregation [4] or by Gaussian quadrature [8]. Computational results in [11] show that a combination of Benders decomposition with Monte-Carlo sampling provide optimal or excellent near-optimal solutions. Problems with up to 10 processes, 4 products, 6 chemicals, and with up to 5^{24} scenarios were solved in at most a few CPU minutes on a standard workstation.

From the flexibility objective point of view, one is interested in a plan that maximizes the range of the uncertain parameters over which the plan remains feasible. Problems of this type are typically harder to formulate and require the identification of a suitable measure of flexibility that one can optimize. Such a formulation has been presented in [24] which maximizes their *stochastic flexibility* metric [23] subject to a cost constraint.

The objectives of optimizing cost or profit and maximizing flexibility are typically conflicting. Formulations that combine the objectives by associating a retrofit cost corresponding to design flexibility have been presented in [16], [17].

Liu and Sahinidis [13] applied a *fuzzy programming* approach for the problem of process planning under uncertainty. In this model, the uncertain parameters are considered to be fuzzy numbers with a known range of values, and constraints are treated as 'soft,' i.e. some violation is allowed. The degree of satisfaction of the constraints is then measured in terms of *membership functions*, and the objective is to optimize a measure of constraint satisfaction.

The standard stochastic programming formulation does not address the variability of the uncertain recourse costs across the uncertain parameter scenarios. The need for enforcing robustness of these costs is particularly important to a risk aversive planner in a high variability environment. The stochastic programming formulation of the process planning problem has been extended in [1] to account for robustness of the recourse costs through the use of an appropriate variability criterion. In particular, upper partial mean has been proposed as the measure of variability for its intuitive appeal and to avoid nonlinear formulations. These

models provide the decision maker with a tool to analyze the trade-off associated with the expected profit and its variability. To overcome the difficulty associated with solving the robust models which include nonseparable terms, a heuristic procedure for the restricted recourse formulation has been developed. This method iteratively enforces recourse robustness while solving the standard stochastic program in each step. The heuristic generates similar but more conservative trade-off frontiers for the profit and its upper partial mean.

Conclusion. The purpose of this article has been to review the recent advances in the use of optimization techniques in long range chemical process planning. Considerable attention has been devoted to the mixed integer linear programming formulation of the problem and efficient solution schemes that exploit the structure of the problem have been developed. Continuous models have also been successfully solved using global optimization techniques. The combinatorial complexity of the problem has recently motivated the need for heuristics and their performance analysis. Some exciting new results have been obtained in this regard. Uncertainty of the problem parameters has been dealt with through stochastic programming and fuzzy programming models. Various different objective criteria including expected value, flexibility and variability have been considered in extensions of the two-stage stochastic programming formulation and a number of efficient algorithms have been developed for industrially relevant problems.

See also: **Mixed integer linear programming: Mass and heat exchanger networks; Mixed integer nonlinear programming; Generalized Benders decomposition; MINLP: Outer approximation algorithm; Generalized outer approximation; MINLP: Generalized cross decomposition; Extended cutting plane algorithm; MINLP: Logic-based methods; MINLP: Branch and bound methods; MINLP: Branch and bound global optimization algorithm; MINLP: Global optimization with αBB; MINLP: Heat exchanger network synthesis; MINLP: Reactive distillation column synthesis; MINLP: Design and scheduling of batch processes; MINLP: Applications in the interaction of design and control; MINLP: Application in facility location-allocation; MINLP: Applications in blending and pooling problems.**

References

[1] AHMED, S., AND SAHINIDIS, N.V.: 'Robust process planning under uncertainty', *Industr. Engin. Chem. Res.* **37** (1998), 1883–1892.

[2] AHMED, S., AND SAHINIDIS, N.V.: 'Techniques in long range planning in chemical manufacturing systems', in C.T. LEONDES (ed.): *Computer Aided and Integrated Manufacturing Systems: Techniques and Applications*, Internat. Ser. Engin., Techn. and Applied Sci., Gordon and Breach, 1998.

[3] AHMED, S., AND SAHINIDIS, N.V.: 'Analytical investigations of the process planning problem', *Computers Chem. Engin.* **23** (2000), 1605–1621.

[4] CLAY, R.L., AND GROSSMANN, I.E.: 'A disaggregation algorithm for the optimization of stochastic planning models', *Computers Chem. Engin.* **21**, no. 7 (1996), 751–774.

[5] GOREUX, L.M., AND MANNE, A.S.: *Multi-level planning: Case studies in Mexico*, North-Holland, 1973.

[6] GROSSMANN, I.E., AND SANTIBEZ, J.: 'Application of mixed-integer linear programming in process synthesis', *Computers Chem. Engin.* **4**, no. 205 (1980).

[7] HIMMELBLAU, D.M., AND BICKEL, T.C.: 'Optimal expansion of a hydrodesulfurization process', *Computers Chem. Engin.* **4**, no. 101 (1980).

[8] IERAPETRITOU, M.G., AND PISTIKOPOULOS, E.N.: 'Novel optimization approach of stochastic planning models', *Industr. Engin. Chem. Res.* **33** (1994), 1930–1942.

[9] JIMINEZ, A.G., AND RUDD, D.F.: 'Use of a recursive mixed-integer programming model to detect an optimal integration sequence for the mexical petrochemical industry', *Computers Chem. Engin.* **3**, no. 291 (1987).

[10] LIU, M.L., AND SAHINIDIS, N.V.: 'Long range planning in the process industries: A projection approach', *Comput. Oper. Res.* **23**, no. 3 (1995), 237–253.

[11] LIU, M.L., AND SAHINIDIS, N.V.: 'Optimization in process planning under uncertainty', *Industr. Engin. Chem. Res.* **35** (1996), 4154–4165.

[12] LIU, M., AND SAHINIDIS, N.V.: 'Bridging the gap between heuristics and optimization: The capacity expansion case', *AIChE J.* **43** (1997), 2289–2299.

[13] LIU, M.L., AND SAHINIDIS, N.V.: 'Process planning in a fuzzy environment', *Europ. J. Oper. Res.* **100**, no. 1 (1997), 142–169.

[14] LIU, M.L., SAHINIDIS, N.V., AND SHECHTMAN, J.P.: 'Planning of chemical processes via global concave minimization', in I.E. GROSSMANN (ed.): *Global Optimization in Engineering Design*, Kluwer Acad. Publ., 1996.

[15] MANNE, A.S. (ed.): *Investments for capacity expansion*, MIT, 1967.

[16] PISTIKOPOULOS, E.N., AND GROSSMANN, I.E.: 'Stochastic optimization of flexibility in retrofit design on linear systems', *Computers Chem. Engin.* **12** (1988), 1215–1227.

[17] PISTIKOPOULOS, E.N., AND GROSSMANN, I.E.: 'Optimal retrofit design for improving process flexibility in nonlinear systems-II. Optimal level of flexibility', *Computers Chem. Engin.* **13** (1989), 1087–1096.

[18] ROBERTS, S.M.: *Dynamic programming in chemical engineering and process control*, Acad. Press, 1964.

[19] SAHINIDIS, N.V., AND GROSSMANN, I.E.: 'Reformulation of the multiperiod MILP model for capacity expansion of chemical processes', *Oper. Res.* **40**, no. Supp. No. 1 (1992), S127–S144.

[20] SAHINIDIS, N.V., GROSSMANN, I.E., FORNARI, R.E., AND CHATHRATHI, M.: 'Optimization model for long range planning in the chemical industry', *Computers Chem. Engin.* **13**, no. 9 (1989), 1049–1063.

[21] SANTIAGO, M., IGLESIAS, O.A., AND PAMIAGUA, C.N.: 'Optimal technical paths for chemical processes', *Computers Chem. Engin.* **10** (1986), 421–431.

[22] SHIMIZU, Y., AND TAKAMATSU, T.: 'Application of mixed integer linear programming in multiterm expansion planning under multiobjectives', *Computers Chem. Engin.* **9**, no. 367 (1985).

[23] STRAUB, D.A., AND GROSSMANN, I.E.: 'Integrated stochastic metric of flexibility for systems with discrete state and continuous parameter uncertainties', *Computers Chem. Engin.* **14** (1990), 967–980.

[24] STRAUB, D.A., AND GROSSMANN, I.E.: 'Design optimization of stochastic flexibility', *Computers Chem. Engin.* **17** (1993), 339–354.

Shabbir Ahmed
Univ. Illinois at Urbana-Champaign
USA
E-mail address: s-ahmed1@uiuc.edu

Nikolaos V. Sahinidis
Univ. Illinois at Urbana-Champaign
USA
E-mail address: nikos@uiuc.edu

MSC2000: 90C90

Key words and phrases: long range planning, complexity, mixed integer linear programming, concave programming, probabilistic analysis, stochastic programming.

CHOLESKY FACTORIZATION

Solving a linear system of the form $Ax = b$ where $A \in \mathbf{R}^{n\times n}$, $b \in \mathbf{R}^n$ is one of the most fundamental problems in mathematics and science. The two basic categories of numerical solutions are direct methods and iterative methods. Of the direct methods, *Gaussian elimination with backsolving* is the most commonly used technique. Straightforward Gaussian elimination uses row-operations to reduce the system to upper-triangular form before backsolving to find the solution, whereas the equivalent *LU-decomposition* initially factors A into the product of a lower- and upper-triangular matrix: $A = LU$. In the special case where A is both symmetric and positive definite, the matrix may be decomposed into $A = \widetilde{L}\widetilde{L}^\top$ where $\widetilde{L} \in \mathbf{R}^{n\times n}$ is lower-triangular. This decomposition is known as the *Cholesky factorization*, and is named for A.L. Cholesky.

The LU-decomposition of a square matrix, A, is the factorization of A into the product of a lower-triangular matrix, $L \in \mathbf{R}^{n\times n}$ and an upper-triangular matrix, $U \in \mathbf{R}^{n\times n}$. The system $Ax = (LU)x = b$ is then solved by forward solving $Ly = b$ where $y = Ux$, and then backsolving $Ux = y$. The solution can be found in roughly the same number of *floating point operations* (*flops*) as Gaussian elimination with backward substitution. More specifically, both methods require about $n^3/3$ multiplications/divisions and $n^3/3$ additions/subtractions for large n. The main advantage of this method is that once the matrix is factored (which requires $O(n^3)$ steps), the system can be solved repeatedly for different b, which only requires $O(n^2)$ steps. One drawback of this method is that pivoting may be required to find the decomposition.

A special class of problems arises if the matrix in the system is *positive definite*, i.e. $x^\top Ax > 0$ for $x \neq 0$. Note that if A is positive definite, but not symmetric, this implies that $1/2(A+A^\top)$, the symmetric part of A, is positive definite. The matrix A, can then be decomposed into the form $A = LDM^\top$ where L, M are lower-triangular and D is a diagonal matrix containing the *pivots* of A. If, in addition to being positive definite, A is also *symmetric*, i.e. $A = A^\top$, then L is symmetric, $M = L$, and the matrix has the special decomposition $A = LDL^\top$, where D has positive entries. Therefore $\sqrt{D}$ exists and A can be decomposed into $A = \widetilde{L}\widetilde{L}^\top$, where $\widetilde{L} = L\sqrt{D}$ and is referred to as the *Cholesky triangle*. Hence the Cholesky factorization is often referred to as the 'square-rooting method' [5]. The major advantage of this is that it requires around half the flops of the standard LU-decomposition.

The Cholesky factorization, presented below for symmetric and positive definite $A \in \mathbf{R}^{n\times n}$ in pseudocode, is taken from [2].

FOR $k = 1, \ldots, n$,
  $a_{kk} := \left(a_{kk} - \sum_{p=1}^{k-1} a_{kp}^2\right)^{1/2}$
  FOR $i = k+1, \ldots, n$
    $a_{ik} := \left(a_{ik} - \sum_{p=1}^{k-1} a_{ip}a_{kp}\right)/a_{kk}$

A pseudocode for the Cholesky factorization.

EXAMPLE 1 Let $A = \begin{pmatrix} 4 & 2 \\ 2 & 5 \end{pmatrix}$. This matrix is both symmetric and positive definite. Therefore a Cholesky factorization exists for A (see [3] for a proof). An LU-decomposition for it is

$$A = \begin{pmatrix} 1 & 0 \\ 1/2 & 1 \end{pmatrix} \begin{pmatrix} 4 & 2 \\ 0 & 4 \end{pmatrix}.$$

Note that this is not unique, as another LU-decomposition is

$$A = \begin{pmatrix} 4 & 0 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} 1 & 1/2 \\ 0 & 1 \end{pmatrix}.$$

The pivots in both cases are 4 and 4. Hence, the $LDL^\top$-decomposition is

$$A = \begin{pmatrix} 1 & 0 \\ 1/2 & 1 \end{pmatrix} \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix} \begin{pmatrix} 1 & 1/2 \\ 0 & 1 \end{pmatrix}.$$

Finally, the Cholesky factorization is

$$A = LL^\top = \begin{pmatrix} 2 & 0 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix}.$$

□

Additionally, this factorization is unique for symmetric, positive definite matrices. The decomposition can be performed in fixed-point with *no pivoting required* [9]. This implies that the Cholesky decomposition is *guaranteed to be stable without pivoting.*

If the matrix A is *ill-conditioned*, i.e. has *condition number* $\kappa = \|A\|\,\|A^{-}1\| \gg 1$, then the matrix may be nearly singular and the computed solution to the system, $Ax = b$, may not be sufficiently accurate. A process of iterative refinement may be used to assess the accuracy of the solution and then improve upon it when working in higher precision is not practical. See [2], [9] for a more detailed explanation.

The Cholesky factorization can also be used to find the inverse and determinant of a symmetric, positive definite matrix (the LU-decomposition can be used for general $A \in \mathbf{R}^{n\times n}$). It is important for the matrix to be positive definite for a variety of reasons, for example, if A is symmetric, but not positive definite, then the stability is not guaranteed. In the case of finding the inverse of a matrix A, a poor inverse may be obtained even if A is *well-conditioned*, i.e. κ is 'close' to 1.

The efficiency of the Cholesky factorization can be further improved if the matrix is 'banded'. A matrix $A = [a_{ij}]$ is said to have *upper bandwidth* q if $a_{ij} = 0$ whenever $j > i + q$ and *lower bandwidth* p if $a_{ij} = 0$ whenever $i > j + p$. Since A is symmetric, when A has lower bandwidth p, it also has upper bandwidth p. In this case A is said to have *bandwidth* p. For example, if $p = 1$, then A is tridiagonal. The following algorithm from [2] takes advantage of the fact that A is symmetric, positive definite and has bandwidth p. It requires n square roots and

$$\frac{np^2}{2} - \frac{p^3}{3} + \frac{3}{2}(np - p^2)$$

flops or approximately $\frac{n(p^2+3p)}{2}$ flops for $p \ll n$.

FOR $i = 1, \ldots, n$
  FOR $j = \max\{1, i-p\}, \ldots, i-1$
    $a_{ij} := \left(a_{ij} - \sum_{k=\max\{1,i-p\}}^{j-1} a_{ik}a_{jk}\right)/a_{jj}$
  $a_{ii} := \left(a_{ii} - \sum_{k=\max\{1,i-p\}}^{i-1} a_{ik}^2\right)^{1/2}$

A pseudo-code for the banded Cholesky factorization.

Another important application of the Cholesky factorization is in the key role it plays in one of the most commonly used numerical techniques for solving the least squares problem (LS problem; cf. also **Least squares problems**). The *least squares problem* is to find the 'best' solution to $Ax = b$ when the system is inconsistent for $A \in \mathbf{R}^{m\times n}$. Instead, the system $A^\top Ax = A^\top b$, more commonly known as the *normal equations*, is solved by first finding the Cholesky factorization of the symmetric matrix $A^\top A = \widetilde{L}\widetilde{L}^\top$, which is positive definite if A has rank n. Next, $\widetilde{L}y = A^\top b$ is forward-solved, and finally, the 'best least squares' solution, $\widehat{x}$, is found by backsolving $\widetilde{L}^\top x = y$. Note that $\widehat{x}$ minimizes $\|Ax - b\|_2$ and the algorithm requires $O(n^3)$ flops. For the algorithm and an analysis of the accuracy of the method, see [2].

See also: **Symmetric systems of linear equations; Linear programming; Orthogonal triangularization; QR factorization; Solving large scale and sparse semidefinite programs; Large scale trust region problems; Large scale unconstrained optimization; ABS algorithms for linear equations and linear least squares; Overdetermined systems of linear equations.**

References

[1] BURDEN, R.L., AND FAIRES, J.D.: *Numerical analysis*, fifth ed., PWS/Kent Publ., 1993.

[2] GOLUB, G.H., AND LOAN, C.F. VAN: *Matrix computations*, third ed., Johns Hopkins Univ. Press, 1996.

[3] GRIFFEL, D.H.: *Linear algebra and its applications*, Vol. 2, Halsted Press, 1989.

[4] HAGER, W.W.: *Applied numerical linear algebra*, Dept. Math. Univ. Florida, Gainesville, FL, 1995.

[5] HOUSEHOLDER, A.S.: *The theory of matrices in numerical analysis*, Dover, 1964.

[6] KINCAID, D., AND CHENEY, W.: *Numerical analysis*, Brooks/Cole, 1991.

[7] LEON, S.J.: *Linear algebra with applications*, fifth ed., Prentice-Hall, 1998.

[8] STRANG, G.: *Introduction to linear algebra*, second ed., Wellesley and Cambridge Press, 1998.

[9] WILKINSON, J.H.: *Rounding errors in algebraic processes*, Prentice-Hall, 1963.

Caroline N. Haddad

Dept. Math. State Univ. New York at Geneseo

1 College Circle

Geneseo, NY 14454, USA

E-mail address: haddad@geneseo.edu

MSC2000: 15-XX, 65-XX, 90-XX

Key words and phrases: symmetric, ill-conditioned, well-conditioned, positive definite, matrix decomposition or factorization, LU-decomposition, least squares, banded, bandwidth, normal equation, pivot, pivoting, singular.

COMBINATORIAL MATRIX ANALYSIS

Broadly speaking, *matrix analysis* is the study of nonalgebraic properties of matrices and the analysis of matrices in order to reveal their finer properties and structure. (Here, 'algebraic' is understood at least in the classical sense of algebra.) The matrices are usually real or complex matrices. *Combinatorial matrix analysis* is the study of combinatorial properties of matrices and the analysis of matrices which takes into account combinatorial structure. Here combinatorial structure usually refers to the *zero-nonzero pattern* of a matrix, captured through the use of either the directed graph or bipartite graph associated with the nonzero entries of a matrix (or the graph of the nonzero entries in the case of a symmetric matrix), or to the *positive-negative-zero pattern* of a real matrix, captured through the use of the signed digraph or signed bipartite graph of a matrix.

This article is intended as an introduction to combinatorial matrix analysis. More detail can be found in [5] and [6], and the references contained therein. We do not discuss here the many applications of matrix theory and linear algebra to combinatorics, graphs, and discrete structures.

Matrix Patterns and Various Graphs. Let $A = [a_{ij}]$ be a matrix of order n whose entries a_{ij} are real or complex numbers. To A there corresponds a *directed graph* (or *digraph*) $D(A)$ with vertex set $V = \{1, \ldots, n\}$ and with an arc (i, j) from vertex i to vertex j if and only if $a_{ij} \neq 0$. The *bipartite graph* (or *bigraph*) $BG(A)$ of A has vertex set $\{1_r, \ldots, n_r\}$ (corresponding to the rows of A) and $\{1_c, \ldots, n_c\}$ (corresponding to the columns of A); the edges of $BG(A)$ are all pairs $\{i_r, j_c\}$ for which $a_{ij} \neq 0$. The bipartite graph of a matrix can be defined for a rectangular $m \times n$ matrix in the same way except that the vertices corresponding to the rows are $\{1_r, \ldots, m_r\}$. Both the digraph and the bigraph reveal the zero-nonzero pattern of a square matrix A.

If A is a real matrix and we want to capture the sign $(+, -, 0)$ of the entries of A, then we assign a $+$ or $-$ to each arc of $D(A)$ (to each edge of $BG(A)$) according as the corresponding entry of A is positive or negative, and in this way obtain the *signed digraph* and *signed bigraph* of A. We use the same notations $D(A)$ and $BG(A)$ for the signed versions of the digraph and bigraph of A. Thus two matrices A and B have the same *sign pattern* if and only if they have the same signed digraphs (equivalently, the same signed bigraphs).

If A is a symmetric matrix (or has a symmetric pattern in the sense that $a_{ij} \neq 0$ if and only if $a_{ji} \neq 0$), then the *graph* $G(A)$ of A has vertex set $\{1, \ldots, n\}$ with an edge $\{i, j\}$ between i and j if and only if $a_{ij} \neq 0$ (equivalently, $a_{ji} \neq 0$). Thus $G(A)$ is obtained from $D(A)$ by 'removing'

the directions on arcs (this may result in two edges joining certain pairs of vertices and one edge of each such pair is removed as well). Sometimes in $D(A)$ and $G(A)$ it is convenient to ignore the arcs (i,i) and edges $\{i,i\}$ (called loops) corresponding to nonzero entries a_{ii} on the main diagonal of A.

A square matrix A of order n is *irreducible* provided there does not exist a permutation matrix P such that

$$PAP^\top = \begin{bmatrix} A_1 & O \\ A_{21} & A_2 \end{bmatrix},$$

where A_1 is a square matrix of order k for some k with $0 < k < n$. (The matrix $PAP^\top$ is obtained from A by simultaneously permuting its rows and columns. The digraphs of A and $PAP^\top$ are isomorphic.) A digraph is *strongly connected* provided for each ordered pair of distinct vertices i and j there is a path from i to j.

PROPOSITION 1 The matrix A is irreducible if and only if the digraph $D(A)$ is strongly connected, [5]. □

A square matrix can be brought to a very special form by simultaneous row and column permutations.

THEOREM 2 Let A be a matrix of order n. Then there exist a permutation matrix P and an integer $k \geq 1$ such that

$$PAP^\top = \begin{bmatrix} A_1 & O & \cdots & O \\ A_{21} & A_2 & \cdots & O \\ \vdots & \vdots & \ddots & \vdots \\ A_{k1} & A_{k2} & \cdots & A_k \end{bmatrix},$$

where $A_1, \ldots, A_k$ are square, irreducible matrices. The matrices $A_1, \ldots, A_k$ are uniquely determined to within simultaneous permutations of their rows and columns, but their order on the diagonal is not necessarily unique. □

The matrices $A_1, \ldots, A_k$ in this theorem are called the *irreducible components* of A and correspond to the *strongly connected components* of the digraph $D(A)$.

Irreducible matrices have an *inductive structure* that is revealed in the next theorem [5].

THEOREM 3 Let A be an irreducible matrix of order $n \geq 2$. Then there exist a permutation matrix P and an integer $m \geq 2$ such that

$$PAP^\top = \begin{bmatrix} A_1 & O & \cdots & O & E_1 \\ E_2 & A_2 & \cdots & O & O \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ O & O & \cdots & A_{m-1} & O \\ O & O & \cdots & E_m & A_m \end{bmatrix},$$

where $A_1, \ldots, A_m$ are irreducible matrices and $E_1, \ldots, E_m$ are matrices having at least one nonzero entry. □

Allowing independent row and column permutations in the definition of irreducibility leads to full indecomposability. A square matrix A of order n is *fully indecomposable* provided there do not exist permutation matrices P and Q such that

$$PAQ = \begin{bmatrix} A_1 & O \\ A_{21} & A_2 \end{bmatrix},$$

where A_1 is a square matrix of order k for some k with $0 < k < n$. The matrices A and PAQ have isomorphic bigraphs.

Theorems analogous to Theorems 2 and 3 hold with independent permutations replacing simultaneous permutations and fully indecomposable replacing irreducible. The connection is provided by the fact that a square matrix A is fully indecomposable if and only if there are permutation matrices P and Q such that PAQ has a nonzero main diagonal and PAQ is irreducible [5].

Eigenvalues and Digraphs. The following theorem is the Perron–Frobenius theorem [7], [8] and is one of the first instances of the influence of the digraph of a matrix on its spectral properties.

THEOREM 4 (*Perron–Frobenius theorem*) Let A be a matrix of order $n > 1$ each of whose entries is a nonnegative real number. Assume that A is irreducible, equivalently $D(A)$ is strongly connected. Then there is a positive number $\rho(A)$ such that

1) $\rho(A)$ is a simple eigenvalue of A;
2) every eigenvalue λ of A satisfies $|\lambda| \leq \rho(A)$;
3) the number of eigenvalues λ of A with $|\lambda| = \rho(A)$ equals the greatest common divisor k of the lengths of the circuits of $D(A)$, and these eigenvalues are $\rho(A)e^{2\pi ij/k}$, $(j = 1, \ldots, k)$.

□

A more recent application of the digraph of a matrix to localization of its eigenvalues concerns a generalization [2] of the *Gershgorin theorem*. Let $A = [a_{ij}]$ be a complex matrix of order n and let

$$R_i = \sum_{j \neq i} |a_{ij}| \qquad (1 \leq i \leq n).$$

Then Gershgorin's theorem asserts that the n eigenvalues of A lie in that part $\mathcal{R}$ of the complex plane determined by the union of the n closed disks

$$\{z\colon\ |z - a_{ii}| \leq R_i\}, \qquad (1 \leq i \leq n).$$

If A is irreducible, then a boundary point of $\mathcal{R}$ is an eigenvalue of A only if it is a boundary point of each of the n closed disks.

By considering the *circuits*, or *directed cycles*, of the digraph of A, a better inclusion region can be obtained.

THEOREM 5 Let $A = [a_{ij}]$ be a complex matrix of order n. Then the n eigenvalues of A lie in that part $\mathcal{S}$ of the complex plane determined by the union of the regions

$$\mathcal{S}(\gamma) = \left\{ z\colon \prod_{\gamma} |z - a_{ii}| \leq \prod_{\gamma} R_i \right\},$$
$$(\gamma \text{ a circuit of } D(A)),$$

where $\prod_\gamma$ denotes the product over all vertices i belonging to the circuit γ. If A is irreducible, then a boundary point of $\mathcal{S}$ is an eigenvalue of A only if it is a boundary point of each $\mathcal{S}(\gamma)$. □

Theorems 4 and 5 demonstrate how information concerning the combinatorial structure of a matrix can be used to give information on spectral properties of the matrix.

Sign-Nonsingular Matrices. It is easy to characterize real matrices $A = [a_{ij}]$ of order n whose singularity is a consequence of their zero pattern, equivalently, of their nonzero pattern or bigraph $BG(A)$. Let $\mathcal{Z}(A)$ denote the set of all real matrices B of order n that have the same zero pattern as A, that is, satisfy $BG(B) = BG(A)$. Then the following are equivalent:

i) Each matrix $B \in \mathcal{Z}(A)$ is singular;

ii) Each of the $n!$ terms in the standard determinant expansion of A is zero (the *standard determinant expansion of a matrix* A is

$$\det A = \sum \epsilon(i_1, \ldots, i_n) a_{1i_1} \cdots a_{ni_n}$$

where the sum extends over each permutation $i_1 \ldots i_n$ of $\{1, \ldots, n\}$ and $\epsilon(i_1 \ldots i_n)$ is $+$ or $-$ depending on whether the permutation is even or odd);

iii) The bigraph $BG(A)$ does not have a perfect matching (i.e. a set of n pairwise vertex disjoint edges meeting all vertices);

iv) There is a set of fewer than n rows and columns which together contain all the nonzero entries of A;

v) There is a set of fewer that n vertices of $BG(A)$ which together meet all the edges of $BG(A)$.

Properties ii) and iii) are clearly equivalent, as are properties iv) and v). Properties i) and ii) are equivalent, since if there is a nonzero term in the standard determinant expansion of A, then by sufficiently emphasizing the entries of A in that term we obtain a nonsingular matrix. Properties iii) and iv) are equivalent by the Frobenius–König theorem [5].

Now let $\mathcal{Q}(A)$ denote the set of all real matrices of order n that have the same sign pattern $(+, -, 0)$ as A. $\mathcal{Q}(A)$ consists of all real matrices of order n that have the same signed digraph (equivalently, the same signed bigraph) as A and is called the *qualitative class* of A. The matrix A is called *sign-nonsingular* provided each matrix in $\mathcal{Q}(A)$ is nonsingular. Some equivalent characterizations of sign-nonsingularity are:

i) A is sign-nonsingular;

ii) There is a nonzero term in the standard determinant expansion of A and each such nonzero term has the same sign;

iii) $\det(A) \neq 0$ and the determinants of the matrices in $\mathcal{Q}(A)$ all have the same sign.

The matrix

$$\begin{bmatrix} 1 & -1 & 0 \\ 1 & 1 & -1 \\ 1 & 1 & 1 \end{bmatrix}$$

is a sign-nonsingular matrix.

If a matrix $A = [a_{ij}]$ is sign-nonsingular, then so is every matrix PAQ where P and Q are permutation matrices, as is every matrix of the form DA

where D is a nonsingular diagonal matrix. Also a sign-nonsingular matrix must have a nonzero term in its standard determinant expansion. Thus in dealing with sign-nonsingular matrices we may assume that each entry on the main diagonal is negative, that is, A has a *negative main diagonal*. With this normalization, we have the following theorem [1]. The *sign of a circuit*

$$\gamma\colon j_1 \to \cdots \to j_k \to j_1$$

of the signed digraph of A is

$$\operatorname{sign}(\gamma) = \operatorname{sign} a_{j_1 j_2} \cdots a_{j_{k-1} j_k} a_{j_k j_1},$$

the products of the signs of the arcs of the cycle.

THEOREM 6 (*Bassett–Maybee–Quirk theorem*) Let A be a real matrix of order n with a negative main diagonal. Then A is a sign-nonsingular matrix if and only if each circuit of the signed digraph of A is negative. □

Sign-nonsingularity allows one to characterize square, homogeneous systems of linear equations $Ax = 0$ for which $\widetilde{A}x = 0$ has only the zero solution (thus all solutions of $\widetilde{A}x = 0$ have the same sign pattern) for all matrices $\widetilde{A}$ with the same sign pattern as A. A more general problem is to characterize linear systems $Ax = b$ that are sign-solvable in the sense that the sign pattern of the solution is determined solely by the sign patterns of A and b. More precisely, $Ax = b$ is *sign-solvable* provided that for all $\widetilde{A} \in \mathcal{Q}(A)$ and all $\widetilde{b} \in \mathcal{Q}(b)$ there is a vector $\widetilde{x}$ such that $\widetilde{A}\widetilde{x} = \widetilde{b}$ and all of the vectors in

$$\left\{\widetilde{x}\colon \exists \widetilde{A} \in \mathcal{Q}(A), \widetilde{b} \in \mathcal{Q}(b) \text{ s.t. } \widetilde{A}\widetilde{x} = \widetilde{b}\right\}$$

have the same sign pattern.

Sign-solvable linear systems can be characterized in terms of two classes of matrices, called S^*-matrices and L-matrices. An $n \times (n+1)$ matrix B is an *S^*-matrix* provided each matrix of order n obtained from B by deleting a column is a sign-nonsingular matrix. Cramer's rule implies that B is an S^*-matrix if and only if there is a vector w with no zero coordinates such that the right null spaces of the matrices in $\widetilde{B} \in \mathcal{Q}(B)$ are contained in $\{0\} \cup \mathcal{Q}(w) \cup \mathcal{Q}(-w)$. The matrix

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & 1 & 1 & -1 \end{bmatrix}$$

is an S^*-matrix. A matrix A is an *L-matrix* provided every matrix in $\mathcal{Q}(A)$ has linearly independent rows. Sign-nonsingular matrices are square L-matrices. Every S^*-matrix is an L-matrix and so is any matrix obtained from an L-matrix by appending columns.

The following theorem characterizes sign-solvable linear systems [9].

THEOREM 7 Let $A = [a_{ij}]$ be an $m \times n$ matrix and let b be an $m \times 1$ column vector. Let $z = (z_1, \ldots, z_n)^\top$ be a solution of the linear system $Ax = b$, and let

$$\beta = \{j\colon z_j \neq 0\},$$
$$\alpha = \{i\colon a_{ij} \neq 0 \quad \text{for some } j \in \beta\}.$$

Then $Ax = b$ is sign-solvable if and only if the matrix

$$[A[\alpha, \beta] - b[\beta]]$$

is an S^*-matrix and the matrix $A(\alpha, \beta)^\top$ is an L-matrix. □

(Here $A[\alpha, \beta]$, respectively $A(\alpha, \beta)$, is the submatrix of A formed by the rows in α and the columns in β, respectively not in α and not in β, and $b[\alpha] = b[\alpha, \{1\}]$.)

A detailed study of sign-solvability and related issues is contained in [6].

Doubly Stochastic Matrices. A real matrix $A = [a_{ij}]$ of order n is *doubly stochastic* provided each of its entries is nonnegative, and all row and column sums equal 1:

$$a_{ij} \geq 0 \quad (i, j = 1, \ldots, n),$$
$$\sum_{j=1}^{n} a_{ij} = 1, \quad \sum_{i=1}^{n} a_{ij} = 1.$$

Doubly stochastic matrices arise quite naturally in many different contexts:

i) Let $U = [u_{ij}]$ be a real orthogonal matrix or a complex unitary matrix. Then

$$\widehat{U} = [|u_{ij}|^2] \qquad (i, j = 1, \ldots, n)$$

is a doubly stochastic matrix.

ii) (*Optimal assignment problem*) Consider an assignment of n people to n positions in which the 'value' of the ith person to the jth position is $v_{ij} \geq 0$ $(i, j = 1, \ldots, n)$. An *opti-*

mal assignment is an assignment $i \to j_i$ $(i = 1, \ldots, n)$ of people to positions (here $j_1 \ldots j_n$ is a permutation of $\{1, \ldots, n\}$) which maximizes the total value $\sum_{i=1}^{n} v_{ij_i}$. The set Ω_n of doubly stochastic matrices of order n is a convex polytope and, according to *Birkhoff's theorem* [5], the set of vertices of this polytope is the set $\mathcal{P}_n$ of permutation matrices of order n. Thus the vertices of Ω_n correspond to the $n!$ possible assignments, and the optimal assignment problem can be solved as a linear programming problem on Ω_n.

iii) Let $\mathcal{P}_n = \{P_1, \ldots, P_{n!}\}$, and let $(c_i : i = 1, \ldots, n!)$ be a probability distribution on $\mathcal{P}_n$: $c_i \geq 0$ $(i = 1, \ldots, n!)$ and $\sum_{i=1}^{n!} c_i = 1$. Then the expectation of a permutation $R \in \mathcal{P}_n$ chosen at random is

$$E = E[R] = \sum_{i=1}^{n!} c_i P_i = [e_{ij}],$$

a doubly stochastic matrix. It is a consequence of *Birkhoff's theorem* that every doubly stochastic matrix of order n arises from a probability distribution on $\mathcal{P}_n$ in this way. The probability that a function f chosen at random according to the probabilities

$$\text{prob}(f(i) = j) = e_{ij} \qquad (i, j = 1, \ldots, n)$$

is a permutation equals the *permanent* of A defined by

$$\text{per}(A) = \sum a_{1j_1} \cdots a_{nj_n},$$

where the sum extends over all permutations $j_1 \ldots j_n$ of $\{1, \ldots, n\}$.

Let $A = [a_{ij}]$ be a real, symmetric matrix (or a complex, Hermitian matrix). Then there exists a real, orthogonal matrix U such that

$$UAU^{\top} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix},$$

where $\lambda_1 \geq \ldots \geq \lambda_n$ are the n eigenvalues of A. Comparing diagonal entries, we get

$$(\lambda_1, \ldots, \lambda_n)^{\top} = S(a_{11}, \ldots, a_{nn})^{\top}, \qquad (1)$$

where $S = \widehat{U}$ is a doubly stochastic matrix. Without loss of generality, assume that $a_{11} \geq \cdots \geq a_{nn}$. Then equation (1) implies that

$$\lambda_1 + \cdots + \lambda_i \leq a_{11} + \cdots + a_{ii} \quad (i = 1, \ldots, n), \qquad (2)$$

with equality for $i = n$. When the inequalities (2), with equality for $i = n$, hold between two vectors $\lambda = (\lambda_1, \ldots, \lambda_n)$ and $\mu = (a_{11}, \ldots, a_{nn})$ (that have been arranged in nonincreasing order), then λ is said to be *majorized* by μ. A *Hardy-Littlewood-Pólya theorem* states that if λ is majorized by μ, then there exists a doubly stochastic matrix S such that $\lambda = S\mu$ [10]. Hence by Birkhoff's theorem, λ is majorized by μ if and only if λ is in the convex hull of all vectors obtained from μ by permuting its coordinates. There exist doubly stochastic matrices S of very special form such that $\lambda = S\mu$ when λ is majorized by μ [10], [4].

As noted above, the vector of eigenvalues of a real, symmetric matrix is majorized by the vector of its entries on the main diagonal. Conversely, if λ and μ are two n-vectors with λ majorized by μ, then according to a theorem of A. Horn, there exists a real, symmetric matrix of order n, whose eigenvalues are given by λ and whose main diagonal entries are given by μ [10].

Let A be a doubly stochastic matrix, and let $A_1, \ldots, A_k$ $(k \geq 1)$ be the fully indecomposable components of A. Since all row and column sums of A equal 1, it follows easily that up to row and column permutations A is the direct sum of its fully indecomposable components: there exist permutation matrices P and Q such that

$$PAQ = A_1 \oplus \cdots \oplus A_k,$$

where $\oplus$ denotes direct sum. The polytope Ω_n has dimension $(n-1)^2$. Each doubly stochastic matrix determines a face of Ω_n equal to the set of all doubly stochastic matrices S such that $BG(S)$ is a subgraph of $BG(A)$ (i.e. each edge of $BG(A)$ is also an edge of $BG(A)$). This face is the smallest face of Ω_n containing A, and each nonempty face of Ω_n arises in this way. Since no entry of a doubly stochastic matrix can exceed 1, the nonempty faces of Ω_n can be described as follows: Let C be a $(0,1)$-matrix of order n which, up to row and column permutations, is a direct sum of fully indecomposable matrices (such matrices are said to have *total support*). Then

$$\mathcal{F}(C) = \{A\colon A \in \Omega_n, A \leq C \text{ entrywise}\}$$

is a face of Ω_n and its dimension equals $\sigma(C) - 2n + k$, where $\sigma(C)$ is the number of 1s of C and k is the number of its fully indecomposable components [3].

See also: **Fractional combinatorial optimization; Replicator dynamics in combinatorial optimization; Neural networks for combinatorial optimization; Combinatorial optimization algorithms in resource allocation problems; Multi-objective combinatorial optimization; Combinatorial optimization games; Evolutionary algorithms in combinatorial optimization.**

References

[1] BASSETT, L., MAYBEE, J., AND QUIRK, J.: 'Qualitative economics and the scope of the correspondence principle', *Econometrica* **36** (1968), 544–63.

[2] BRUALDI, R.A.: 'Matrices, eigenvalues and directed graphs', *Linear Multilinear Algebra* **11** (1982), 143–65.

[3] BRUALDI, R.A., AND GIBSON, P.M.: 'The convex polytope of doubly stochastic matrices I: Applications of the permanent function', *J. Combin. Th. A* **22** (1977), 194–230.

[4] BRUALDI, R.A., AND HWANG, S.-G.: 'Vector majorization via Hessenberg matrices', *J. London Math. Soc. Ser. 2* **53** (1996), 28–38.

[5] BRUALDI, R.A., AND RYSER, H.J.: *Combinatorial matrix theory*, Vol. 39 of *Encycl. Math. Appl.*, Cambridge Univ. Press, 1991.

[6] BRUALDI, R.A., AND SHADER, B.L.: *Matrices of sign-solvable linear systems*, Vol. 116 of *Cambridge Tracts in Math.*, Cambridge Univ. Press, 1995.

[7] HORN, R., AND JOHNSON, C.R.: *Matrix analysis*, Cambridge Univ. Press, 1985.

[8] HORN, R., AND JOHNSON, C.R.: *Topics in matrix analysis*, Cambridge Univ. Press, 1991.

[9] KLEE, V., LADNER, R., AND MANBER, R.: 'Sign solvability revisited', *Linear Alg. & Its Appl.* **59** (1984), 131–57.

[10] OLKIN, I., AND MARSHALL, A.W.: *Inequalities: Theory of majorization and its applications*, Acad. Press, 1979.

Richard A. Brualdi
Dept. Math. Univ. Wisconsin
Madison, WI 53706, USA
E-mail address: brualdi@math.wisc.edu

MSC2000: 90C10, 90C09
Key words and phrases: matrix analysis.

COMBINATORIAL OPTIMIZATION ALGORITHMS IN RESOURCE ALLOCATION PROBLEMS

The *resource allocation problem* seeks to find an optimal allocation of a fixed amount of resources to activities so as to minimize the cost incurred by the allocation. A simplest form of the problem is to minimize a separable convex function under a single constraint concerning the total amount of resources to be allocated. The amount of resources to be allocated to each activity is treated as a continuous or integer variable, depending on the cases. This can be viewed as a special case of the nonlinear programming problem or the nonlinear integer programming problem.

Due to its simple structure, this problem is encountered in a variety of application areas, including load distribution, production planning, computer resource allocation, queueing control, portfolio selection, and apportionment. The first explicit investigation of the resource allocation problem is due to B.O. Koopman [15] (1953), who dealt with the problem of the *optimal distribution of efforts*, which arises in the problem of searching for an object whose position is a random variable. Since then, a great number of papers have been published on resource allocation problems. Efficient algorithms have also been developed, depending on the form of objective functions and constraints or on the type of variables (i.e., continuous or integer).

See [11] for a comprehensive review of the state-of-the-art of the problems (as of 1988). After this book was published, many papers have been published on resource allocation problems. A significant progress has been made on the algorithm side. Also, new generalizations and variants of the problem have been investigated, and new application fields have been discovered. Such new progress has been reviewed in [13].

We first classify the resource allocation problems. A generic form of the resource allocation problem discussed in this article is described as follows:

$$(\mathrm{P}) \quad \begin{cases} \min & f(x_1, \ldots, x_n) \\ \text{s.t.} & \sum_{j=1}^{n} x_j = N, \\ & x_j \geq 0, \quad j = 1, \ldots, n. \end{cases} \tag{1}$$

That is, given one type of resource whose total amount is equal to N, we want to allocate it to n activities so that the objective value $f(x_1, \ldots, x_n)$ is minimized. The objective value may be interpreted as the cost or loss, or the profit or reward, incurred by the resulting allocation. In case of profit or reward, it is natural to maximize f, and we shall sometimes consider maximization problems. The difference between maximization and minimization is not essential because maximizing f is equal to minimizing $-f$.

Each variable x_j represents the amount of resource allocated to activity j. If it represents persons, processors or trucks, however, variable x_j becomes a discrete variable that takes nonnegative integer values, and the constraint

$$x_j : \text{integer}, \quad j = 1, \ldots, n, \tag{2}$$

is added to the constraints in (1). The resource allocation problem with this constraint is often referred to as the *discrete resource allocation problem.*

As for the objective function, it usually has some special structure according to the intended applications. Typically, the following special case, called *separable*, is often considered:

$$\sum_{j=1}^{n} f_j(x_j). \tag{3}$$

If each f_j is convex, the objective function is called *separable convex objective function.*

Resource allocation problems are classified according to the types of objective functions, constraints and variables. We shall describe the classification scheme, and several types of problem formulations according to the classification scheme. In general, we use the notation $\alpha/\beta/\gamma$ to denote the type of a resource allocation problem. Here, α specifies the type of objective function, β the constraint type, and γ the variable type;

$$\gamma = D, \quad \gamma = C$$

denote the case of integer variable, respectively continuous variable. We shall now explain the notations for α and β.

α: Objective functions. The objective function $f(x_1, \ldots, x_n)$ may take the following special structures:

1) *Separable* (S, for short): $\sum_{j=1}^{n} f_j(x_j)$, where each f_j is a function of one variable.
2) *Separable and convex* (SC, for short): $\sum_{j=1}^{n} f_j(x_j)$, where each f_j is a convex function of one variable. In particular, if each f_j is quadratic and convex, we denote such a subclass by SQC.
3) *Minimax*: minimize $\max_{1 \leq j \leq n} f_j(x_j)$, or *Maximin*: maximize $\min_{1 \leq j \leq n} f_j(x_j)$; here, all f_j are monotone nondecreasing in x_j.
4) *Lexicographically minimax* (Lexico-Minimax, for short): Since the objective value of Minimax is determined by the single variable x_k^* satisfying $f_k(x_k^*) = \max_j f_j(x_j^*)$, there may be many optimal solutions. To remove such ambiguity, we introduce the *lexicographical ordering for n-dimensional vectors*: Given $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$, a is *lexicographically smaller* than b (or b is *lexicographically greater* than a) if $a_j = b_j$ for $j = 1, \ldots, k-1$ and $a_k < b_k$ some k. This is denoted by $a \leq_{\text{lex}} b$ or $b \geq_{\text{lex}} a$. For $a = (a_1, \ldots, a_n)$, let $\mathrm{DEC}(a)$ (respectively, $\mathrm{INC}(a)$) denote the n-tuple of a_j, $j = 1, \ldots, n$, arranged in nonincreasing order (respectively, nondecreasing order) of their values (e.g., for $a = (4, 3, 1, 5)$, we have $\mathrm{DEC}(a) = (5, 4, 3, 1)$ and $\mathrm{INC}(a) = (1, 3, 4, 5)$). The objective of Lexico-Minimax is to find an allocation vector $x = (x_1, \ldots, x_n)$ such that $\mathrm{DEC}(x)$ is minimal. Notice that an optimal solution to Lexico–Minimax is also optimal to Minimax, but the converse is not generally true. This is a refined objective of Minimax. Similarly, we define Lexico-Maximin as the one that maximizes $\mathrm{INC}(x)$.
5) *Fair*: minimize the expression

$$g\left(\max_{1 \leq j \leq n} f_j(x_j), \min_{1 \leq j \leq n} f_j(x_j)\right),$$

where $g(u,v)$ is nondecreasing (respectively, nonincreasing) in u (respectively, v). This objective is a generalization of Minimax and Maximin.

β: Constraints. In addition to the simple first resource constraint of (1), other additional constraints are also imposed. Typical additional constraints which appeared in various resource allocation problems are as follows. We refer the case of no additional constraints as 'simple'.

1) *Lower and upper bounds* (LUB, for short): $l_j \leq x_j \leq u_j$, $j = 1, \ldots, n$.
2) *Generalized upper bounds* (GUB, for short): $\sum_{j \in S_i} x_j \leq b_i$, $i = 1, \ldots, m$, where $S_1, \ldots, S_m$ is a partition of $\{1, \ldots, n\}$.
3) *Nested constraints* (Nested, for short): $\sum_{j \in S_i} x_j \leq b_i$, $i = 1, \ldots, m$, where $S_1 \subset \cdots \subset S_m$. We can assume $b_1 \leq \cdots \leq b_m$, since if $b_i > b_{i+1}$, the ith constraint is redundant.
4) *Tree constraints* (Tree, for short): $\sum_{j \in S_i} x_j \leq b_i$, $i = 1, \ldots, m$, where the sets S_i are derived by some hierarchical decomposition of E into disjoint subsets.
5) *Network constraints* (Network, for short): The constraint is defined in terms of a directed network with a single source and multiple sinks. Given a directed graph $G = (V, A)$ with node set V and arc set A, let $s \in V$ be the source and $T \subseteq V$ be the set of sinks. The amount of supply from the source is $N > 0$, and the capacity of arc (u,v) is $c(u,v)$. Denote the flow vector by $\varphi = \{\varphi(u,v) \colon (u,v) \in A\}$. φ is a *feasible flow* in G if it satisfies

$$0 \leq \varphi(u,v) \leq c(u,v), \quad (u,v) \in A, \tag{4}$$

$$\sum_{(v,w) \in A_+(v)} \varphi(v,w) - \sum_{(u,v) \in A_-(v)} \varphi(u,v) = 0,$$

$$v \in V - T - \{s\}, \tag{5}$$

$$\sum_{(s,v) \in A_+(s)} \varphi(s,v) - \sum_{(u,s) \in A_-(s)} \varphi(u,s) = N, \tag{6}$$

$$x_t(\varphi) \tag{7}$$

$$\equiv \sum_{(u,t) \in A_-(t)} \varphi(u,t) - \sum_{(t,v) \in A_+(t)} \varphi(t,v) \geq 0,$$

$$t \in T,$$

$$\sum_t x_t(\varphi) = N. \tag{8}$$

The value $x_t(\varphi)$ denotes the amount of flow entering a sink $t \in T$. For a feasible flow φ, the vector $\{x_t(\varphi) \colon x \in T\}$ is called the *feasible flow vector* with respect to φ. For instance, the problem SC/Network/C (i.e., the separable convex resource allocation problem under network constraints) is defined as follows:

$$\begin{cases} \min & \sum_{t \in T} f_t(x_t(\varphi)) \\ \text{s.t.} & (4) - 8, \end{cases} \tag{9}$$

where f_t, for each $t \in T$, is a convex function.

6) *Submodular constraints* (SM, for short): A set of feasible solutions is defined by a base polyhedron $B(r) = \{x \in \mathbf{R}^E \colon x(S) \leq r(S)$ for all $S \in \mathcal{D}$, $x(E) = r(E)\}$ of a submodular system $(\mathcal{D}, r)$, i.e.,

$$x \in B(r). \tag{10}$$

Here, we use the notation $E = \{1, \ldots, n\}$, and $x(S) \equiv \sum_{i \in S} x_i$ for $S \subseteq E$ and $x \in \mathbf{R}^E$. $\mathcal{D} \subseteq 2^E$ is a *distributive lattice* such that $\emptyset, E \in \mathcal{D}$, i.e., $\mathcal{D}$ is closed under union and intersection operations. Also, the function $r \colon \mathcal{D} \to \mathbf{Z}$ is *submodular* over $\mathcal{D}$, i.e.,

$$r(X) + r(Y) \geq r(X \cup Y) + r(X \cap Y).$$

For a *submodular system* $(\mathcal{D}, r)$,

$$P(r) = \{x \in \mathbf{R}^E \colon x(S) \leq r(S) \text{ for all } S \in \mathcal{D}\}$$

is called the *submodular polyhedron* of $(\mathcal{D}, r)$.

Notice that the first constraint in (1) is included in the constraints of (10), as $x(E) = r(E)$ in the above definition. If we consider the case of integer variables, the constraint is defined by

$$x \in B(r) \cap \mathbf{Z}^E.$$

It is assumed, in general, that $B(r)$ of the constraint (10) is not explicitly given as an

input, but is implicitly given through an *oracle* that tells the value $r(X)$ when X is given.

7) *General linear constraints* (Linear, for short): Constraints defined by a set of linear inequalities

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i, \quad i = 1, \ldots, m. \tag{11}$$

No other special assumption is imposed on the structure of the constraints.

Notice that all the constraints, LUB, GUB, Nested, Tree, Network are special cases of submodular constraints (see [11]), and SM is a special case of Linear.

Algorithms. We first introduce an *incremental algorithm* for the simple resource allocation problem, SC/Simple/D. We assume that each f_j is defined over the interval $[0, N]$. Since f_j is convex, we have

$$d_j(1) \leq \cdots \leq d_j(N), \tag{12}$$

where

$$d_j(y_j) = f_j(y_j) - f(y_j - 1).$$

The incremental algorithm is a kind of *greedy algorithm*, and is also called a *marginal allocation* method. Starting with the initial solution $x = (0, \ldots, 0)$, one unit of resource is allocated at each iteration to the most favorable activity (in the sense of minimizing the increase in the current objective value) until $\sum x_j = N$ is attained.

Input: An instance of SC/Simple/D. Output: An optimal solution x^*.

Let $x := (0, \ldots, 0)$ and $k := 0$;
WHILE $k < N$ DO
Find j^* such that
$d_{j^*}(x_{j^*} + 1) = \min_{1 \leq j \leq n} d_j(x_j + 1)$;
$x_{j^*} := x_{j^*} + 1$;
$k := k + 1$
END;
Output x as x^*.

Procedure INCREMENT.

It has been shown this procedure correctly computes an optimal solution in $O(N \log n + n)$ time.

Several polynomial time algorithms have been developed for problem SC/Simple/D [3], [7], [14]. The fastest among them is proposed in [3]. Its running time is $O(\max\{n, n\log(N/n)\})$, but the algorithm is very complicated. All these algorithms are based on *divide-and-conquer*.

The incremental algorithm presented above also works for problem SC/SM/D. In this case, among all the elements such that $x + e(j) \in P(r)$ (i.e., feasible except for the constraint $(x + e(j))(E) = r(E)$), the x_j with the minimum increase in $f_j(x_j)$ is incremented by one. This process is repeated until $x(E) = r(E)$ is finally attained.

A polynomial time algorithm for SC/SM/D is also known [2], [4]. It first solves a problem of SC/Simple/D type, which is obtained from the original problem by by considering only the simple constraint $x(E) = r(E)$ but disregarding the rest. If the obtained solution y is feasible, we are done, i.e., it is an optimal solution of the original problem. Otherwise, the problem is decomposed into subproblems using the information obtained from the vector y and the submodular constraints.

When specialized to problem SC/Network/D, the running time becomes $O(|T|(\tau(n, m, C_{\max}) + |T| \log(N/|T|)))$, where $\tau(n, m, C_{\max})$ denotes the running time for the maximum flow algorithm for a graph with n vertices, m arcs and the maximum arc capacity $C_{\max}$. The direct consequence of this result is that problems SC/GUB/D, SC/Nested/D and SC/Tree/D can be solved in $O(n^2 \log(N/n))$ time. For SC/Nested/D, the running time was improved to $O(n \log n \log(N/n))$ in [8]. The idea of the improvement is based on a general and beautiful proximity theorem between integral and continuous optimal solutions for SC/SM/D and SC/SM/C.

For SQC/−/D with − equal to Simple, GUB, Nested, Tree or Network, D.S. Hochbaum and S. Hong [9] developed improved algorithms based on proximity result between SQC/−/C and SQC/−/D and efficient algorithms for SQC/−/C.

Minimax/−/D and Maximin/−/D, are equivalently transformed into problems of SC/−/D. Therefore, equally efficient algorithms can be developed for minimax and maximin problems. The transformation is done as follows: We only show this fact for the most general case, i.e., Minimax/SM/D, which are described as follows:

$$\text{MINIMAX} \begin{cases} \min & \max\limits_{j \in E} f_j(x_j), \\ \text{s.t.} & x \text{ is an integral base of } B(r). \end{cases}$$

Here, all f_j, $j \in E$, are assumed to be nondecreasing. When all f_j are nonincreasing, problems MINIMAX and MAXIMIN are mutually transformed into MAXIMIN and MINIMAX, respectively, by the following identities:

$$-\min_x \max_{j \in E} f_j(x_j) = \max_x \min_{j \in E} -f_j(x_j),$$

$$-\max_x \min_{j \in E} f_j(x_j) = \min_x \max_{j \in E} -f_j(x_j).$$

Define for $j \in E$,

$$g_j(x_j) = \sum_{y=0}^{x_j} f_j(y), \quad x_j = 0, 1, \ldots . \tag{13}$$

Note that

$$g_j(x_j) - g_j(x_j - 1) = f_j(x_j)$$

holds for each $x_j = 0, 1, \ldots$. From the nondecreasingness of f_j, it follows that g_j is convex over the nonnegative integers. Now consider the following problems of SC/SM/D:

$$Q_g : \min \left\{ \sum_{j \in E} g_j(x_j) : x \in B(r) \cap Z^E \right\}.$$

It is then shown that an optimal solution of problem Q_g is optimal to MINIMAX.

Generalizations. We finally note a recent development. K. Ando, S. Fujishige and T. Naitoh [1], [6] considered the separable convex resource allocation problem for a *bisubmodular system* and for a *finite jump system*, whose underlying constraint can be viewed as a generalization of the submodular constraint. They developed greedy algorithms for such problems. For the case of a bisubmodular system, a polynomial time algorithm has been given in [5]. Also, Hochbaum and J.G. Shanthikumar [10] showed that, for a class of general linear constraints, efficient algorithms can be developed. The running time of their algorithm depends on the maximum absolute value of the subdeterminants, Δ, and if $\Delta = 1$ (i.e., the constraint matrix is *totally unimodular*), the running time becomes polynomial. The idea is based on the proximity result between the integral and continuous optimal solutions. When $\Delta = 1$, V.V. Karzanov and S.T. McCormick [12] proposed another polynomial time algorithm.

In addition to these efforts to generalize the constraints, new progress has recently been made towards generalizing objective functions for which efficient algorithms can still be developed. This research was done by K. Murota [16], [17] who identified a subclass of nonseparable convex functions, M-convex functions, which is defined on the base polyhedron of a submodular system as follows.

A function $f : \mathbf{Z}^E \to \mathbf{R} \cup \{\infty\}$ is said to be *M-convex* if it satisfies the following property:

- (M-EXC): For any $x, y \in \operatorname{dom} f$ and for any $i \in \operatorname{supp}^+(x - y)$, there exists a $j \in \operatorname{supp}^-(x - y)$ such that

$$f(x) + f(y) \geq f(x - e(i) + e(j)) + f(y + e(i) - e(j)),$$

where

$$\operatorname{dom} f = \{x \in \mathbf{Z}^E : f(x) < +\infty\},$$

$$\operatorname{supp}^+(x - y) = \{k \in E : x_k > y_k\},$$

$$\operatorname{supp}^-(x - y) = \{k \in E : x_k < y_k\}.$$

The M-convex functions can enjoy nice theorems of *discrete convex analysis* in a parallel manner to the traditional convex analysis. A polynomial time algorithm has been developed for this class of problems [18].

See also: **Optimizing facility location with rectilinear distances; Single facility location: Multi-objective Euclidean distance location; Single facility location: Multi-objective rectilinear distance location; Single facility location: Circle covering problem; Multifacility and restricted location problems; Network location: Covering problems; Warehouse location problem; Facility location with externalities; Production-distribution system design problem; Global optimization in Weber's problem with attraction and repulsion; Facility location with staircase costs; Stochastic transportation and location problems; Facility location problems with spatial interaction; Voronoi diagrams in facility location; MINLP: Application in facility location-allocation; Resource allocation for epidemic control; Competitive fa-**

cility location; Fractional combinatorial optimization; Replicator dynamics in combinatorial optimization; Neural networks for combinatorial optimization; Combinatorial matrix analysis; Multi-objective combinatorial optimization; Combinatorial optimization games; Evolutionary algorithms in combinatorial optimization; Simple recourse problem.

References

[1] ANDO, K., FUJISHIGE, S., AND NAITOH, T.: 'A greedy algorithm for minimizing a separable convex function over an integral bisubmodular polyhedron', *J. Oper. Res. Soc. Japan* **37** (1994), 188–196.

[2] FEDERGRUEN, A., AND GROENEVELT, H.: 'The greedy procedure for resource allocation problems - necessary and sufficient conditions for optimality', *Oper. Res.* **34** (1986), 909–918.

[3] FREDERICKSON, G.N., AND JOHNSON, D.B.: 'The complexity of selection and ranking in $X+Y$ and matrices with sorted columns', *J. Comput. Syst. Sci.* **24** (1982), 197–208.

[4] FUJISHIGE, S.: 'Lexicographically optimal base of a polymatroid with respect to a weight vector', *Math. Oper. Res.* **21** (1980), 186–196.

[5] FUJISHIGE, S.: 'A min-max theorem for bisubmodular polyhedra', *SIAM J. Discrete Math.* **10** (1997), 294–308.

[6] FUJISHIGE, S., AND NAITOH, T.: 'A greedy algorithm for minimizing a separable convex function over a finite jump system', *J. Oper. Res. Soc. Japan* **38** (1995), 362–375.

[7] GALIL, Z., AND MEGIDDO, N.: 'A fast selection algorithm and the problem of optimum distribution of effort', *J. ACM* **26** (1979), 58–64.

[8] HOCHBAUM, D.S.: 'Lower and upper bounds for the allocation problem and other nonlinear optimization problems', *Math. Oper. Res.* **19** (1994), 390–409.

[9] HOCHBAUM, D.S., AND HONG, S.: 'About strongly polynomial time algorithms for quadratic optimization over submodular constraints', *Math. Program.* **55** (1995), 269–309.

[10] HOCHBAUM, D.S., AND SHANTHIKUMAR, J.G.: 'Nonlinear separable optimization is not much harder than linear optimization', *J. ACM* **37** (1990), 843–862.

[11] IBARAKI, T., AND KATOH, N.: *Resource allocation problems: Algorithmic approaches*, MIT, 1988.

[12] KARZANOV, A.V., AND MCCORMICK, S.T.: 'Polynomial methods for separable convex optimization in totally unimodular linear spaces with applications', *SIAM J. Comput.* **26** (1997), 1245–1275.

[13] KATOH, N., AND IBARAKI, T.: 'Resource allocation problems', in D.-Z. DU AND P.M. PARDALOS (eds.): *Handbook Combinatorial Optim.*, Vol. 2, Kluwer Acad. Publ., 1998.

[14] KATOH, N., IBARAKI, T., AND MINE, H.: 'A polynomial time algorithm for the resource allocation problem with a convex objective function', *J. Oper. Res. Soc.* **30** (1979), 159–260; 449–455.

[15] KOOPMAN, B.O.: 'The optimum distribution of effort', *Oper. Res.* **1** (1953), 52–63.

[16] MUROTA, K.: 'Convexity and Steinitz's exchange property', *Adv. Math.* **124** (1996), 272–311.

[17] MUROTA, K.: 'Discrete convex analysis', *Math. Program.* **83** (1998), 313–371.

[18] SHIOURA, A.: 'Minimization of an M-convex function', *Discrete Appl. Math.* **84** (1998), 215–220.

Naoki Katoh
Kyoto Univ.
Kyoto, Japan
E-mail address: naoki@archi.kyoto-u.ac.jp

MSC2000: 90C09, 90C10
Key words and phrases: resource allocation, combinatorial algorithm.

COMBINATORIAL OPTIMIZATION GAMES, *CRPM*

In combinatorial optimization games, we consider *cooperative games* for which the value of the game is obtained via a *combinatorial optimization* problem. For a cooperative game (a class of *games with side payments*), the set of participating players is denoted by N and a value $v(S)$ is achieved by each subset S of players without any help from other players (in the set $N-S$). Usually, we set $v(\emptyset)=0$. In general, the representation of the game requires an input size exponential in the number of players. For a combinatorial optimization game, however, the value $v(S)$ is often succinctly defined as a solution to a combinatorial optimization problem for which the combinatorial structure is determined by the subset S of players. The income distributed to individual player i is represented by x_i, $1 \leq i \leq N$, and $x=(x_1,\ldots,x_N)$.

The main issue in cooperative games is how to fairly distribute the income collectively earned by the whole group of players in the game, cooperating with each other. For simplicity, let $x(S)=\sum_{i\in S} x_i$. The income vector x is called an *imputation* if $x(N)=v(N)$, and $\forall i \in N$: $x_i \geq v(\{i\})$ (*individual rationality*). Additional requirements may be added to ensure fairness, stability and ratio-

nality. And they lead to different sets of income vectors which are generally referred to as solution concepts. Among many of these solution concepts, the core, which consists of all the imputations satisfying the subgroup rationality condition $\forall S \subseteq N$: $x(S) \geq v(S)$, is naturally defined and has attracted much attention from researchers. It has also led to many fruitful results in combinatorial optimization games. Our focus in this article will be on the core. Readers interested in other solution concepts for cooperative games in general can find them in many *game theory* books and survey papers. For example, [12] gives an interesting discussion for several classical solution concepts in cooperative games and their applications to political economy.

Recently, *computational complexity* has been suggested as another metric for evaluating the rationality of these solution concepts [2]. In this argument, computational complexity is suggested as a measure of bounded rationality [13] for players not to spend *super-polynomial time* to search for the most suitable solution. For combinatorial optimization games, N. Megiddo [7] suggested that algorithms polynomial in the number of players (as good algorithms following the concept introduced by J. Edmonds [3]) be sought for solutions. As the value of any subset of players is defined as the optimal solution to a combinatorial optimization problem, the input size can often be restricted to be bounded by a polynomial in the number of players. This is usually the case for many practical collective optimization problems. The value of a subgroup of players is the optimal objective function value that this subgroup can achieve under the constraints imposed by resources controlled by players in the subgroup. Very often the collective optimization problem requires an integer solution. It is under this context the game is then referred to as a *combinatorial optimization game*.

An example to formulate a two-sided market (the assignment game) is given in [11]. The underlying structure is a bipartite graph $(V_1, V_2; E)$. One interpretation given by L.S. Shapley and M. Shubik is that V_1 is the set of sellers, and V_2 is the set of buyers. For the simplest case, each seller has an item (say a house) to sell and each buyer wants to purchase an item. The ith seller, $i \in V_1$, values its item at c_i dollars and the jth buyer values the item of the ith seller at h_{ij} dollars. Between this pair, we may define a value $v(\{i,j\}) = h_{ij} - c_i$ if $h_{ij} \geq c_i$ and set (i,j) an edge in E with weight $v(\{i,j\})$. Otherwise, there is no edge between i and j since no deal is possible if the seller values the item more than the buyer does. Considering a game with side-payment, the value $v(S)$ of a subset S of buyers and sellers is defined to be the weight of maximum matching in the bipartite graph $G[S]$ induced by the corresponding set S of vertices (an edge is in $G[S]$ if and only if its two end vertices are both in S). In a linear programming formulation, this is

$$\begin{cases} v(S) = \max & \displaystyle\sum_{(i,j)\in E} v(i,j)x_{ij} \\ \text{s.t.} & \displaystyle\sum_{i\in V_1\cap S} x_{ij} \leq 1 \\ & \displaystyle\sum_{j\in V_2\cap S} x_{ij} \leq 1 \\ & x \geq 0. \end{cases}$$

Shapley and Shubik have shown that the core for this assignment game is precisely the set of solutions for the dual program of the above linear program with $S = V_1 \cup V_2$. Such nice properties are not unusual in combinatorial optimization games. For example, the same fact is established for another game, a cost allocation game on trees, by A. Tamir. Tamir has shown that the core is exactly the set of optimal solutions to the dual program of the linear program formulation for the total cost of the cost allocation problem on trees [14].

The Shapley–Shubik model is a theoretical formulation for a *pure exchange economy*. The linear production game of G. Owen [8] applies their ideas to a production economy. In Owen's model, each player j ($j \in N$) owns a resource vector, b^j. For a subset S of players, their value is the objective function value of the optimal solution for the following linear program:

$$\begin{cases} \max & c^\top y \\ \text{s.t.} & Ay \leq \displaystyle\sum_{j\in S} b^j \\ & y \geq 0. \end{cases} \tag{1}$$

Thus, the value is what the subset of players can achieve in the linear production model with the resources under their control. The core for the lin-

ear production game is always nonempty [8] if all the above linear programs have finite optimum. A constructive proof presented by Owen obtains an imputation in the core from any optimal solution of the dual program

$$\begin{cases} \min & \sum_{j\in N} w^\top b^j \\ \text{s.t.} & w^\top A \geq c^\top \end{cases}$$

of the linear program for all the players

$$\begin{cases} \max & c^\top y \\ \text{s.t.} & Ay \leq \sum_{j\in N} b^j. \end{cases}$$

In fact, let w be the optimal solution for the dual program. Set $x_j = w^\top b^j$, $j \in N$. Then $x = (x_1, \ldots, x_N)$ is an imputation in the core. To see so, for each subset $S \subseteq N$, consider $x(S) = \sum_{i\in S} x_i$. By definition of x, we have $x(S) = \sum_{i\in S} w^\top b^i$. Let y_S^* be the optimal solution for the linear program for $v(S)$. Then, $Ay_S^* \leq \sum_{j\in S} b^j$. Therefore, $x(S) \geq w^\top Ay_S^*$. On the other hand, $w^\top A \geq c^\top$. It follows that $x(S) \geq c^\top y_S^*$, which is the same as $x(S) \geq v(S)$ since $v(S) = c^\top y_S^*$.

Notice that this proof depends on the fact that, if for each $S \subseteq N$, the linear program (1) has a finite optimal value. In general, a linear program may be unbounded or infeasible. If for any $S \subseteq N$, the linear program (1) is unbounded, obviously the core does not exist. If it is infeasible, we may define $v(S) = -\infty$. This allows the extension of the above result to the case when the following conditions are satisfied:

1)

$$\begin{cases} \max & c^\top y \\ \text{s.t.} & Ay \leq \sum_{j\in N} b^j \\ & y \geq 0 \end{cases}$$

has a finite optimal value.

2) For each $S \subseteq N$, 1) has a finite optimal value or is infeasible.

However, unlike the assignment game, there may in general be imputations in the core which cannot be obtained from the dual program for Owen's linear production game [8]. In general, it is not known how to decide whether an imputation is in the core in polynomial time.

There is a weakness in applying the linear production game model to the studies of coalition optimization problems. That is, in reality, many variables are required to be of integer values. It happens that for the assignment game of Shapley and Shubik, the linear production model of Owen's always results in an integer solution. There are, however, many other situations for which the integer optimal solution cannot be obtained in the framework of the linear program.

A generalized linear production model introduced by D. Granot retains the main linear program structure of Owen's model but allows right-hand sides of the resource constraints not to be linear in the resource vectors b^j of individual players [6]. Thus, $v(S)$ is defined to be $\max\{c^\top y\colon Ay \leq b(S), y \geq 0\}$, where $b(S) = (b_1(S), \ldots, b_m(S))$ is a general function of S. It is shown that, if for each i, $1 \leq i \leq N$, the game consisting of player set N with value function $b_i(S)$ has a nonempty core, the generalized linear production game has a nonempty core. As the game of Owen's model, an imputation in the core is constructed from the optimal solution for the dual program and vectors in the core associated with (N, b_i) [6]. This would in general need an exponential number of function values $b(S)$ for all the subset S of N. For some collective combinatorial optimization problems, $b(S)$ is given implicitly as a solution to some optimization problem and thus the problem input size is polynomially bounded. The extended power of Granot's model can be applied to prove nonemptiness for the cores of many games beyond those of Owen's linear production game.

In particular, the generalized linear production game model is applied to show the nonemptiness of a certain minimum cost spanning tree game [6]. In this problem, we have a complete graph as the underlying structure. A cost is assigned to each edge. There is a distinguished node 0. Players are vertices $\{1, \ldots, n\}$. The cost $c(S)$ of a subset S of players is defined to be the cost of minimum spanning tree in the graph $G[S \cup \{0\}]$ induced by $S \cup \{0\}$. (Notice that the cost game is different from the value game defined as above but can be handled similarly.) Even though an imputation in

the core can be found in polynomial time for this game, in [4] it is shown that it is *NP*-hard to decide whether an imputation is not in the core.

Another way to extend Owen's model to include games of combinatorial optimization nature is to explicitly require integer solutions in the definition of the linear production model. That is, one may define game value $v(S)$ for a subset $S \subseteq N$ to be the maximum value of an integer program instead of a linear program. Therefore,

$$v(S) = \max \left\{ c^\top x \colon Ax \leq \sum_{j \in S} b^j,\ x \text{ integers} \right\}.$$

For the assignment game of Shapley and Shubik and the cost allocation game on trees of Tamir, the integer program can be solved by its linear program relaxation, since there is always an integer solution for the latter. In the work of Shapley and Shubik, as well as that of Tamir, b^j is a unit vector and $b(N)$ is a vector of all ones. It is this particular structure of linear constraints that makes the core to be identified with the set of optimal solutions for the dual linear program to the linear program of the game value for the set of players [11], [14]. It is no wonder this property is further exploited in [5] for a partition game, and in [1] for packing/covering games.

The *packing game*, for example, is defined for a set N of players whose game value is given by the following integer program

$$\begin{cases} \max & c^\top x \\ \text{s.t.} & x^\top A_{M,N} \leq 1^N \\ & x \in \{0,1\}^m, \end{cases}$$

where 1^N is a vector of $|N|$ ones, and $A_{M,N}$ is a 0–1 matrix of rows indexed by M and columns indexed by N. For each subset S of players, its value is given by

$$\begin{cases} \max & c^\top x \\ \text{s.t.} & x^\top A_{M,S} \leq 1^\top_{|S|}, \quad x^\top A_{M,\overline{S}} \leq 0^\top_{n-|S|}, \\ & x \in \{0,1\}^m, \end{cases}$$

where $A_{M,S}$ is the submatrix of A with row set M and column set S, $\overline{S} = N - S$ and $v(\emptyset)$ is defined to be 0.

The covering game and the partition game are defined similarly. It is a necessary and sufficient condition for the core of the packing (and covering, and partitioning) game to be nonempty that the linear relaxation of the corresponding optimization problem always has an integer optimal solution. In additional, the core, if nonempty, is exactly the set of optimal solutions to the dual program of the linear relaxation of the corresponding integer program [5], [1].

These results allow for a characterization of combinatorial structures for the corresponding combinatorial optimization game to have a nonempty core. Because of the linear program characterization of the core, questions such as whether the core is empty or not, whether we can find an imputation in the core, and whether an imputation is in the core, can often be determined in polynomial time. Notice that, there are cases that the linear program may be of exponential size in the number of players, it is not immediate that all these questions can be solved in polynomial time. But even for cases when there are an exponential number of constraints, the linear program may be solvable in polynomial time [9].

First established by Shapley and Shubik for the assignment game, the connection of the core for a combinatorial optimization game with dual program of the linear program relaxation has been a successful tool in the characteration of the core, design and analysis of algorithms to find an imputation in the core and to test membership of an imputation in the core. It is expected that this approach would continously lead to fruitful results in cooperative game theory.

See also: **Fractional combinatorial optimization**; **Replicator dynamics in combinatorial optimization**; **Neural networks for combinatorial optimization**; **Combinatorial matrix analysis**; **Multi-objective combinatorial optimization**; **Evolutionary algorithms in combinatorial optimization**.

References

[1] Deng, X., Ibaraki, T., and Nagamochi, H.: 'Algorithmic aspects of the core of combinatorial optimization games', *Math. Oper. Res.* **24**, no. 3 (1999), 751–766.

[2] Deng, X., and Papadimitriou, C.: 'On the complexity of cooperative game solution concepts', *Math. Oper. Res.* **19**, no. 2 (1994), 257–266.

[3] EDMONDS, J.: 'Paths, tree, and flowers', *Canad. J. Math.* **17** (1965), 449–469.

[4] FAIGLE, U., FEKETE, S., HOCHSTÄTTLER, W., AND KERN, W.: 'On the complexity of testing membership in the core of min-cost spanning tree games', *Internat. J. Game Theory* **26** (1997), 361–366.

[5] FAIGLE, U., AND KERN, W.: 'Partition games and the core of hierarchically convex cost games', *Memorandum Fac. Toegepaste Wiskunde Univ. Twente* **1269**, no. June (1995).

[6] GRANOT, D.: 'A generalized linear production model: A unified model', *Math. Program.* **34** (1986), 212–222.

[7] MEGIDDO, N.: 'Computational complexity and the game theory approach to cost allocation for a tree', *Math. Oper. Res.* **3** (1978), 189–196.

[8] OWEN, G.: 'On the core of linear production games', *Math. Program.* **9** (1975), 358–370.

[9] SCHRIJVER, A.: *Theory of linear and integer programming*, Wiley, 1986.

[10] SHAPLEY, L.S.: 'On balanced sets and cores', *Naval Res. Logist. Quart.* **14** (1967), 453–460.

[11] SHAPLEY, L.S., AND SHUBIK, M.: 'The assignment game', *Internat. J. Game Theory* **1** (1972), 111–130.

[12] SHUBIK, M.: 'Game theory models and methods in political economy', in K.J. ARROW AND M.D. INTRILIGATOR (eds.): *Handbook Math. Economics*, Vol. I, North-Holland, 1981, pp. 285–330.

[13] SIMON, H.: 'Theories of bounded rationality', in R. RADNER (ed.): *Decision and Organization*, North-Holland, 1972.

[14] TAMIR, A.: 'On the core of cost allocation games defined on location problems', *Preprints, Second Internat. Conf. Locational Decisions (ISOLDE 81) (Skodsborg, Denmark)* (1981), 387–402.

Xiaotie Deng
City Univ. Hong Kong
Kowloon, Hong Kong SAR, China
E-mail address: deng@cs.cityu.edu.hk

MSC2000: 91A12, 90C27, 90C60

Key words and phrases: combinatorial optimization, cooperative game, complexity, algorithms.

COMMUNICATION NETWORK ASSIGNMENT PROBLEM, *CAP*

In the *communication network assignment problem* (CAP) a system of communication centers $C_1, \ldots, C_n$ is given. The centers have to be embedded into a given (undirected) network $N = (V, E)$ with vertex set V, $|V| = n$, and edge set E. The centers exchange messages at given rates per time unit through a selected routing pattern. Let t_{ij} be the amount of messages sent from center C_i to center C_j per time unit. If there is no direct connection between C_i and C_j the messages sent from C_i to C_j pass through several intermediate centers. The messages exchanged between C_i and C_j may be sent along a single path or they may be split into several parts, each part being sent along its own path. For any fixed embedding $\mathcal{E}$ of the centers into the network and for any fixed routing pattern ρ of the messages, let $IMT_{\mathcal{E},\rho}(C_i)$ denote the overall amount of traffic going through the center C_i as intermediate center. The goal is to find an embedding $\mathcal{E}$ of the centers into the network and a routing pattern ρ which minimizes the maximum intermediate traffic over all centers:

$$\min_{\mathcal{E},\rho} \max \{IMT_{\mathcal{E},\rho}(C_i) \colon 1 \leq i \leq n\} \qquad (1)$$

A typical application of the problem arises in the case of locating stations (terminals, computers) in a *local-area computer network* (LAN) as described by T.B. Boffey and J. Karkazis [1]. Usually, a given segment of the LAN serves different pairs of communicating stations. In order to prevent interference and garbled messages, only one message at a time can be sent through a given segment of the LAN. On the other hand one has to restrict the offered traffic through the same segment so as to maintain a reasonable throughput in the network. To this end it is reasonable to locate *bridges* at the endpoints of each segment. All bridges will work as intermediate centers and all stations will work as bridges. The result is that each pair of stations (or bridges) communicates through its own segment. It is reasonable to require an embedding of stations and additional bridges into the nodes of the LAN such that the intermediate traffic going through the busier station (or bridge) is minimized. Boffey and Karkazis [1] proposed and discussed also a continuous version of the problem.

A similar problem, the so-called *elevator problem* leads also to the optimization problem (1) as described by Karkazis [5]. The elevator problem arises when a single elevator has to be replaced by two elevators, each covering contiguous subsets of floors. It might be reasonable to place the connecting landing so as to minimize the traffic intensity on the busier elevator. More specifically assume that the first elevator serves floors $\{1, 2, \ldots, i\}$ and the second elevator serves floors $\{i, i+1, \ldots, n\}$, and let t_{ij} represent the traffic intensity from floor

i to floor j. Then the traffic load of the first elevator is given as $T_i^{(1)} = \sum_{k=1}^{i} \sum_{l=1}^{n} (t_{kl} + t_{lk})$ and the traffic load of the second elevator is given as $T_i^{(2)} = \sum_{k=i}^{n} \sum_{l=1}^{n} (t_{kl} + t_{lk})$. Then we want to choose i so as to minimize $\max\{T_i^{(1)}, T_i^{(2)}\}$. Obviously this problem setting can be generalized for more than two elevators.

Essentially there are two distinct models of routing patterns in (1): the *single path* model and the *fractional* model. In the single path model, for every pair of communication centers C_i and C_j, a single route in the network is selected and all t_{ij} messages are sent along this fixed route. In the fractional model, the amount t_{ij} is split into a number of positive parts and every part is sent along its own path. Most of the results available in the literature concern the CAP on *trees*. In this case, for each pair of vertices in the network there is only one path to join them and hence, both models coincide.

R.E. Burkard, E. Çela and G.J. Woeginger have proved in [3] that in general the CAP is *NP*-hard. More specifically has been shown that the CAP is *NP*-hard for networks that are i) paths; ii) stars of branch length three; iii) cycles (*NP*-hardness in both models); or iv) doublestars (*NP*-hardness in the single path model). Moreover, it has been proved in [3] that the CAP is polynomially solvable in the case of stars of branch length two and in the case of doublestars in the fractional model. In the case of a star of branch length two the CAP can be formulated as a maximum weight perfect matching problem (MWPMP) if the communication center to be assigned to the central node of the star is kept fixed. Since there are only n possibilities for the selection of the center to be placed at the central node, one just has to solve n MWPMPs. In the fractional model, finding an embedding of the communication centers into the nodes of the network and a routing pattern which minimize the intermediate traffic can be done by solving a specified number of linear programs with $O(P)$ variables and $O(n^2)$ constraints each, where P is the number of pairwise disjoint paths in N. In the case of doublestars $P = O(n^2)$ and there are $O(n^2)$ programs to be solved (see [3]). This implies that in this case the CAP is polynomially solvable in the fractional model.

Some *exact algorithms* and *heuristic approaches* to solve the CAP on trees have been proposed in [3], [5]. Karkazis has proposed a *branch and bound* algorithm in the case where N is a path [5], and Burkard, Çela and Woeginger [3] have proposed a branch and bound approach in the case that N is a tree. The algorithms have been tested on randomly generated trees and communication rates t_{ij}. The tests show that only small instances of the CAP of size up to 12 can be solved in reasonable time. For large instances the number of the branched nodes in the branch and bound tree explodes. In order to approximately solve larger instances of the CAP on trees Burkard, Çela and T. Dudàs proposed in [2] *simulated annealing* and *tabu search* approaches. The proposed heuristics are tested on randomly generated instances of size up to 32. The comparison of the heuristic solutions with the optimal solution produced by the branch and bound algorithm for instances of small size shows that the performance of these heuristics is quite satisfactory.

Finally, in [2] the *asymptotic behavior* of the CAP on trees has been investigated. Under natural probabilistic assumptions on the problem data the CAP on trees shows a very interesting behavior: The ratio between the maximum and the minimum values of the objective function, i.e., the ratio between the maximum and the minimum values of the intermediate traffic through the busiest center, approaches 1 with probability tending to 1 as the size of the problem tends to infinity. The proof of this fact is based on the strong relationship between the CAP-T and a special version of the *quadratic assignment problem*. It is shown that the latter fulfills the condition of a theorem of Burkard and U. Fincke [4] on the asymptotic behavior of combinatorial optimization problems. From a practical point of view the asymptotic behavior described above implies that the CAP on trees becomes trivial as its size tends the infinity: every feasible solution provides a good approximation of an optimal solution.

See also: **Minimum cost flow problem; Nonconvex network flow problems; Traffic network equilibrium; Network location: Covering problems; Maximum flow prob-**

lem; Shortest path tree algorithms; Steiner tree problems; Equilibrium networks; Survivable networks; Directed tree networks; Dynamic traffic networks; Auction algorithms; Piecewise linear network flow problems; Generalized networks; Evacuation networks; Network design problems; Stochastic network problems: Massively parallel solution; Frequency assignment problem; Bi-objective assignment problem; Assignment and matching; Assignment methods in clustering; Quadratic assignment problem; Maximum partition matching.

References

[1] BOFFEY, T.B., AND KARKAZIS, J.: 'Location of transfer centers on segments of a communication network with proportional traffic', *J. Oper. Res. Soc.* **40** (1989), 729–734.

[2] BURKARD, R.E., ÇELA, E., AND DUDÀS, T.: 'A communication assignment problem on trees: heuristics and asymptotic behavior': *Network optimization*, Vol. 450 of *Lecture Notes Economics and Math. Systems*, Springer, 1996, pp. 127–156.

[3] BURKARD, R.E., ÇELA, E., AND WOEGINGER, G.J.: 'A minimax assignment problem in treelike communication networks', *Europ. J. Oper. Res.* **87** (1995), 670–684.

[4] BURKARD, R.E., AND FINCKE, U.: 'Probabilistic asymptotic properties of some combinatorial optimization problems', *Discrete Appl. Math.* **12** (1985), 21–29.

[5] KARKAZIS, J.: 'A minimax assignment problem on a linear communication network', *Belgian J. Oper. Res., Statist. and Computer Sci.* **33** (1993), 5–17.

Rainer E. Burkard
Technical Univ. Graz
Steyrergasse 30
A-8010 Graz, Austria
E-mail address: burkard@opt.math.tu-graz.ac.at

MSC2000: 90B80, 90C05, 90C27, 68Q25

Key words and phrases: optimization, assignment problem, communication network, computational complexity, exact algorithms, heuristic approaches, asymptotic behavior.

COMPETITIVE FACILITY LOCATION

Facility location models deal, for the most part, with the location of plants, warehouses, distribution centers and other industrial facilities. These location models do not account for competition or for differences among facilities and therefore allocate customers to facilities by proximity. In reality, retail facilities operate in a competitive environment with an objective of profit and market share maximization. These facilities are also different from each other in their overall attractiveness to consumers. One branch of location analysis focuses on the location of retail and other commercial facilities which operate in a competitive environment, namely competitive facility location. The basic problem is the optimal location of one or more new facilities in a market area where competition already exists or will exist in the future. Assuming that profit increases when market share increases, maximizing profit is equivalent to maximizing market share. It follows, then, that the location objective is to locate the retail outlet at the location that maximizes its market share.

A unique feature of competitive facility location models is facility attractiveness (its appeal to consumers). Facilities differ in the total 'bundle of benefits' they offer customers. They vary in one or more of the attributes which make up their total attractiveness to customers. Furthermore, varying importance assigned to each of these attributes by different customers will result in a selective set of consumers patronizing each. Facility attractiveness level, therefore, needs to be incorporated in the location model. Facility attractiveness needs to first be assessed using one of a variety of methods. Once attractiveness is assumed known, market share captured can be calculated. Facility attractiveness is estimated using a utility function (a composite index of attractiveness) or some other measure (floor area) serving as a surrogate for a latent attractiveness. Utility models are predicated on consumer spatial choice models as well as on the premise that facilities of the same type are not necessarily comparable.

Also unique to competitive facility location is the modeling of demand in terms of buying power. Income levels and discretionary spending become a measure of demand. For a review of competitive models see [4], [9].

The underlying theme running through all competitive models is the existence of an interrelationship between four variables: buying power(demand), distance, facility attractiveness, and market share, with the first three variables being independent variables and the last the depen-

dent variable. Buying power, or effective buying income, is known (for example, *Sales and Marketing Management*). Distance from demand points to facilities can be measured. The most difficult link in the interrelationship between the four variables is the determination of facility attractiveness. As is mentioned above and discussed below, it is estimated using a utility function. Once buying power, distance, and attractiveness are known, market share can be calculated.

Proximity Model. The first modern paper on competitive facility location is generally agreed to be H. Hotelling's paper on duopoly in a linear market [15]. Hotelling considered the location of two competing facilities on a segment (for example, two ice-cream vendors along a beach strip). The distribution of buying power along the segment is assumed uniform and customers patronize the closest facility. When one facility is located and there is no competition, all customers patronize the existing facility. However, when a competing facility is introduced and is located at a different point on the segment, the customers on one side of the midpoint between the two facilities patronize one facility and the customers on the other side of the midpoint patronize the second facility. If one facility is held fixed in place, the best location for the second is either immediately left or right of the fixed one, depending on which segment — left or right of the existing facility — is longer. In models based on Hotelling's formulation it is assumed that customers patronize the closest facility.

Location-Allocation Model. An extension to Hotelling's approach is the location-allocation model for the selection of sites for facilities that serve a spatially dispersed population. Both the facilities' locations and the allocation of customers to them are determined simultaneously. The allocation of customers to facilities is made using Hotelling's proximity assumption — each facility attracts the consumers closest to it. The market share attracted by each facility is calculated and the best locations for the new facilities are then found. Multifacility location-allocation models analyze the system-wide interactions among all facilities. C. ReVelle [22] introduced location-allocation models to competitive location. M.F. Goodchild [13] suggested the location-allocation market share model (MSM). A retail firm is planning to open a chain of outlets in a market in which a competing chain already exists. The entering firm's goal is to maximize the total market share captured by the entire chain. Most location-allocation solution methods rely on heuristic approaches that do not guarantee an optimal solution, rather they provide good solutions for implementation. The best locations are selected from a user-provided, prespecified set of potential sites. Typically, these problems are formulated on a network and the location solution is on a node. See [12] for a collection of papers on the subject. A comprehensive review of location-allocation models can be found in [11].

The assumption that customers patronize the facility closest to them implies that the competing facilities are equally attractive. For equally attractive facilities, the plane is partitioned by a Voronoi diagram [20], [21]. It is implicitly assumed that all customers located at a demand point patronize the same facility. This, in turn, implies an 'all or nothing' property. The combined buying power at a demand point is assigned entirely to one facility and none is assigned to other facilities, unless two or more facilities are equidistant.

Deterministic Utility Model. When the facilities are not equally attractive, the proximity premise for allocating consumers to facilities is no longer valid. To account for variations in facility attractiveness, a deterministic utility model for competitive facility location is introduced in [2]. Hotelling's approach is extended by relaxing the proximity assumption. Consumers are known to make their choice of a facility based on factors other than distance alone. Therefore, it is assumed that customers base their choice of a facility on facility attractiveness which is represented by a utility function. This utility function is a composite index of facility attributes and the distance to the facility, representing the expected satisfaction from that facility (either an additive or a multiplicative utility function). It is generally agreed that customers, through a decision-making process, choose the facility with the highest utility, the facility which is expected to maximize their satisfaction.

This choice is determined by some formula according to which customers evaluate alternative facilities' attributes weighted by their personal salience to arrive at an overall facility attractiveness.

A trade-off between distance and attractiveness takes place. Based on this premise the degree of expected satisfaction with each alternative as a function of the relevant characteristics of that facility is measured. It is suggested that a customer will patronize a better and farther facility as long as the extra distance to it does not exceed its attractiveness advantage. For example, paramedics transporting a motor vehicle accident victim will bypass a nearby hospital in favor of a farther, better equipped trauma center as long as the difference in quality of care exceeds the adverse effect to the patient caused by the extra distance and time delay. A break-even distance is defined. At the break-even distance the attractiveness of two competing facilities is equal. This break-even distance, therefore, is the maximum distance that a customer will be willing to travel to a farther facility (new or existing) based on his perception of its attractiveness and advantage relative to other facilities. All customers at a demand point will patronize the new facility if it is located within the break-even distance. While customers are no longer assumed to patronize the closest facility, customers at a certain demand point are assumed to apply the same utility function, therefore, they all patronize the same facility. The 'all or nothing' property is maintained in this extension.

Based on aggregated utility values for existing facilities and a utility function for a new facility, the best location is found for the new one. The optimal location for the new facility is sensitive to its attractiveness. Different attractiveness levels may yield different optimal locations.

Random Utility Model. To address the 'all or nothing' assumption of the deterministic utility model and to account for variations in individual utility functions, a random utility model is introduced in [6]. The deterministic utility model is extended by assuming that each customer draws his utility from a random distribution of utility functions. The probability that a customer will prefer a certain facility over all other facilities is calculated by applying the multivariate normal distribution. Once the probabilities are calculated, the market share captured by a particular facility (new or existing) can be calculated as a weighted sum of the buying power at all demand points. This formulation eliminates the 'all or nothing' property since a probability that a customer will patronize a particular facility can be established and is no longer either 0% or 100%. To circumvent the mathematically complicated formulation of the random utility model, [8] suggested using the simpler logit model. The probability that a customer will patronize a facility as a function of the distance to that facility, can be approximated by a logit function of the distance.

Gravity Based Models. An alternative approach to the location of competing facilities, based on the gravity model, was introduced by D.L. Huff [16], [17] and is extensively used by marketers. According to the gravity model two cities attract retail trade from an intermediate town in direct proportion to the populations of the two cities and in inverse proportion to the square of the distances from them to the intermediate town. Huff proposed that the probability that a consumer patronizes a retail facility is proportional to its size (floor area) and inversely proportional to a power of the distance to it. Facility size, or square footage, is a surrogate for facility attractiveness. Huff depicted equi-probability lines. A customer located on such a line between two facilities patronizes the two facilities with equal probability. These equi-probability lines divide the region into catchment areas, each dominated by a facility, in a manner similar to the Voronoi diagram [20]. These lines do not define an 'all or nothing' assignment of customers to facilities, rather, at any demand point, the proportion of consumers attracted to each facility is a function of its square footage (attractiveness) and distance. The model finds the market share captured at each potential site, and thus the best location for new facilities whose individual measures of attractiveness are known.

Suppose there are k existing facilities and n demand points. The attractiveness of facility j is A_j for $j = 1, \ldots, k$, and the distance between demand point i and facility j is d_{ij}. The buying power at

demand point i is b_i. Therefore, the proportion of the buying power (market share) M_j attracted by facility j is:

$$M_j = \sum_{i=1}^{n} b_i \frac{A_j/d_{ij}^{\lambda}}{\sum_{m=1}^{k} A_m/d_{im}^{\lambda}},$$

where λ is the power to which distances are raised.

In the original Huff formulation, facility floor area serves as a surrogate for attractiveness. A major improvement on Huff's approach was suggested by M. Nakanishi and L.G. Cooper [19] who introduced the *multiplicative competitive interaction* (MCI) model. The MCI coefficient replaces the floor area with a product of factors, each a component of attractiveness. Each factor in the product is raised to a power. Thus, the attractiveness of a facility is a composite of a set of attributes rather than the floor area alone. Nakanishi and Cooper's idea was elaborated on and applied in [18] to food retailing using specific attractiveness attributes. Gravity based models suggest the evaluation in terms of market share of a user provided discrete set of potential sites for the location of a new facility.

Huff's and the Nakanishi–Cooper models were extended to the location of multiple facilities in [1], [10]. See [1] for an extension of the MCI model to the location of multiple facilities which belong to the same chain. The problem was modeled as a nonlinear integer programming problem and a random search procedure combined with an interchange heuristic was employed to identify optimal and near-optimal sets of locations. A. Ghosh and C.S. Craig [10] proposed a franchise distribution model. An expanding franchise seeks to maximize sales while minimizing canibalization of franchise outlets. This model was also formulated as a nonlinear integer programming problem but included additional factors such as advertising. These two models select the best locations from a user-provided set of alternative sites as well.

Other papers [14], [23] suggest variations on Huff's formulation by replacing the distance raised to a power with an exponent of the distance. This formulation accelerates the distance decay.

Finding the best location for a new facility (or multiple facilities) in a continuous space using the gravity model objective is discussed in [3] for the single facility case, and in [5] for the location of multiple facilities.

Anticipating Future Competition. The competitive facility location models discussed above are myopic and short-term oriented in that they attempt to find the optimal location for a new facility (facilities) by maximizing current market share against existing competition. A different approach to competitive location focuses on anticipating and preempting future competition. It is assumed that a new competing facility will enter the market at some point in the future. The competitor will establish his facility at the location which maximizes his market share. Therefore, one's present location decision will affect the competitor's location decision. Conversely, assuming a future competitive entry has implications for one's present location decision. The objective is to find the location that maximizes the market share captured by one's own facility following the competitor's entry. This problem is known in the economic literature as the *Stackelberg equilibrium problem* or the *leader-follower problem* and as the Simpson problem in voting theory. See [7] for a review of the topic.

Conclusions. There are two main applications for competitive facility location models. The first application is the location analysis of a new facility. The best location for the new facility, based on market share maximization at that location, is found. The second application is an analysis of the impact of changes in quality in existing facilities (either own's, competitor's, or both) on the market share captured by one's facility and on its optimal location. In addition, a decision maker will be able to perform a 'what-if analysis' and anticipate the impact on his facility of either competitor's improvements or of the introduction of a new facility. In this case one needs to know the overall attractiveness of the proposed new facility or the difference in overall attractiveness pre-post improvements in an existing one. Using the models, a decision maker can assess:

- the impact on location of changes in attractiveness for his new facility;

- the impact on market share of change in location for his new facility;
- the impact on market share of changes in attractiveness at his existing facility (facilities);
- the impact on his facility of changes in other facilities or the introduction of a new facility.

These models afford the anticipation and analysis of the impact of likely future scenarios. In a highly competitive market such as exists domestically, and in the face of increasing global competition, the ability to optimize location in terms of market share provides a strategic advantage for decision makers.

See also: **Resource allocation for epidemic control; MINLP: Application in facility location-allocation; Combinatorial optimization algorithms in resource allocation problems; Optimizing facility location with rectilinear distances; Single facility location: Multi-objective Euclidean distance location; Single facility location: Multi-objective rectilinear distance location; Single facility location: Circle covering problem; Multifacility and restricted location problems; Network location: Covering problems; Warehouse location problem; Facility location with externalities; Production-distribution system design problem; Global optimization in Weber's problem with attraction and repulsion; Facility location with staircase costs; Stochastic transportation and location problems; Facility location problems with spatial interaction; Voronoi diagrams in facility location.**

References

[1] ACHABAL, D., GORR, W.L., AND MAHAJAN, V.: 'MULTILOC: A multiple store location decision model', *J. Retailing* **58** (1982), 5–25.

[2] DREZNER, T.: 'Locating a single new facility among existing unequally attractive facilities', *J. Reg. Sci.* **34** (1994), 237–252.

[3] DREZNER, T.: 'Optimal continuous location of a retail facility, facility attractiveness, and market share: An interactive model', *J. Retailing* **70** (1994), 49–64.

[4] DREZNER, T.: 'Competitive facility location in the plane', in Z. DREZNER (ed.): *Facility Location: A Survey of Applications and Methods*, Springer, 1995.

[5] DREZNER, T.: 'Location of multiple retail facilities with a limited budget", *J. Retailing and Consumer Services* **5** (1998), 173–184.

[6] DREZNER, T., AND DREZNER, Z.: 'Competitive facilities: market share and location with random utility', *J. Reg. Sci.* **36** (1996), 1–15.

[7] DREZNER, T., AND DREZNER, Z.: 'Location of retail facilities in anticipation of future competition', *Location Sci.* **6** (1998), 155–173.

[8] DREZNER, T., DREZNER, Z., AND WESOLOWSKY, G.O.: 'On the logit approach to competitive facility location', *J. Reg. Sci.* **38** (1998), 313–327.

[9] DREZNER, Z., AND DREZNER, T.: 'Applied location models', in G. HARCOULIDES (ed.): *Modern Methods for Business Research*, Lawrence Erlbaum Ass., 1998.

[10] GHOSH, A., AND CRAIG, C.S.: 'FRANSYS: A franchise location model', *J. Retailing* **67** (1991), 212–234.

[11] GHOSH, A., AND HARCHE, F.: 'Location-allocation models in the private sector: progress, problems, and prospects', *Location Sci.* **1** (1993), 81–106.

[12] GHOSH, A., AND RUSHTON, G.: *Spatial analysis and location allocation models*, v. Nostrand Reinhold, 1987.

[13] GOODCHILD, M.F.: 'ILACS: A location allocation model for retail site selection', *J. Retailing* **60** (1984), 84–100.

[14] HODGSON, J.M.: 'A location-allocation model maximizing consumers' welfare', *Regional Studies* **15** (1981), 493–506.

[15] HOTELLING, H.: 'Stability in competition', *Economic J.* **39** (1929), 41–57.

[16] HUFF, D.L.: 'Defining and estimating a trade area', *J. Marketing* **28** (1964), 34–38.

[17] HUFF, D.L.: 'A programmed solution for approximating an optimum retail location', *Land Economics* **42** (1966), 293–303.

[18] JAIN, A.K., AND MAHAJAN, V.: *Research in marketing*, JAI Press, 1979, Chap. Evaluating the competitive environment in retailing using multiplicative competitive interactive models.

[19] NAKANISHI, M., AND COOPER, L.G.: 'Parameter estimate for multiplicative interactive choice model: least squares approach', *J. Marketing Res.* **11** (1974), 303–311.

[20] OKABE, A., BOOTS, B., AND SUGIHARA, K.: *Spatial tessellations: Concepts and applications of Voronoi diagrams*, Wiley, 1992.

[21] OKABE, A., AND SUZUKI, A.: 'Stability of spatial competition for a large number of firms on a bounded two-dimensional space', *Environm. Plan. A* **16** (1987), 107–114.

[22] REVELLE, C.: 'The maximum capture or sphere of influence problem: Hotelling revisited on a network', *J. Reg. Sci.* **26** (1986), 343–357.

[23] WILSON, A.G.: *Theory and practice in regional science*, Pion, 1976, Chap. Retailers' profits and consumers' welfare in a spatial interaction shopping model.

Tammy Drezner
California State Univ.-Fullerton
Fullerton, CA, USA
E-mail address: tdrezner@fullerton.edu

MSC2000: 90B60, 90B80, 90B85
Key words and phrases: location, competitive.

COMPETITIVE RATIO FOR PORTFOLIO MANAGEMENT, *CRPM*

Portfolio management is a typical decision making problem under incomplete, sometimes unknown, information. Very often, a *probability distribution* is assumed for stock/bond prices in the future. In the classical work of H.M. Markowitz [9], the investors are assumed to base their decisions for portfolio management on their preference of *return and risk*. In this model, the return is specified as the expected value of the portfolio, and the risk its variance. One of the great achievements of this work is its predictive power of *diversified investment decisions*.

The assumption that future events would follow some *probability distribution* is also widely accepted for many other problems for which information on future events is uncertain. Very often, *uncertainty* is used as a synonym for probability distribution. However, a fundamental problem still remains: What decisions should we make in presence of future unknown events for which we are simply ignorant of any information? In such situations, the quality of a solution made under ignorance can only be known after future events reveal themselves. Therefore, the quality of a decision should be evaluated in comparison with the optimal available strategy we could have chosen knowing the outcome. Along this approach, the concept of *competitive ratio*, which optimizes the ratio of the outcome of a strategy under *incomplete information* and the optimal outcome under complete information, has been widely applied to solve computational problems under incomplete information, [8], [11], [7], [4]. In particular, R. El-Yaniv, et al., applied competitive analysis to the problem of foreign currency purchase [5]. X. Deng has suggested to apply the competitive analysis to portfolio management problems [3].

Consider a maximization problem. Let $X = (x_1, \ldots, x_n)$ be the variables we have no complete information until in the future. Let $Y = (y_1, \ldots, y_m)$ be the decision variables for which we have to choose their values now. Let $A = (a_1, \ldots, a_k)$ be the variables we know of their values at the time we make decisions on Y. A decision rule is a function $S : A- > Y$. Let $v(A, Y, X)$ be the value of the *objective function*. Denote by $v_S(A, X) = v(A, S(A), X)$ be the value of the objective function achieved by the decision rule S if the future outcome is X. Let $\mathrm{OPT}(A, X) = \max_{\text{all } Y} v(A, Y, X)$. The competitive ratio of decision rule S is

$$\min_{\text{all } X} \frac{v_S(A, X)}{\mathrm{OPT}(A, X)}.$$

We are interested in a decision rule which achieves the optimal competitive ratio:

$$\max_{\text{all } S} \min_{\text{all } X} \frac{v_S(A, X)}{\mathrm{OPT}(A, X)}.$$

Consider the portfolio management problem of choosing from a set of n stocks. We may scale units of stocks so that one unit of money and the current price for each stock is one. The portfolio choice decision can be represented by a vector $(x_1, \ldots, x_n)$, $1 \le i \le n$, $\sum_{i=1}^n x_i = 1$.

To illustrate the competitive analysis method, we first consider the extreme case when we know no information about future prices of the stocks. A simple strategy is to distribute the fund equally to all the stocks such that $x_1 = \cdots = x_n = 1/n$. Let $1 + c_i$ be the price of stocki at the end of the period. Therefore, in retrospective, the best strategy would be to invest all the money in the stock of the best performance: $1 + c_k = \max\{1 + c_i : 1 \le i \le n\}$. The income of the above strategy achieves $\sum_{i=1}^n (1+c_i)/n$. Since we may assume that $1+c_i \ge 0$, we have

$$\frac{\sum_{i=1}^n \frac{1+c_i}{n}}{1 + c_k} \ge \frac{1}{n}.$$

This simple strategy achieves a competitive ratio of $1/n$. On the other hand, it is natural that this strategy is optimal when we have no information whatsoever about the stocks. Consider any strategy which invests x_i in stock i ($\sum_{i=1}^n x_i = 1$). Its outcome will be $\sum_{i=1}^n x_i(1+c_i)$. Since $\sum_{i=1}^n x_i = 1$, there exists some j such that $x_j \le 1/n$. In the

worst case, it may happen that we have $1+c_j = 2$ and $1 + c_i = 0$ for all other stocks. Therefore, the optimal investment will be put all the money in stock c_j. The competitive ratio of this strategy is no more than $x_j(1 + c_j)/(1 + c_j) \leq 1/n$. Therefore, the above simple strategy achieves the optimal competitive ratio when no information is available.

To illustrate this idea further, consider another case is when we have some information about future prices of the stocks. Suppose that the only information we have is that stock i will fluctuate between $[(1 - \epsilon_i), (1 + \delta_i)]$ $(-\epsilon_i \leq \delta_i)$, $1 \leq i \leq n$. It is easy to see that we may normalize the value versus the risk-free rate of interest and make it as the first option so that $-\epsilon_1 = \delta_1 = 0$. E.g., we may divide outcomes of other securities by $(1 + r)$, the *riskless interest rate*.

Given a portfolio choice decision, x_i, $1 \leq i \leq n$, $\sum_{i=1} x_i = 1$. That is, one unit of investment is distributed to n options with a fraction of x_i on option i: $1 \leq i \leq n$. Let $(1+c_i)$ be the unknown future price of option i by the projected time of sales $(-\epsilon_i \leq c_i \leq \delta_i)$. In retrospect, the optimal solution would have been $\max_{j=1}^{n}(1 + c_j)$ by investing all one unit on the option achieving the optimum. For a fixed strategy of assigning x_i: $1 \leq i \leq n$, its ratio versus the optimum will be

$$\frac{\sum_{i=1}^{n}(1 + c_i)x_i}{\max\{1 + c_i : 1 \leq i \leq n\}}.$$

Taking all situations into consideration, the competitive ratio of this strategy is

$$\min\left\{\frac{\sum_{i=1}^{n}(1 + c_i)x_i}{\max\{1 + c_j : 1 \leq j \leq n\}} : -\epsilon_i \leq c_i \leq \delta_i\right\},$$

where the minimum is taken over all the ranges of c_i: $-\epsilon_i \leq c_i \leq \delta_i$, $1 \leq i \leq n$. Suppose now that $\max_{j=1}^{n}(1 + c_j)$ is achieved at some i: $1 \leq i \leq n$. Then, the above ratio is at least x_i since $x_j \geq 0$ and $1 + c_j \geq 0$, for all j: $1 \leq j \leq n$. If $c_i < \delta_i$, the adversary can choose a new value $c_i' = \delta_i$. In this case, the denominator $\max_{j=1}^{n}(1+c_j)$ increases by $\delta_i - c_i > 0$. The numerator $\sum_{i=1}^{n}(1+c_i)x_i$ increases by $(\delta_i - c_i)x_i$. Therefore it is to the benefit of the *adversary* to choose $c_i = \delta_i$. Similarly, it is to the benefit of the adversary to set $c_j = -\epsilon_i$, for all $j \neq i$. That is, the minimum ratio is achieved at

$$\frac{(1 + \delta_i)x_i + \sum_{j\neq i}(1 - \epsilon_j)x_j}{1 + \delta_i}$$

for some i, $1 \leq i \leq n$. Therefore, the adversary will choose some i such that

$$\min\left\{\frac{(1 + \delta_i)x_i + \sum_{j\neq i}(1 - \epsilon_j)x_j}{1 + \delta_i} : 1 \leq i \leq n\right\}.$$

Given a portfolio decision vector x, we can search through all n possible situations to find the minimum in *polynomial time*. This allows us to evaluate the quality of portfolio choices in terms of their competitive ratios.

As a portfolio manager aiming at a solution with the best competitive ratio, its goal is to choose the decision vector x which maximizes

$$\min\left\{\frac{(1 + \delta_i)x_i + \sum_{j\neq i}(1 - \epsilon_j)x_j}{1 + \delta_i} : 1 \leq i \leq n\right\}.$$

In a *linear program* formulation, this is

$$\begin{cases} \max & z \\ \text{s.t.} & \dfrac{(1 + \delta_i)x_i + \sum_{j\neq i}(1 - \epsilon_j)x_j}{1 + \delta_i} \geq z, \\ & 1 \leq i \leq n, \\ & x_i \geq 0, \quad \sum_{i=1}^{n} x_i = 1. \end{cases}$$

Therefore, the optimal competitive ratio for the above portfolio management problem can be solved in polynomial time.

In the general case, information about future may be different for different investigators. Compare two situations where two investors each has two options, one government bond of riskless interest rate $1 + r$ and a security. One investor knows that the future price of the security will be in $[1+\epsilon, 1+\delta]$ with $\epsilon < \delta$ and another knows nothing about future prices of the security. The most interesting case will be $2\epsilon < r < \delta$. Apply the above analysis, the more informed investor will decide a proportion x of his fortune on riskless bond with

$$\frac{x(1+r) + (1-x)(1+\epsilon)}{1+r} = \frac{x(1+r) + (1-x)(1+\delta)}{1+\delta}.$$

Therefore, it invests

$$x = \frac{(1+\delta)(r-\epsilon)}{(1+\delta)(r-\epsilon) + (1+r)(\delta-r)}$$

on the riskless bond and

$$1 - x = \frac{(1+r)(\delta - r)}{(1+\delta)(r-\epsilon) + (1+r)(\delta - r)}$$

on the other security. Its competitive ratio will be

$$\frac{(1+\delta)(r-\epsilon) + (1+\epsilon)(\delta - r)}{(1+\delta)(r-\epsilon) + (1+r)(\delta - r)}.$$

Applying the analysis above, we see that the person knowing nothing will invest 1/2 for the riskless bond and 1/2 for the other security. However, the worst situations considered by the less informed investor would not occur at all. Therefore, its competitive ratio will be the minimum of

$$\frac{(1+r) + (1+\epsilon)}{2(1+r)}$$

and

$$\frac{(1+r) + (1+\delta)}{2(1+\delta)}.$$

From the above discussion, the decision of the investor knowing nothing has a worse competitive ratio than that of the more informed one.

In comparison with the general approach of using probability distribution for events of uncertainty, the above situation shows that the *competitive analysis* method allows analysis for information asymmetry of investors. It is not easy to apply the probability method here since, in principle, the real world should not have two different probability distributions. This advantage is not only for the above case when the range of future prices is known. It can also be applied to other types of information about future.

Other decision rules based on rationality other than probability argument have also been suggested for financial problems. In particular, T.M. Cover has suggested a solution, the *universal portfolio*, which requires no information (not even probability distribution) about the future prices of the stocks under consideration [1]. In contrast to competitive analysis which evaluates a strategy with all other strategies, Cover has evaluated his solution in comparison with a class of strategies called constant rebalanced portfolio, which maintains a fixed proportion of one's fortune in each of the securities. Notice that, this would require frequent adjustment the holdings of the securities as their prices change. Surprisingly, Cover has shown his solution to approximate, under mild conditions, the best constant rebalanced portfolio (chosen after the stock outcomes are known) which out-perform any *constant rebalanced portfolio*, any single stock and index fund such as Down Jones Index Average (DJIA) [1]. However, Cover's algorithm requires higher-dimensional integration to calculate his solution and the dimension grows with the number of securities under consideration. This may make it computationally difficult to apply this method. In comparison, the competitive analysis would suggest a solution which is a constant rebalanced portfolio with the same weight for all the securities.

Dembo and King have discussed a tracking model for asset allocation which minimizes an investor's regret (defined as the difference of the solution of a strategy under incomplete information and the optimal solution) distribution in the L_2 metric [2]. In general, one may express the regret of a decision maker with strategy S as a function of $f(v_S(X), \mathrm{OPT}(X))$, where X is the revealed future event, $v_S(X)$ is the value achieved under strategy S operating under ignorant of the future event X, and $\mathrm{OPT}(X)$ is the optimal value achievable knowing the complete information. One such function often used is the L_∞ metric distance of these two values in the feasible space [10]. However, since the authors use the absolute difference for the basis of evaluation of strategies, probability assumption is still necessary in this model. The competitive analysis and the solution of Cover [1] base the evaluation on the ratio of the performance of a strategy with unknown information and the performance of the best solution in the class of strategies under consideration.

R.M. Hogarth and H. Kunreuther have discussed situations when financial decisions are made under ignorance. They have designed experiments to study it by evaluating human empirical judgements. However, decision making processes of economic agents are ignored in this study[6].

Some information is still available in reality, though not necessarily in the form of a well shaped probability distribution. The competitive analysis provides an approach which does not rely on probability distribution, allows for analysis under *asymmetrical information* of agents in the market, and in principle, has no difficulty to include

available information in the analysis. The remaining difficulties in applying it successful to portfolio management are mainly modeling of available information and *efficient algorithms* for computational purpose.

See also: **Financial optimization; Robust optimization; Semi-infinite programming and applications in finance; Financial applications of multicriteria analysis; Portfolio selection and multicriteria analysis.**

References

[1] COVER, TH.M.: 'Universal portfolios', *Math. Finance* **1** (1991), 1–29.

[2] DEMBO, R.S., AND KING, A.J.: 'Tracking models and the optimal regret distribution in asset allocation', *Applied Stochastic Models and Data Analysis* **8** (1992), 151–157.

[3] DENG, X.: 'Portfolio management with optimal regret ratio': *Proc. Internat. Conf. Management Sci., Hong Kong, July*, 1996, pp. 289–295.

[4] DENG, X., AND PAPADIMITRIOU, C.H.: 'Competitive distributed decision-making', *Algorithmica* **16** (1996), 133–150.

[5] EL-YANIV, R., FIAT, A., KARP, R.M., AND TURPIN, G.: 'Competitive analysis of financial games': *Proc. 33rd Annual Symp. Foundations of Computer Sci.*, 1992, pp. 327–333.

[6] HOGARTH, R.M., AND KUNREUTHER, H.: 'Decision making under ignorance: Arguing with yourself', *J. Risk and Uncertainty* **10** (1995), 15–36.

[7] KARLIN, A.R., MANASSE, M.S., RUDOLPH, L., AND SLEATOR, D.D.: 'Competitive snoopy caching', *Algorithmica* **3** (1988), 79–119.

[8] MANASSE, M.S., MCGEOCH, L.A., AND SLEATOR, D.D.: 'Competitive algorithms for on-line problems', *J. Algorithms* **11** (1990), 208–230.

[9] MARKOWITZ, H.M.: *Portfolio selection: efficient diversification of investments*, Wiley, 1959.

[10] SAVAGE, L.J.: 'The theory of statistical decision', *J. Amer. Statist. Assoc.* **46** (1951), 55–67.

[11] SLEATOR, D.D., AND TARJAN, R.E.: 'Amortized efficiency of list update and paging rules', *Comm. ACM* **28** (1985), 202–208.

Xiaotie Deng
City Univ. Hong Kong
Kowloon, Hong Kong SAR, China
E-mail address: deng@cs.cityu.edu.hk

MSC2000: 91B28, 68Q25

Key words and phrases: portfolio management, unknown information, algorithms, competitive ratio.

COMPLEXITY CLASSES IN OPTIMIZATION

We survey a number of the basic ideas, results, and references, for the complexity classes P, NP, $CoNP$, $PSPACE$, $DEXPTIME$, $NDEXPTIME$, and $EXSPACE$, the most important complexity classes in optimization. These ideas and results include the following:

i) the time and space complexities of both deterministic and nondeterministic multitape Turing machines as formalized in [1];

ii) the complexity classes above including the known results on their intercontainments; and

iii) the concepts of *polynomial reducibility*, *F-hardness*, and *F-completeness*, for the complexity classes F above.

We present brief historical surveys of the results obtained in ii) and iii). We also briefly survey many of the results on the above complexity classes in the basic references [1], [12], [30], [25], [37], [29], [16], emphasizing results especially relevant to the area of optimization.

The following are a list of some of the basic notation and terminology used here.

DEFINITION 1 A *finite alphabet* Σ is a finite nonempty set of characters. A set of strings over some finite alphabet is said to be a *language*. □

Further, we denote 'infinitely often' by 'i.o.'.

DEFINITION 2 By an *exponential function* in n we mean a function $f(n) = 2^{c \cdot n^r}$ where $c, r > 0$ are constants independent of n. □

The languages or problems *3-SATISFIABILITY* (*3-SAT*), *3-DIMENSIONAL MATCHING*, *VERTEX COVER*, *CLIQUE*, *HAMILTON CIRCUIT*, and *PARTITION* are defined as in [12]. Thus, for example, the language 3-SAT is defined to be the set of all satisfiable CNF formulas with no more than 3 literals per clause, when suitably encoded as a language over some finite alphabet. We note that this language, its quantified variants, and its succinctly-specified variants are the languages in the literature most widely used to prove *NP-*, *PSPACE-*, *DEXPTIME-*, and *NDEXPTIME-*

hardness results (Definition 8 and [12], [29], [30], [22], [21]).

Finally, we denote the linear programming, $\{0,1\}$-integer linear programming, integer linear programming, and quadratic programming problems as defined in [12], [25], [30], [37] by LP, $\{0,1\}$-ILP, ILP, and QP, respectively.

Time and Space Complexity of Turing Machines. In the literature of computational complexity, the most common models of computational devices and the problems solvable by such devices are *Turing machines* (TMs) and *language recognition problems*, respectively [1], [9], [12], [15], [17], [29]. Here, we only consider multitape deterministic and nondeterministic Turing machines (denoted DTMs and NDTMs, respectively) and their associated language recognition problems as described in [1]. Informally, such a Turing machine M consists of the following:

1) a finite state control together with a finite nonempty set Q of possible *states of the control*;
2) finite nonempty *tape* and *input alphabets* T and I, respectively, and distinct symbols b and $\vdash$, denoting 'blank' and 'leftmost cell of tape', such that $I \subset T$ and $b, \vdash \in T - I$;
3) a finite number $k \geq 1$ of *tapes*, each of which is infinite to the right only and is divided into individual *tape cells* such that each cell can contain exactly one symbol in T at any one time;
4) k *tape heads*, one for each tape, each head capable of scanning a single cell at any one time;
5) a *start state* $q^o \in Q$ and a set of *accepting states* $F \subset Q$; and
6) a finite set μ of *moves*, each of the form

$$(q, s_1, \ldots, s_k, r, (t_1, d_1), \ldots, (t_k, d_k)) \in Q \times T^k \times Q \times (T \times \{L, R, S\})^k.$$

M is said to be *deterministic* if, for each $k+1$ tuple $(q, s_1, \ldots, s_k) \in Q \times T^k$, there is at most one move in μ whose initial $k+1$ components are $q, s_1, \ldots, s_k$, respectively. Otherwise, M is said to be *nondeterministic*. A state $q \in Q$ is said to be *final* if there is no move in μ whose first component equals q. We assume that the following two restrictions hold on F and μ:

7) Each accepting state is final.
8) There are no moves

$$(q, s_1, \ldots, s_k, r, (t_1, d_1), \ldots, (t_k, d_k))$$

in μ such that letting $1 \leq i \leq k$, $s_i = \vdash$ and $t_i \neq \vdash$, $s_i \neq \vdash$ and $t_i = \vdash$, or $s_i = \vdash$ and $d_i = L$.

Let $w \in I^*$. A *partial computation of M on w* is a finite sequence $\sigma_w = (m_1, \ldots, m_l)$ with $l \geq 0$ of moves of M such that the following hold:

1) Initially, M is in its start state q^o; the first tape of M holds the string $\vdash w$, one symbol per cell starting at its leftmost cell; the contents of the leftmost cells of each of the other tapes of M equal $\vdash$; the contents of all other cells of M equal b; and each of the tape heads of M scans the leftmost cell of its corresponding tape.
2) When initialized as in 1), M executes the move-rules $m_1, \ldots, m_l$ consecutively and in that order. The *length of the partial computation* $\sigma_w = (m_1, \ldots, m_l)$ of M on w equals l. A partial computation $\sigma_w = (m_1, \ldots, m_l)$ of M on w is said to be an *accepting computation* (respectively, a *nonaccepting computation*) of M on w if, after executing the sequence of moves σ_w, M is in an accepting state (respectively, M is in a final state that is not an accepting state).

The restrictions 7) and 8) on the sets F and μ ensure that *no* accepting or nonaccepting computation of M on w is an initial subsequence of any other partial computation of M on w and at *no* point during a partial computation of M on w does one of the tape heads of M attempt to move off the left end of its corresponding tape.

The *language accepted by a Turing machine M*, denoted by $L(M)$, is the set of all strings $w \in I^*$ such that there exists an accepting computation on M on w. The *language recognition problem* of M is the problem of verifying, given $w \in I^* \cap L(M)$, that w is, in fact an element of $L(M)$. The time and space complexities of M are defined in terms of partial computations of M on strings $w \in I^*$ as follows:

DEFINITION 3 Let M be a deterministic Turing machine. The *time complexity* of M, denoted by $T_M(\cdot)$, is the function from $\mathbf{N}$ to $\mathbf{N} \cup \{\infty\}$ defined, for all $n \in \mathbf{N}$, by

- $T_M(n) = \max\{l \in \mathbf{N} : l$ is the length of a partial computation of M on w, where $w \in I^n\}$, if this maximum exists;
- $T_M(n) = \infty$ otherwise.

The *space complexity* of M, denoted by $S_M(\cdot)$ is the function from $\mathbf{N}$ to $\mathbf{N} \cup \{\infty\}$ defined, for all $n \in \mathbf{N}$, by

- $S_M(n) = \max\{s \in \mathbf{N} : s$ is the maximum number of tape cells scanned on any of the tapes of M during a partial computation of M on $w\}$, where $w \in I^n$, if this maximum exists;
- $S_M(n) = \infty$ otherwise.

Let M be a nondeterministic Turing machine. The *time complexity* of M, denoted by $T_M(\cdot)$, is the function from $\mathbf{N}$ to $\mathbf{N}$ defined, for all $n \in \mathbf{N}$, by

- $T_M(n) = 0$ if no string $w \in I^n$ is in $L(M)$;
- $T_M(n) = \max\{l \in \mathbf{N} : l$ is the minimum length of an accepting computation M on $w\}$, where $w \in I^n \cap L(M)$;
- $T_M(n) = \infty$ otherwise.

The *space complexity* of M, denoted by $S_M(\cdot)$ is the function from $\mathbf{N}$ to $\mathbf{N}$ defined, for all $n \in \mathbf{N}$, by

- $S_M(n) = 0$ if no string $w \in I^n$ is in $L(M)$;
- $S_M(n) = \max\{m \in \mathbf{N} | m$ equals the minimum of the maximum numbers of tape cells scanned on any of the tapes of M during an accepting computation of M on w, where $w \in I^n \cap L(M)\}$;
- $S_M(n) = \infty$ otherwise.

□

There is a fundamental difference between the definitions of time and space complexity, for DTMs and for NDTMs, respectively. For a DTM M, the functions T_M and S_M are defined in terms of the numbers of moves executed and tape cells scanned in an *arbitrary* partial computation of M on strings $w \in I^*$. In contrast, for an NDTM M, these functions are defined *only* in terms of the numbers of moves executed and tape cells scanned in minimum 'cost accepting computations' of M on strings $w \in I^* \cap L(M)$. The following are two easy implications of this difference:

PROPOSITION 4 If L is accepted by a DTM M with time and space complexities T_M and S_M, then L is also accepted by an NDTM M' with time and space complexities $T_{M'}$ and $S_{M'}$ such that, for all $n \in \mathbf{N}$,

$$T_{M'}(n) \leq T_M(n) \quad \text{and} \quad S_{M'}(n) \leq S_{M'}(n).$$

□

PROPOSITION 5 If L is accepted by a DTM M with input alphabet I such that, for all $n \in \mathbf{N}$, $T_M(n) \in \mathbf{N}$ (equivalently, $T_M(n) \neq \infty$), then the language $I^* - L$ is accepted by a DTM M' such that, for all $n \in \mathbf{N}$, $T_{M'}(n) = T_M(n)$. □

Consequently, deterministic time complexity classes as defined in Definition (6) are closed under complementation. At present (1999), nondeterministic time complexity classes are not known to be closed under complementation.

Definition of the Complexity Classes. In Definition (6) we use Definition (3) to define the time and space complexity classes most relevant to optimization. Next in Theorem (7), we give several basic properties of these classes, whose proofs do not require the concept of polynomial time reducibility defined in Definition (8).

DEFINITION 6 Let T, S: $\mathbf{N} \to \mathbf{N}$.

A Turing machine M is said to be *polynomially time-bounded*, respectively *polynomially space-bounded*, if the function $T_M(n)$, respectively $S_M(n)$, is bounded above by a polynomial function in n. M is said to be *exponentially time-bounded*, respectively *exponentially space-bounded*, if the function $T_M(n)$, respectively $S_M(n)$, is bounded above by an exponential function in n.

- $DTIME(T(n))$, respectively $NDTIME(T(n))$, is the class of all languages L for which there exist a DTM, respectively an NDTM, M and and a $c > 0$ such that

  $$L = L(M)$$

  and, for all $n \in \mathbf{N}$,

$$T_M(n) \leq c \cdot T(n).$$

- $DSPACE(S(n))$, respectively $NDSPACE(S(n))$, is the class of all languages L for which there exist a DTM, respectively an NDTM, M and a $c > 0$ such that

$$L = L(M)$$

and, for all $n \in \mathbf{N}$,

$$S_M(n) \leq c \cdot S(n).$$

- P, NP, $PSPACE$, $DEXPTIME$, $NDEXPTIME$, and $EXSPACE$ are the classes of all languages L such that L is the language accepted by a polynomially time-bounded DTM, a polynomially time-bounded NDTM, a polynomially space-bounded TM, an exponentially time-bounded DTM, an exponentially time-bounded NDTM, and an exponentially space-bounded TM, respectively.
- $CoNP$, respectively $CoNDEXPTIME$, is the class of all languages L for which there exists an NDTM M with tape alphabet I such that $L = I^* - L(M)$ and the function $T_M(n)$ is polynomially time-bounded, respectively exponentially time-bounded.

□

THEOREM 7 1) The following containments hold among the complexity classes defined in Definition 6:

a) $P \subset NP \cap CoNP$.
b) $NP, CoNP \subset PSPACE \subset DEXPTIME$.
c) $DEXPTIME \subset NDEXPTIME \cap CoNDEXPTIME$.
d) $NDEXPTIME, CoNDEXPTIME \subset EXSPACE$.

2) a) $P = NP$ if and only if $NDTIME(n) \subset P$;
b) $P = PSPACE$ if and only if $\exists \epsilon > 0$ such that $DSPACE(n^\epsilon) \subset P$;
c) $NP = PSPACE$ if and only if $\exists \epsilon \geq 0$ such that $DSPACE(n^\epsilon) \subset NP$;
d) for all integers $k \geq 1$, $NDTIME(n^k) \subset DSPACE(n^k)$.

3) $PSPACE = DEXPTIME$ if and only if $\exists \epsilon \geq 0$ such that $DTIME(2^{n^\epsilon}) \subset PSPACE$.

4) If we restrict the classes P and NP to languages over a single letter alphabet, denoting these restrictions by P_{sla} and NP_{sla}, respectively, then
a) $P_{\text{sla}} = NP_{\text{sla}}$ if and only if $\cup_{k \geq 1} DTIME(2^{kn}) = \cup_{k \geq 1} NDTIME(2^{kn})$;
b) NP_{sla} is closed under complementation if and only if the class $\cup_{k \geq 1} NDTIME(2^{kn})$ is closed under complementation.

□

PROOF. The claims in 1), 2), and 3) of the theorem follow directly from Definitions 3 and 6, the discussion after Definition 3, and simple well-known arguments involving 'padding', e.g. see [4], [13], [17]. As an example, let $L \in DSPACE(n^l)$, where $l \geq 1$ is an integer. Let $k \geq 2$ be an integer. Let $L' = \{w \cdot \#^m : m = k \cdot l,\ w \in L\}$, with $\#$ a symbol not occurring in L. Then $L' \in DSPACE(n^{1/k})$; and $L' \in P$ (respectively $L' \in NP$) if and only if $L \in P$ (respectively $L \in NP$). The claims of 4) follow from simple arguments about 'tally languages', see [13]. □

General Comments. The Turing machine model is due to A.M. Turing [36]. Additional discussions of this model can be found in [9], [17], [1].

The time and space complexity of DTMs were first studied in [15], and [14], respectively.

The time-bounded complexity classes P, NP, $CoNP$, etc. are invariant under several other formal computer models, including multihead multitape Turing machines, Turing machines with multidimensional tapes, and both the RAM and $RASP$ models of [35] and [11] under the logarithmic cost function. (For a detailed discussion of this, see [1, Chap. 1]).

It is widely assumed that a computational problem is 'practically computationally tractable' only if it can be solved by a deterministic polynomially time-bounded Turing machine [1], [12], [30], [25], [29], [37]. The resulting importance of the class P was first observed by A. Cobham [7] and J. Edmonds [10].

By the well-known *Savitch theorem* [33] that $NDSPACE(\log n) \subset DSPACE([\log n]^2)$, the classes of languages accepted by polynomially space-bounded DTMs and NDTMs are equal and the classes of languages accepted by exponentially space-bounded DTMs and NDTMs are equal.

This is the reason for defining the complexity classes *PSPACE* and *EXSPACE*, rather than the classes *DPSPACE*, *NDPSPACE*, *DEXSPACE*, and *NDEXSPACE*.

Efficient Reducibility and 'Hard' Problems. Throughout this section F is a class of languages. Following [17], [1], [12], we define polynomial reducibility, F-hardness and F-completeness under such reducibilities. To do this, we must extend the definition of DTMs given above so that the resulting machines have outputs, and thus can be viewed as computing partial functions of their inputs. This is accomplished by augmenting each DTM M with an additional 'output' tape o_M such that the tape head of o_M can only move one cell to the right or stay stationary during any move of M. In addition to all usual constraints on M, for all inputs w of M:

1) initially, during a partial computation of M on w, all cells of o_M are blank and the tape head of o_M scans its leftmost cell;

2) the value computed of M on w is the final nonblank contents of o_M during the accepting computation or the nonaccepting computation of M on w, if such a computation exists, and is undefined otherwise.

The time complexity T_M of such augmented DTMs (henceforth, referred to simply as DTMs) is defined exactly as in Definition 3.

DEFINITION 8 Let Σ and Δ be finite alphabets.

A function f from Σ^* to Δ^* is said to be *polynomial time computable* if and only if there exists a polynomially time-bounded DTM M with input alphabet Σ such that, for all $w \in \Sigma^*$, the value computed by M on input w is $f(w)$.

Let $L \subset \Sigma^*$ and $M \subset \Delta^*$. L is said to be *polynomially reducible* to M, denoted $L \leq_p M$, if and only if there is an $f\colon \Sigma^* \to \Delta^*$ such that f is polynomial time computable and, for all $w \in \Sigma^*$: $w \in L$ if and only if $f(w) \in M$.

A language L is said to be *F-hard* if and only if for all languages $L' \in F$, $L' \leq_p L$. L is said to be *F-complete* if and only if L is both F-hard and is in F. □

Henceforth, let F be any of the complexity classes *NP*, *CoNP*, *PSPACE*, *DEXPTIME*, *NDEXPTIME*, *CoNDEXPTIME*, or *EXSPACE*. The following two propositions underlie most of the work on F-hard and F-complete problems in the literature on algorithmic analysis and computational complexity.

PROPOSITION 9 1) Let Σ, Δ, and Π be finite alphabets. Let $L \subset \Sigma^*$, $M \subset \Delta^*$, and $N \subset \Pi^*$. If $L \leq_p M$ and $M \leq_p N$, then $L \leq_p N$. (Thus, polynomial reducibility is transitive.)

2) If L and M are languages such that $L \leq_p M$ and L is F-hard, then M is also F-hard.

3) $P = NP$ if and only if some *NP*-complete language is in P.

4) $P = PSPACE$ if and only if some *PSPACE*-complete language is in P.

5) $NP = PSPACE$ if and only if some *PSPACE*-complete language is in *NP*.

6) $NP = CoNP$ if and only if some *CoNP*-complete language is in *NP* if and only if some *NP*-complete language is in *CoNP*.

7) If L is *DEXPTIME*-, *NDEXPTIME*-, or *EXSPACE*-hard, then the recognition of L requires more than 2^{n^ϵ} time, 2^{n^ϵ} time, respectively 2^{n^ϵ} space, i.o. on any DTM, NDTM, respectively TM, where $\epsilon > 0$ is a constant independent of n.

□

PROOF. We sketch a proof: 1) follows from the fact that the polynomial time computable functions are closed under composition. The proofs of 2) through 6) follow directly from the correctness of 1) and Definitions 6 and 8. The proof of 7) follows directly from well-known 'hierarchy' theorems, for deterministic and nondeterministic time and space-bounded Turing machines [15], [14], [34], [1], [17], [12]. □

PROPOSITION 10 There exists an F-complete language, for each of the complexity classes F. □

PROOF. It is easy to construct F-complete languages defined in terms of Turing machines and coded versions of their inputs, for each of the complexity classes F. For example, the language $L = \{\# \cdot M_i \cdot \# \cdot \text{code}(x_1, \ldots, x_n) \cdot \#^m\colon\ x_1 \cdots x_n$

is accepted by the one-tape NDTM M_i in time $t\}$, where $m = 3 \cdot |M_i| \cdot t$, is both an element of $NDTIME(n)$ and is NP-hard. □

The first complete problems for NP and, consequently, the importance of the class NP in nonnumerical computation are due to S.A. Cook [8], and R.M. Karp [18]. Following the terminology of [12], these initial NP-complete problems include the problems

- 3-SAT,
- 3-DIMENSIONAL MATCHING,
- VERTEX COVER, CLIQUE,
- HAMILTONIAN CIRCUIT, and
- PARTITION.

The first complete problems for $PSPACE$ and $NDEXPTIME$, are due to A.R. Meyer and L.J. Stockmeyer [24]. Subsequently, a very large number of natural computational problems have been shown to be F-hard or F-complete, for each of the complexity classes F. Many examples and historical references can be found in [1], [12], [30], [25], [37], [29], [16]. References [12], [30] [25], [37], [28], are especially relevant to problems in the area of optimization, including:

- LP (which is solvable deterministically in polynomial time as show initially in [19]);
- $\{0,1\}$-ILP and ILP (the feasibility problems of which are NP-complete [17], [5], [12], [30]); and
- QP (which is NP-complete) [32], [12], [37].

Reference [12] also discusses much of the early work (prior to 1979) on the complexity of approximating NP-hard optimization problems. Reference [16] consists of several separately authored chapters surveying many of the more recent results on the complexity of approximating NP-hard optimization problems. These results include the important result of [3] that unless $P = NP$, no MAX SNP-hard optimization problem [31] has a PTAS (i.e. a polynomial time approximation scheme, [12]).

Many basic polynomial time solvable and NP-hard optimization problems become $PSPACE$-hard, $DEXPTIME$-hard, $NDEXPTIME$-hard, and even $EXSPACE$-hard, when problem instances are specified succinctly by hierarchical specifications or by 1- and 2-dimensional periodic specifications, see [20], [27], [26], [22], [21], [29]. References [2], [6] discuss algorithms for and the computational complexity of solving systems of multivariable polynomial equations over real closed fields. Most of these last problems are NP-hard or worse. Finally, the problems of determining the solvability of a system of multivariable polynomial equations over **N** or **Z** are recursively undecidable, by a straightforward effective reduction from *Hilbert's tenth problem* [23], [9].

See also: **Complexity theory: Quadratic programming; Complexity of degeneracy; Complexity theory; Information-based complexity and information-based optimization; Fractional combinatorial optimization; Complexity of gradients, Jacobians, and Hessians; Computational complexity theory; Mixed integer nonlinear programming; Parallel computing: Complexity classes; Kolmogorov complexity; NP-complete problems and proof methodology.**

References

[1] AHO, A.V., HOPCROFT, J.E., AND ULLMAN, J.D.: *The design and analysis of computer algorithms*, Addison-Wesley, 1974.

[2] ARNON, D.S., AND BUCHBERGER, B.: *Algorithms in real algebraic geometry*, Acad. Press, 1988.

[3] ARORA, S., LUND, C., MOTWANI, R., SUDAN, M., AND SZEGEDY, M.: 'Proof verification and hardness of approximation problems': *Proc. 33rd IEEE Symp. Foundations of Computer Sci.*, 1992, pp. 14–23.

[4] BOOK, R.V.: 'Comparing complexity classes', *JCCS* **9** (1974), 213–229.

[5] BOROSH, I., AND TREYBIG, L.B.: 'Bounds on positive integral solutions of linear Diophantine equations': *Proc. Amer. Math. Soc.*, Vol. 55, 1976, pp. 294–304.

[6] CANNY, J.: 'Some algebraic and geometric computations in PSPACE': *Proc. 20th Annual ACM Symposium on Theory of Computing*, 1988, pp. 460–467.

[7] COBHAM, A.: 'The intrinsic computational difficulty of functions', in Y. BAR-HILLEL (ed.): *Proc. 1964 Internat. Congress for Logic, Methodology and Philosophy of Sci.*, North-Holland, 1964, pp. 24–30.

[8] COOK, S.A.: 'The complexity of theorem-proving procedures': *Proc. Third ACM Symp. Theory of Computing*, 1971, pp. 151–158.

[9] DAVIS, M.: *Computability and unsolvability*, McGraw-Hill, 1958.

[10] EDMONDS, J.: 'Paths, trees, and flowers', *Canad. J. Math.* **17** (1965), 449–467.

[11] ELGOT, C.C., AND ROBINSON, A.: 'Random access stored program machines', *J. ACM* **11** (1964), 365–399.

[12] GAREY, M.R., AND JOHNSON, D.S.: *Computers and intractability. A guide to the theory of NP-completeness*, Freeman, 1979.

[13] HARTMANIS, J., AND HUNT III, H.B.: 'The LBA problem and its importance in the theory of computing', *Complexity of Computation* (1974), 1–26.

[14] HARTMANIS, J., LEWIS, P.M., AND STEARNS, R.E.: 'Classification of computations by time and memory requirements': *Proc. IFIP Congress*, 1965, pp. 31–35.

[15] HARTMANIS, J., AND STEARNS, R.E.: 'On the computational complexity of algorithms', *Trans. Amer. Math. Soc.* **117** (1965), 285–306.

[16] HOCHBAUM, D.S.: *Approximation algorithm for NP-hard problems*, PWS, 1979.

[17] HOPCROFT, J.E., AND ULLMAN, J.D.: *Formal languages and their relation to automata*, Addison-Wesley, 1969.

[18] KARP, R.M.: 'Reducibility among combinatorial problems', in R.E. MILLER AND J.W. THATCHER (eds.): *Complexity of Computer Computations*, Plenum, 1972, pp. 85–103.

[19] KHACHIAN, L.G.: 'A polynomial algorithm for linear IV programming', *Soviet Math. Dokl.* **20**, 191–194. (*Dokl. Akad. Nauk USSR* **224**, 1093–1096.)

[20] LENGAUER, T., AND WAGNER, K.W.: 'The correlation between the complexities of non-hierarchical and hierarchical versions of graph problems', *JCSS* **44** (1992), 63–93.

[21] MARATHE, M.V., HUNT III, H.B., ROSENKRANTZ, D.J., AND STEARNS, R.E.: 'Theory of periodically specified problems: complexity and approximability': *Proc. 13th IEEE Conf. Computational Complexity*, 1998.

[22] MARATHE, M.V., HUNT III, H.B., STEARNS, R.E., AND RADHAKRISHNAN, V.: 'Complexity of hierarchically and 1-dimensional periodically specified problems I: hardness results', in D.-Z. DU, J. GU, AND P.M. PARDALOS (eds.): *Satisfiability Problem Theory and Appl.*, DIMACS 35, Amer. Math. Soc., 1997, pp. 225–259.

[23] MATIJASEVIC, Y.V.: 'Enumerable sets are Diophantine', *Soviet Math. Dokl.* **11** (1970), 354–357. (*Dokl. Akad. Nauk USSR* **191** (1970), 279–282.)

[24] MEYER, A.R., AND STOCKMEYER, L.J.: 'The equivalence problem for regular expressions with squaring requires exponential times': *Proc. 13th Annual Symposium on Switching and Automata Theory*, 1972, pp. 125–129.

[25] NEMHAUSER, G.L., AND WOLSEY, L.A.: *Integer and combinatorial optimization*, Interscience Ser. Discrete Math. and Optim. Wiley, 1988.

[26] ORLIN, J.B.: 'Some problems on dynamic/periodic graphs', *Program. Combin. Optim.* (1984), 273–293.

[27] ORLIN, J.B.: *The complexity of dynamic/periodic languages and optimization problems*, Vol. 1676-86 of *Sloan W.P.*, MIT, 1985, A preliminary version of this paper appears in Proc. 13th annual ACM Symposium on Theory of Computing,1978, pp 218-227.

[28] PADALOS, P.M.: *Complexity in numerical optimization*, World Sci., 1993.

[29] PAPADIMITRIOU, C.H.: *Computational complexity*, Addison-Wesley, 1994.

[30] PAPADIMITRIOU, C.H., AND STEIGLITZ, K.: *Combinatorial optimization: Algorithms and complexity*, Prentice-Hall, 1982.

[31] PAPADIMITRIOU, C.H., AND YANNAKAKIS, M.: 'Optimization, approximation, and complexity classes', *JCCS* **43** (1991), 425–440.

[32] SAHNI, S.: 'Computationally related problems', *SIAM J. Comput.* **3** (1974), 262–279.

[33] SAVITCH, W.J.: 'Relationship between nondeterministic and deterministic tape complexities', *JCSS* **4**, 177–192.

[34] SEIFERAS, J.J., FISCHER, M.J., AND MEYER, A.R.: 'Refinements of nondeterministic time and space hierarchies': *Proc. 14th Annual IEEE Symp. Switching and Automata Theory*, 1973, pp. 130–137.

[35] SHEPERDSON, J.C., AND STURGIS, H.E.: 'Computability of recursive functions', *JACM* **10** (1963), 217–255.

[36] TURING, A.M.: 'On computable numbers, with an application to the Entscheidungsproblem', *Proc. London Math. Soc. (2)* **49** (1936/7), 230–265.

[37] VAVASIS, S.A.: *Nonlinear optimization: Complexity issues*, Oxford Sci. Publ., 1991.

H.B. Hunt III
Univ. Albany
New York, USA
E-mail address: hunt@cs.albany.edu

MSC2000: 90C60
Key words and phrases: complexity, complexity classes.

COMPLEXITY OF DEGENERACY

Degeneracy in Linear Programming. In mathematical programming, the terms *degeneracy*, and its absence, *nondegeneracy*, have arisen first in the simplex method of linear programming (LP), where they have been given precise definitions. The notions were first introduced by G.B. Dantzig in his seminal paper [7] when he invoked the nondegeneracy assumption to prove the finite convergence of the simplex method.

In the study of the simplex method for LP, degeneracy and nondegeneracy are properties de-

fined for basic feasible (or *extreme point*) solutions of systems of linear constraints or for the systems themselves. To give the definition, let us consider the general system of linear constraints

$$A_{i.}x \begin{cases} = b_i, & i = 1, \ldots, r, \\ \geq b_i, & i = r+1, \ldots, m, \end{cases} \tag{1}$$

where $A_{i.} \in \mathbf{R}^n$ is the row vector of coefficients in the ith constraint, and we assume that the equality constraints in it are linearly independent. Let K denote its set of feasible solutions.

Given $\bar{x} \in K$, the ith constraint in (1) is said to be: *active* or *tight* at $\bar{x}$ if either it is an equality constraint (i.e., $i \in \{1, \ldots, r\}$), or it is an inequality constraint that holds as an equation at $\bar{x}$; *inactive* or *slack* at $\bar{x}$ otherwise. We denote the index set of active constraints at $\bar{x}$, i.e., $\{i\colon i \in \{1, \ldots, m\}, A_{i.}\bar{x} = b_i\}$ by $I(\bar{x})$.

The feasible solution $\bar{x}$ for (1) is said to be an *extreme feasible solution* or a *BFS* (*basic feasible solution*) if it is the unique solution of the system of equations defined by the active constraints at it in (1); i.e., $A_{i.}\bar{x} = b_i$ for $i \in I(\bar{x})$.

The BFS $\bar{x}$ for (1) is said to be: a *nondegenerate BFS* if the set of active constraints at it, treated as equations, forms a square nonsingular system of equations; a *degenerate BFS* if that system has one or more redundant equations, i.e., if $|I(\bar{x})| > n$. Thus at a degenerate BFS, this system of equations formed from the active constraints is an overdetermined system of linear equations with a unique solution.

The general system (1) is said to be: a *degenerate system* if it has at least one degenerate BFS; *nondegenerate system* if all its BFSs are nondegenerate.

Degeneracy in Standard Form Systems. Before solving an LP, the simplex method transforms the constraints into a *standard form* which is

$$\begin{aligned} Ax &= b, \\ x &\geq 0, \end{aligned} \tag{2}$$

where the matrix A is of order $m \times n$, and without any loss of generality we assume that $\text{rank}(A) = m$. Let $A_{.j}$ denote the jth column vector of A for $j = 1, \ldots, n$, it is the column of x_j in (2) and we assume it is $\neq 0$ for all j. Let Γ denote the set of feasible solutions of (2).

Specializing the above definitions to the standard form, we conclude that a feasible solution $\bar{x}$ of (2) is a BFS if and only if $\{A_{.j}\colon j \text{ such that } \bar{x}_j > 0\}$ is linearly independent. The BFS $\bar{x}$ is: degenerate if $|\{j\colon \bar{x}_j > 0\}| < m$, nondegenerate if $|\{j\colon \bar{x}_j > 0\}| = m$.

So, for a nondegenerate BFS $\bar{x}$, the submatrix with column vectors $\{A_{.j}\colon j \text{ such that } \bar{x}_j > 0\}$ is a basis for the system of equations in (2). If $\bar{x}$ is a degenerate BFS, this submatrix has to be augmented with the columns of some variables having 0 values in $\bar{x}$ in order to become a basis; usually this augmentation can be carried out in many ways. Hence, while each nondegenerate BFS is associated with a unique basis, each degenerate BFS is associated with several (usually a huge number of) bases.

System (2) is said to be: degenerate if it has at least one degenerate BFS; nondegenerate otherwise. From this we see that if system (2) is degenerate, then the right-hand side constants vector b lies in a subspace that is the linear hull of a set of $m-1$ or less column vectors of the coefficient matrix A. These observations imply the following facts.

1) Keeping the coefficient matrix A fixed in (2), but letting the right-hand side constants vector b vary over $\mathbf{R}^m$, the set of all b for which (2) is degenerate is a set of Lebesgue measure zero in $\mathbf{R}^m$.

2) If (2) is degenerate, the right-hand side constants vector b in it can be perturbed ever so slightly to make the *perturbed system* nondegenerate.

   The *perturbation technique* for resolving the problem of cycling in the simplex method caused by degeneracy is based on this fact. We will discuss more on this later.

3) If (2) has at least one nondegenerate BFS, the dimension of Γ is $d = n - m$.

   Whether (2) is degenerate or not, every nondegenerate BFS of (2) is incident to exactly $d = n - m$ edges of Γ.

   However, a degenerate BFS is usually incident to more than d (could be very large)

edges of Γ, and the number of edges of Γ incident at different degenerate extreme points of Γ may be very different.

4) If (2) is nondegenerate, Γ is said to be a *regular* or *simple polyhedron* because every one of its vertices is incident to exactly d (the dimension of Γ) edges. This nice regular property may not hold for Γ if (2) is degenerate. Thus, degeneracy has the effect of making the polyhedron more complex geometrically.

The Complexity of Checking Whether a System of Constraints is Degenerate. Despite the rarity of degeneracy among all possible LP models, surprisingly, many real world LP models turn out to be degenerate. However, R. Chandrasekaran, S.N. Kabadi and K.G. Murty [4] showed that the problem of checking whether a given system of linear constraints is degenerate is *NP-complete*.

A nondegenerate BFS of (2) is said to be *nearly degenerate* if some variables have positive values which are very close to zero in it. In practice, while computing BFSs of (2), unless exact arithmetic is used, it is very hard to distinguish between degenerate and nearly degenerate BFSs because of *round-off errors* introduced in digital computation.

Problems Posed By Degeneracy for the Simplex Method of LP.

Cycling. Very soon after developing the simplex method for LP, Dantzig realized that it may not lead to an optimum solution under degeneracy but instead may cycle indefinitely among a set of nonoptimal degenerate bases. The first example of *cycling* in the simplex method was constructed by A.J. Hoffman [13].

For implementing the simplex method, the user has to select two *tie breaking rules* to be used in each pivot step, one for selecting the entering nonbasic variable, and the other for selecting the dropping basic variable, among all those that tie. For cycling to occur, these tie breaking rules are very crucial.

Any technique that makes sure that the simplex method cannot cycle under degeneracy is said to *resolve degeneracy*. Quite early in the development of LP, techniques for resolving degeneracy in theory, using a virtual perturbation involving powers of an infinitesimal indeterminate, without altering the data were developed ([5], [8], [21]; also see [16] for an extension of this technique to the bounded variable simplex method). These fix the tie breaker for the dropping variable as one based on lexicographic ordering, but leave the entering variable choice arbitrary among those eligible. Over the years several other techniques have been developed for resolving degeneracy in theory; some, e.g., *Bland's technique* [3], fix the tie breakers for both the entering and dropping variables. Bland's technique and others like it, however, lead to implementations which are very slow in practice.

Computationally, it is also important that techniques for resolving degeneracy pay attention to the possible effects of round-off errors in near degenerate solutions. See [10].

It is commonly believed that the problem of cycling is not encountered in practice; however, degeneracy related problems have been discovered to contribute substantially to the difficulty in using LP based methods in scheduling and related combinatorial and integer programming problems.

Stalling. Even after resolving the problem of cycling, yet another phenomenon called *stalling* at a degenerate BFS can occur in the simplex method. Unlike cycling which is an infinite repetition of the same sequence of degenerate bases, stalling is a finite but exponentially long sequence of consecutive degenerate pivot steps at the same objective value. Examples of stalling in network flow models have been exhibited by J. Edmonds, see [6], [17].

A technique is said to resolve both cycling and stalling under degeneracy, if it can be established that the total number of consecutive degenerate pivot steps in the simplex method, using this technique, is bounded above by a polynomial function of n and the size of the LP. Such techniques that fix the tie breakers for both the entering and the dropping variables have been developed in [6] for the special case of minimum cost pure network flow problems. Extending this work to the general LP model seems to be hard, as it can be shown that resolving both cycling and stalling in a general LP model is only possible if there exist tie breakers

for both the entering and dropping variables which guarantee to make the simplex method a polynomial time method for LP. To establish whether such tie breakers exist has been a long standing open problem in LP theory.

Degeneracy Handling in Commercial Codes. In spite of the folklore that cycling is very unlikely to occur in practice, commercial LP codes have sought to implement *anti-cycling procedures* that involve little overhead and are effective in practice.

The lexicographic technique for resolving degeneracy is not very desirable, as it needs the explicit basis inverse in every step (most commercial codes do not compute the basis inverse explicitly, they use matrix factorizations of the basis inverse for preserving sparsity and for numerical stability).

For handling degeneracy, commercial codes normally use procedures based on perturbing the bounds on the variables. If there is no progress in the objective value after some number of iterations dependent on problem size, then the bounds on the variables in the present basic vector are enlarged (i.e., if the previous lower and upper bounds on x_j are ℓ_j and u_j, they are changed to $\ell_j - \delta_j$ and $u_j + \delta_j$, where δ_j is a small positive quantity chosen appropriately), and the application of the algorithm is continued on the perturbed problem. When the perturbed problem reaches optimality, the bounds are reset to their original values to see if the resulting basis is optimal to the original problem (this happens very often). Otherwise, the resulting basis satisfies the optimality criterion but may be infeasible. Then a Phase I procedure is used to get feasibility, this works fine in almost all cases since the optimal basis for the perturbed problem is close to one for the original problem. The dual simplex algorithm can also be used for this later part. See [11] for details.

Effect of Degeneracy on the Optimum Face of an LP. From LP theory we know that if an LP has at least one optimum nondegenerate BFS, then the dual problem has a unique optimum solution. Conversely, if the dual problem has at least one optimum nondegenerate BFS, then the primal LP optimum solution is unique.

Effects of Degeneracy on Post-optimality Analysis in LP. After having found an optimum solution for an LP, an integral part of a good report generator is *marginal analysis*.

Consider the LP in standard form: minimize $z = cx$ subject to (2), and let $z^*(b)$ denote the optimum objective value in this LP as a function of the right-hand side constants vector b while all the other data remains fixed. The *marginal value* or *shadow price* vector for this LP is defined to be $(\partial z(b)/\partial b_i : i = 1, \ldots, m)$ when it exists.

If the LP has a nondegenerate optimum BFS, then the dual optimum solution is unique, it is the vector of marginal values; and for each right hand side constant b_i there is an interval of positive length containing the present value of b_i in its interior, which is its optimality range. As b_i varies in this range while all the other data remains fixed, the dual optimum solution remains unchanged and remains as the marginal value vector.

In practical applications, the right-hand side constants vector b in the LP model for a company's operations usually contains parameters such as the limits on raw material supplies, etc. When it exists, practitioners use the marginal value vector to derive many facts of great use in planning, such as identifying which raw material supplies are critical, what the break even price is for additional supply of each raw material, etc. Marginal analysis is the process of drawing such conclusions, and practitioners rely on it heavily to provide valuable planning information.

The situation changes dramatically when all the optimum BFSs of the LP are degenerate. The dual optimum solution may not be unique, and the marginal value vector as defined above may not exist. In its place we have two-sided marginal values: a *positive* (and a *negative*) *marginal value* giving the rate of change in the optimum objective value per unit increase (decrease) in b_i. M. Akgul [1] proved the existence of these two-sided marginal values using convex analysis. Simple proofs based on parametric LP are given in [16] where it is shown that the positive (negative) marginal value with respect to b_i is

$$\max\{\pi_i : \text{over the dual optimum face}\}$$
$$(\min\{\pi_i : \text{over the dual optimum face}\}).$$

Effects of Degeneracy in Interior Point Methods for LP. Unlike the simplex method which walks along edges of the polyhedron, the paths traced by interior point methods (IPMs) are contained in the strict interior of the polyhedron. There are many different classes of IPMs based on the strategy used. At first glance, degeneracy, a concept based on properties of extreme point solutions, does not seem to be as serious a problem for IPMs as it is for simplex methods. In fact proofs of polynomiality for IPMs of the projective, path following, and affine potential reduction categories hold true without any nondegeneracy assumption.

However, degeneracy affects the convergence of the primal-dual pair in the affine scaling method. Under primal nondegeneracy, this method has been shown to be globally convergent for any steplength as long as all the iterates remain in the interior of the feasible region. But this technique breaks down when the primal nondegeneracy assumption is removed, in fact L.H. Hall and R.J. Vanderbei [12] constructed a degenerate example to show that the dual sequence cannot be convergent anymore if any fixed steplength greater than 2/3 to the boundary is taken. Thus stepsize 2/3 to the boundary is the longest stepsize for the affine scaling algorithm that guarantees convergence of the primal-dual pair in the presence of degeneracy.

Although other IPMs go through the interior of the feasible region, degeneracy still has a role to play in them. But the problems here are different from the cycling and stalling problems occurring in the simplex method. Degeneracy and redundant constraints affect the central path which most IPMs aim to follow. Numerical performance of the algorithms may suffer from numerical instability and ill-conditioning if the optimum solutions are degenerate or near degenerate. Also, generating an optimum basis from the near optimum interior solution at the termination of the IPM is strongly polynomial, but the computational effort depends on the degree of degeneracy.

Effect of Degeneracy on Algorithms for Enumerating Extreme Point Solutions. Consider the problem of *enumerating all the extreme point solutions* of a system of linear constraints, say (2). Let ℓ_0 denote the unknown number of extreme point solutions.

If (2) is nondegenerate, all its extreme point solutions can be enumerated in time $O(\ell_0 mn)$, an effort which grows linearly with ℓ_0, D. Avis and K. Fukuda [2]. If (2) is degenerate and is the system of constraints for a network linear program, J.S. Provan [19] has an algorithm for enumerating all its extreme point solutions in time polynomial in ℓ_0 and the input size.

However, it remains an open question whether there is an algorithm for enumerating all extreme point solutions in time polynomial in ℓ_0 and input size, when (2) is a general degenerate system of constraints.

Murty and S.-J. Chung [18] have shown that degenerate polyhedra have proper subsets called *segments* satisfying certain facial incidence properties. For each nondegenerate polyhedron, the only segment possible is the whole polyhedron itself. The difficulty of enumerating extreme point solutions of degenerate systems efficiently is related to the problem of recognizing whether a given segment is the whole polyhedron or a proper subset of it.

Effect of Degeneracy in Extreme Point Ranking Methods. Consider the objective function $z(x) = cx$ defined over Γ, the set of feasible solutions of (2). For simplicity assume that Γ is a convex polytope, i.e., it is bounded. An algorithm for *ranking the extreme points* of Γ in increasing order of $z(x)$ has been discussed in [15]. In each step, this algorithm carries out the operation of enumerating the adjacent extreme points of a given extreme point of Γ.

If (2) is nondegenerate, every extreme point has exactly $n - m$ adjacent extreme points, and the above operation can be carried out efficiently by pivot steps. Hence, the complexity of generating k extreme points in the ranked sequence grows linearly with k, and the ranking algorithm becomes practically effective.

If (2) is degenerate, the number of adjacent extreme points of a degenerate extreme point of Γ may be very large, and the ranking algorithm becomes almost impractical.

The *assignment problem* is a well known exam-

ple of a highly degenerate problem. However all its extreme point solutions known as assignments are $0-1$ vectors, using this property an efficient special algorithm has been developed in [14] for ranking the assignments in increasing order of a linear objective function.

Degeneracy in Nonlinear Programming. In contrast to linear programming where the concept of degeneracy is defined purely using extreme point solutions; in nonlinear programming it is defined for any solution point.

Discussion of degeneracy arises in nonlinear programming, particularly in methods known as *active set methods*. These methods are popular for solving nonlinear programs in which the constraints are linear, say of the form (1); but also used when there are nonlinear constraints in the system. In these methods, when at a feasible point x^0, certain constraints indexed by an active set $\mathcal{A}$ are treated as equations, and the rest are temporarily disregarded, and a search direction y^0 is generated. The next point is taken ideally as the best feasible point on the half-line $\{x^0 + \lambda y^0 : \lambda \geq 0\}$. However, if one or more inequality constraints not from the set $\mathcal{A}$ are violated by $x^0 + \lambda y^0$ whenever $\lambda > 0$ and sufficiently small, then those constraints allow no progress in the search direction, and we have a degenerate situation. See [10] and [11] for a discussion of this degeneracy in active set methods, and its resolution.

There are also other generalizations of the notions of degeneracy and nondegeneracy, to systems of nonlinear constraints. In these generalizations, degeneracy is taken to mean any measure of departure of problem structure from some idealized norm. Simply put, nondegenerate means well-posed in some context, degenerate means absence of such nice structure. For a nonlinear program, nondegeneracy at a solution point has been defined variously as the satisfaction of: LICQ (linear independence constraint qualification of the binding constraint gradients), KKT first order necessary conditions for a local minimum, second order sufficient conditions for a local minimum, or the strict complementary slackness condition. Also connections between nondegeneracy and performance of algorithms has been studied, addressing the local effects of special kinds of nondegeneracy or its lack at a local minimizer. See [9].

See also: **Complexity theory: Quadratic programming; Complexity theory; Complexity classes in optimization; Information-based complexity and information-based optimization; Fractional combinatorial optimization; Complexity of gradients, Jacobians, and Hessians; Computational complexity theory; Mixed integer nonlinear programming; Parallel computing: Complexity classes; Kolmogorov complexity; *NP*-complete problems and proof methodology.**

References

[1] AKGUL, M.: 'A note on shadow prices in linear programming', *J. Oper. Res. Soc.* **35** (1984), 425–431.

[2] AVIS, D., AND FUKUDA, K.: 'A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra', *Discrete Comput. Geom.* **8** (1992), 295–313.

[3] BLAND, R.G.: 'New finite pivoting rules for the simplex method', *Math. Oper. Res.* **2** (1977), 103–107.

[4] CHANDRASEKARAN, R., KABADI, S.N., AND MURTY, K.G.: 'Some NP-complete problems in linear programming', *Oper. Res. Lett.* **1** (1982), 101–104.

[5] CHARNES, A.: 'Optimality and degeneracy in linear programming', *Econometrica* **20** (1952), 160–170.

[6] CUNNINGHAM, W.H.: 'Theoretical properties of the network simplex method', *Math. Oper. Res.* **4** (1979), 196–208.

[7] DANTZIG, G.B.: 'Maximization of a linear function of variables subject to linear inequalities', in T.C. KOOPMANS (ed.): *Activity Analysis of Production and Allocation*, Wiley, 1951, pp. 339–347.

[8] DANTZIG, G.B., ORDEN, A., AND WOLFE, P.: 'The generalized simplex method for minimizing a linear form under linear inequality restraints', *Pacific J. Math.* **5** (1955), 183–195.

[9] FIACCO, A.V., AND LIU, J.: 'Degeneracy in nonlinear programming and the development of results motivated by its presence', in T. GAL (ed.): *Degeneracy in optimization problems*, Vol. 46 of *Ann. Oper. Res.*, 1993, pp. 61–80.

[10] FLETCHER, R.: *Practical methods of optimization*, second ed., Wiley, 1987.

[11] GILL, P.E., MURRAY, W., SAUNDERS, M.A., AND WRIGHT, M.H.: 'A practical anti-cycling procedure for linearly constrained optimization', *Math. Program.* **45** (1989), 437–474.

[12] HALL, L.A., AND VANDERBEI, R.J.: 'Two-thirds is sharp for affine scaling', *Oper. Res. Lett.* **13** (1993), 197–201.

[13] HOFFMAN, A.J.: 'Cycling in the simplex algorithm', *NBS Report* **2974** (1953).
[14] MURTY, K.G.: 'An algorithm for ranking all the assignments in increasing order of cost', *Oper. Res.* **16** (1968), 682–687.
[15] MURTY, K.G.: 'Solving the fixed charge problem by ranking the extreme points', *Oper. Res.* **16** (1968), 268–279.
[16] MURTY, K.G.: *Linear programming*, Wiley, 1983.
[17] MURTY, K.G.: *Network programming*, Prentice-Hall, 1992.
[18] MURTY, K.G., AND CHUNG, S.-J.: 'Segments in enumerating faces', *Math. Program.* **70** (1995), 27–45.
[19] PROVAN, J.S.: 'Efficient enumeration of the vertices of polyhedra associated with network LPs', *Math. Program.* **63** (1994), 47–64.
[20] 'Degeneracy in optimization problems', *Ann. Oper. Res.* **46** (1993).
[21] WOLFE, P.: 'A technique for resolving degeneracy in linear programming', *SIAM J.* **11** (1963), 205–211.

Katta G. Murty
Dept. IOE, Univ. Michigan
Ann Arbor, Michigan 48109-2117, USA
E-mail address: katta_murty@umich.edu
Web address: www-personal.engin.umich.edu/~murty/

MSC2000: 90C60
Key words and phrases: degeneracy, nondegeneracy, near degeneracy, active constraints, tight constraints, inactive constraints, slack constraints, basic feasible solution, regular polyhedron, simple polyhedron, *NP*-complete problem, cycling, stalling, resolving degeneracy, positive marginal values, negative marginal values, extreme point enumeration, extreme point ranking, assignment ranking, segments of polyhedra, active set methods.

COMPLEXITY OF GRADIENTS, JACOBIANS, AND HESSIANS, *SA*

The evaluation or approximation of derivatives is a central part of most nonlinear optimization calculations. The gradients of objectives and active constraints enter directly into the Karush–Kuhn–Tucker conditions so that inaccuracies in their evaluation limit the achievable solution accuracy. The latter depends also crucially on the conditioning of the projected Hessian of the Lagrangian. Hence accurate values of this symmetric matrix allow the design of appropriate stopping criteria including the verification of second order conditions. Second derivatives also facilitate a rapid final rate of convergence, provided the step-defining linear systems can be solved by factorization or iteration at a reasonable cost. The same observations apply to more general optimization calculations like the solution of nonlinear complementarity problems.

Whether or not the obvious benefits of evaluating first and higher derivatives accurately justify the costs incurred, does strongly depend on the suitability of the differentiation method employed for the particular problem at hand. We may distinguish five principal options for evaluating or approximating derivatives

- *symbolic differentiation*;
- *handcoded derivatives*;
- *automatic differentiation*;
- *difference quotients*;
- *secant updating*.

'Numerical' Differentiation Methods. The last two options are widely used in practical optimization, primarily because they require no extra effort whatsoever on the part of the user. Difference quotients are often called *divided differences* or *finite differences*, though the last term invites confusion with a related method for discretizing differential equations. Other popular labels are *differencing* or *numerical differentiation*, because the results are floating point numbers rather than algebraic expressions. The latter are often presumed to be the output of more symbolic methods. Even though we shall see that the distinction is not quite that easy, there is no doubting the importance of the fundamental relation

$$F'(x)\dot{x} = \frac{d}{d\alpha}F(x+\alpha\dot{x})\Big|_{\alpha=0}$$
$$= \frac{1}{\varepsilon}[F(x+\varepsilon\dot{x}) - F(x)] + O(\varepsilon).$$

Here, the vector function $F\colon \mathbf{R}^n \to \mathbf{R}^m$ is assumed Lipschitz-continuously differentiable on some neighborhood of the base point $x \in \mathbf{R}^n$. In other words, the directional derivative of F along some vector $\dot{x} \in \mathbf{R}^n$ is the product of the Jacobian matrix $F'(x) \in \mathbf{R}^{m\times n}$ with the direction $\dot{x}$ and it can be approximated by a difference quotient. The quality of this approximation depends strongly on the choice of ε and one must expect a halving in the number of significant digits under the best of circumstances. Quasi-Newton, or secant methods may be viewed as an ingenious way

of sequentially incorporating difference quotients into a Jacobian approximation while iterating towards the solution vector of a nonlinear system of equations. The corresponding theory of superlinear convergence is quite beautiful from a mathematical point of view, though perhaps not terribly relevant in practice for large, structured problems.

It is important to note that the quality of the approximate derivative matrices generated by quasi-Newton methods influences only the rate of convergence but not so much the solution accuracy itself. The latter depends on the accurate evaluation of residual vectors, which may be composed of gradients as is the case for the KKT conditions. The importance of accurate residual values is particularly well understood in numerical linear algebra, and replacing them with approximations of uncertain reliability is generally a dicy proposition. Fortunately, it just so happens that gradients can usually be evaluated with working precision at a moderate cost relative to that of the underlying functions. This is far from true for Jacobians and Hessians, whose cost is very hard to predict (and even define) as we shall demonstrate further below on various examples.

The relative cost of evaluating one-sided difference quotients in p directions $\dot{x}$ from the same base point x is clearly $p + 1$. Theoretically one might sometimes reduce the evaluation costs by exploiting the fact that the p points $x + \varepsilon\dot{x}$ are close to x. This proximity may arise in the topological sense that the stepsize $\varepsilon\|\dot{x}\|$ is small as well as in the structural sense that $\dot{x}$ is sparse and thus leaves many components of x unchanged. In practice such savings are rarely realized and they would certainly destroy the main advantage of differencing, namely its black box quality, which does not require any insight or access to the process by which function values are generated. Of course, there is the optimistic assumption that they vary smoothly as a function of the argument x, and usually the selection of a suitable increment ε causes enough trouble for the user and possibly even quite a few extra trial evaluations.

Hence it is indeed fair to assume that one-sided or centered differences in p directions $\dot{x}$ at a common x require $1 + p$ or $1 + 2p$ separate function evaluations but little extra storage. By letting $\dot{x}$ range over all n Cartesian basis vectors one obtains an approximate Jacobian with first or second order accuracy at the cost of $1 + n$ or $1 + 2n$ function evaluations. The number of dependent variables does not matter for differencing so that the cost of a gradient, where $m = 1$, is also $1 + n$ or $1 + 2n$ times that of the underlying scalar function. To compute the Hessian or more generally a full second derivative tensor one needs $n(n + 1)/2$ function evaluations for one-sided and twice that many for the more accurate centered differences.

Since multiple function evaluations are an 'embarrassingly' parallel task the availability of several processors can be used to achieve a nearly perfect speed up for derivative approximations by differencing [7]. In the sparse case, the number of independent variables n can be replaced in the cost ratios above by a number $p \leq n$ that represents either the maximal number of nonzeros in any row of $F'(x)$ or the usually slightly larger chromatic number of the *column incidence graph*. The latter reduction can be achieved by the by now classical grouping or coloring technique originally due to Curtis–Powell–Reid [6] and further developed by Coleman–Moré [4]. An alternative way to compress the rows of the Jacobian even further at the expense of some linear equation solving is due to Newsam–Ramsdell [12] and has recently been adopted to automatic differentiation.

'Analytical' Differentiation Methods. The first three options listed at the beginning are based on the chain rule and may therefore be combined under the label *analytical differentiation*. They all would yield exact derivative values if real arithmetic could be performed in infinite precision. Moreover, even the actual sequence of operations performed to evaluate a particular partial derivative would quite likely be the same and thus yield identical results if the same floating point arithmetic was used. Only the way in which the instruction for this floating point calculation are generated and stored differ significantly between the three approaches. Also, there may be more or less recalculation of intermediates that are common to several partial derivatives, which can have drastic effects on the computational efficiency.

The result of the second option *handcoding* may

in principle be always similarly obtained by symbolic or automatic differentiation, provided the computer algebra package or the differentiation software is sufficiently smart. Hence we will discuss only the pure options one and three, which might of course also be combined by a highly sophisticated programmer or software tool.

Symbolic differentiation is usually performed in *computer algebra* packages like Maple, Mathematica and Reduce. Most users have the notion that the differentiation commands in these sophisticated systems turn formulas for functions into formulas for derivatives. Moreover there is a tendency to assume that having a 'formula' means directly expressing dependent variables as algebraic expressions of independents without allowing any named intermediates. The natural data structures for such formulas would be expression trees. There, every node has only one parent, so that the whole thing can be easily linearized and printed by enumeration in a depth first order. In reality computer algebra packages do not restrict themselves to expression trees, because for any nontrivial function the corresponding tree structure is very likely to represent an incredible amount of *redundancy*, even before any differentiation takes place.

Two-Stranded Chain Scenario. Consider for example a sequence of complex function evaluations

$$x_{k+1} + iy_{k+1} = \phi_k(x_k + iy_k) + i\psi_k(x_k + iy_k)$$

for $k = 0, \ldots, l-1$ starting from some initial $x_0 + iy_0 \in \mathbf{C}$. Suppose all function pairs $\phi_k + i\psi_k$ are nonlinear and do not allow any algebraic simplifications. Then eliminating the intermediates x_1 and y_1 yields the formula

$$\begin{aligned} x_2 + iy_2 = \phi_1(\phi_0(x_0, y_0) + i\varphi_0(x_0, y_0)) \\ +\psi_1(\phi_0(x_0, y_0) + i\varphi_0(x_0, y_0)), \end{aligned}$$

which involves already twice as many terms as the one-level original formula. The same doubling occurs at each subsequent level so that expressing x_l and y_l directly in terms of the initial components x_0 and y_0 yields an exponentially long formula with the symbols x_0 and y_0 each occurring exactly 2^l times. In this case one could avoid the highly undesirable expression swell by merely substituting $z_k \equiv x_k + iy_k$, which turns the binary expression tree into a simple chain of the same height l.

While this example may appear rather algebraic and somewhat contrived, exactly the same effect occurs if the real pairs (x_k, y_k) specify straight lines in the plane. Specifically, one might think of light-beams being reflected in a maze of mirrors or some other optical arrangement in the plane. Each ray (x_k, y_k) that is incoming to a mirror or lense uniquely determines an outgoing ray (x_{k+1}, y_{k+1}) via some simple algebraic relationship. Then expressing the final ray parameters (x_l, y_l) directly as functions of the initial parameters (x_0, y_0) will again yield an expression of size 2^l.

Rather than dealing with this algebraic monster one should of course keep all the intermediate pairs (x_k, y_k) with $0 \le k \le l$ as named variables. Along this chain one can easily propagate all information of interest, including the 2×2 Jacobian of (x_k, y_k) with respect to (x_0, y_0), at a temporal and spatial complexity of order l. To achieve this result one may employ suitable variants of computer algebra, automatic differentiation or, of course, handcoding. Before discussing them in more detail let us discuss a general model of function and derivative evaluations.

Computational Model. All analytical differentiation methods are based on the observation that most vector functions F of practical interest are being evaluated by a sequence of assignments

$$v_i = \varphi_i(v_j)_{j<i} \quad \text{for } i = 1, \ldots, l+m. \qquad (1)$$

Here, the variables v_i are real scalars and the elemental functions φ_i are either binary arithmetic operations or univariate intrinsics. Consequently, only one or two of the partial derivatives

$$c_{ij} \equiv \frac{\partial}{\partial v_j} \varphi_i \, (v_k)_{k<i}$$

do not vanish identically and can be evaluated at a cost comparable to that of the underlying φ_i itself.

Without loss of generality we may require that the first n variables $v_{j-n} = x_j$ with $j = 1, \ldots, n$ represent the *independent variables* and the last m variables $y_i = v_{l+i}$ with $i = 1, \ldots, m$ represent the *dependent variables*. Then the function

$y = F(x)$ is defined by the program (1). Here, the nonnegative integer l represents the number of intermediate variables, which we expect to be much larger than both n and m for seriously nonlinear problems. We will also assume that within a small constant all elemental functions have the same complexity so that we have the approximate operations count

$$\mathrm{OPS}(x \overset{\mathrm{prog}}{\longmapsto} y) \sim l \equiv \#\text{intermediates}.$$

Throughout this article, $\sim$ means proportional with small constants that are independent of the particular problem at hand. Each intermediate variable may be viewed and thus later differentiated as a function $v_i \equiv v_i(x)$ of the independent variable vector x. As long as all intermediates v_i are stored in separate locations the memory requirement for evaluating F will also be proportional to l. This is a very unrealistic assumption as most evaluation programs involve shared allocation of intermediates. Due to space constraints we will not be able to discuss any aspects of spatial complexity in this article. For a detailed treatment of various trade-offs between space and time see [9].

The way in which the elemental partials c_{ij} are handled differs amongst various analytical differentiation methods. They are always evaluated as floating point numbers at the current argument in what is variously known as *automatic* or *algorithmic* or *computational differentiation*. The same can be assumed for hand written derivative codes unless they are programmed within a computer algebra system, where the c_{ij} can be defined and manipulated as algebraic expressions. In some cases applying the chain rule to these expressions may theoretically lead to significant simplifications and thus potentially provide the user with analytical insight. In the following section we reverse engineer one such class of examples and arrive at the tentative conclusion that the practical potential for symbolic simplifications during the differentiation process appears to be very slim indeed.

Indefinite Integral Scenario. Suppose that

$$F(x) = \int_a^x \frac{P(\widetilde{x})}{Q(\widetilde{x})}\, d\widetilde{x}$$

for two polynomials $P(x)$ and $Q(x)$ with $\deg(Q) > \deg(P)$. Besides a rational term the symbolic expression for $F(x)$ is then likely to contain a welter of logarithms and arcus tangents, whose complexity may easily exceed that of the integrand $f(x) \equiv P(x)/Q(x)$ by orders of magnitude. Then fully symbolic differentiation will of course lead back to an algebraic expression for $f(x)$, while automatic differentiation will combine the c_{ij} in floating point arithmetic according to some variant of the chain rule and obtain 'just' a numerical value of $f(x)$ at the given point $x \in \mathbf{R}$. Moreover, due to cancellations that value may well be less accurate than that obtained by plugging the particular argument x into the formula for $f(x)$.

However, similar numerical instabilities are likely to already affect the evaluation of $F(x)$ itself. They may also show up in the form of an imaginary component when the coefficients of $P(x)$ and $Q(x)$ are real but given in floating point format. Then the roots of the denominator polynomial are already perturbed by unavoidable round-off and symbolic differentiation of the resulting expression for $F(x)$ will usually not lead back to $f(x)$ but some other rational function with a higher polynomial degree in the numerator or denominator. To avoid this effect all coefficients of $f(x)$ must be specified as algebraic numbers so that the symbolic integration can be performed exactly. This process which typically involves rational numbers with enormous coefficients and thus requires a large computational effort.

Hence on practical models one may well be better advised to evaluate $F(x)$ by a numerical quadrature yielding highly accurate results at a fraction of the computing time. Analytically differentiating a nonadaptive quadrature procedure yields the same quadrature applied to the derivatives of the integrand, namely $f'(x) = F''(x)$. Hence the resulting values are quite likely to be good approximations to the original integrand $f(x)$ and they are the exact derivatives of the approximate values computed for $F(x)$ by the quadrature.

Lack of Smoothness. Adaptive quadratures on the other hand may vary grid points and coefficient values in a nondifferentiable or even discontinuous fashion. Then derivatives of the quadrature

value may well not exist in the classical sense at some critical arguments x. This difficulty is likely to arise in the form of program branches in all substantial scientific codes and there is no agreement yet on how to deal with it. In most situations one can still compute one-sided directional derivatives as well as generalized gradients and Jacobians [9]. Naturally, computing difference quotients of nonsmooth functions is also a risky proposition. Generally, optimal results in terms of accuracy and efficiency can only be expected from a derivatives code developed by a knowledgeable user, possibly with the help of program analysis and transformation tools.

Predictability of Complexities. With regards to spatial and temporal complexity the following basic distinction applies between the analytical differentiation methods sketched above. The cost of fully symbolic differentiation seems impossible to predict. It can sometimes be very low due to fortuitous cancellations but it is more likely to grow drastically with the complexity of the underlying function. In contrast the relative cost incurred by the various modes of automatic differentiation can always be a priori bounded in terms of the number of independent and dependent variables. Moreover, as we will see below these bounds can sometimes be substantially undercut for certain structured problems.

Another advantage of automatic differentiation compared to a fully symbolic approach is that restrictions and projections of Jacobians and Hessians to certain subspaces of the functions domain and range can be built into the differentiation process with corresponding savings in computational complexity. In the remainder of this article we will therefore focus on the complexity of various automatic differentiation techniques; always making sure that no other known approach is superior in terms of accuracy and complexity on general vector functions defined by a sequence of elemental assignments.

Goal-Oriented Differentiation. The two-stranded chain scenario above illustrates the crucial importance of suitable representations of the mathematical objects, whose complexity we try to quantify here. So one really has to be more specific about what one means by *computing* a function, gradient, Jacobian, Hessian, or their restriction and projection to certain subspaces. At the very least we have to distinguish the (repeated) *evaluation* in floating point arithmetic at various arguments from the *preparation* of a suitable procedure for doing so. This preparation stage comes actually first and might be considered the symbolic part of the differentiation process. It usually involves no floating point operations, except possibly the propagation and simplification of some constants. This happens for example when a source code for evaluating F is precompiled into a source code for jointly evaluating $F(x)$ and its Jacobian $F'(x)$ at a given argument x. In the remainder we will neglect the preparation effort presuming that it can be amortized over many numerical evaluations as is typically the case in iterative or time-dependent computations.

In general, it is not a priori understood that $F'(x)$ should be returned as a rectangular array of floating point numbers, especially if it is sparse or otherwise structured. Its cheapest representation is the sparse triangular matrix

$$C = C(x) \equiv (c_{ij})_{i=1-n,\ldots,l+m}^{j=1-n,\ldots,l+m}.$$

The nonzero entries in C can be obtained during the evaluation of F at a given x for little extra cost in terms of arithmetic operations so that

$$\mathrm{OPS}\{x \mapsto C\} \sim \mathrm{OPS}\{x \overset{\mathrm{prog}}{\longmapsto} F\}.$$

As we will see below, the nonzeros in C allow directly the calculation of the products

$$F'(x)\dot{x} \in \mathbf{R}^m \quad \text{for } \dot{x} \in \mathbf{R}^n,$$

and

$$F'(x)^\top \bar{y}^\top \in \mathbf{R}^n \quad \text{for } \bar{y}^\top \in \mathbf{R}^m,$$

using just one multiplication and addition per $c_{ij} \not\equiv 0$. So if our goal is the iterative calculation of an approximate Newton-step using just a few matrix-vector products, we are well advised to just work with the collection of nonzero entries of C provided it can be kept in memory. If on the other hand we expect to take a large number of iterations or wish to compute a matrix factorization of the Jacobian we have to first accumulate all mn partial derivatives $\partial y_{\bar{i}}/\partial x_{\bar{j}}$ from the elemental partials c_{ij}.

It is well understood that a subsequent inplace triangular factorization of the Jacobian $F'(x)$ yields an ideal representation if one needs to multiply itself as well as its inverse by several vectors and matrices from the left or right. Hence we have at least three possible ways in which a Jacobian can be represented and kept in storage:

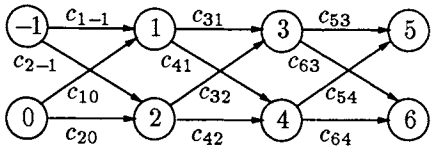
- unaccumulated: computational graph;
- accumulated: rectangular array;
- factorized: two triangular arrays.

Here the arrays may be replaced by sparse matrix structures. For the time being we note that Jacobians and Hessians can be provided in various representation at various costs for various purposes. Which one is most appropriate depends strongly on the structure of the problem function $F(x)$ at hand and the final numerical purpose of evaluating derivatives in the first place. The interpretation of C as computational graph goes back to L.V. Kantorovich and requires a little more explanation.

The Computational Graph. With respect to the precedence relation

$$j \prec i \iff c_{ij} \not\equiv 0 \iff (j,i) \in E,$$

the indices $i, j \in V \equiv [1-n, \ldots, l+m]$ form a directed graph with the edge set E. Since by assumption $j \prec i$ implies $j < i$ the graph is acyclic and the transitive closure of $\prec$ defines a partial ordering between the corresponding variables v_i and v_j. The minimal and maximal elements with respect to that order are exactly the independent and dependent variables $v_{j-n} \equiv x_j$ with $j = 1, \ldots, n$ and the $v_{m+i} \equiv y_i$ with $i = 1, \ldots, m$, respectively. For the two stranded chain scenario with $l = 3$ one obtains a computational graph of the following form:



Assuming that all elemental φ_i are unary functions or binary operations we find $|E| \leq 2(l+m) \sim l$. One may always annotate the graph vertices with the elemental functions φ_i and the edges with the nonvanishing elemental partials c_{ij}. For most purposes the φ_i do not really matter and we may represent the graph (V, E) simply by the sparse matrix C.

Forward Mode. Given some vector $\dot{x} \equiv (\dot{v}_{j-n})_{j=1,\ldots,n} \in \mathbf{R}^n$, there exist derivatives

$$\dot{v}_i \equiv \frac{d}{d\alpha} v_i(x + \alpha \dot{x})\Big|_{\alpha=0} \quad \text{for } 1 \leq i \leq l+m.$$

By the chain rule these $\dot{v}_i$ satisfy the recurrence

$$\dot{v}_i \equiv \sum_{j \prec i} c_{ij} \dot{v}_j \quad \text{for } i = 1, \ldots, l+m. \tag{2}$$

The resulting tangent vector $\dot{y} \equiv (\dot{v}_{l+i})_{i=1,\ldots,m}$ satisfies $\dot{y} = F'(x)\dot{x}$ and it is obtained at a cost proportional to l. Instead of propagating derivatives with respect to just one direction vector $\dot{x}$ one may amortize certain overheads by bundling p of them into a matrix $\dot{X} \in \mathbf{R}^{n \times p}$ and then computing simultaneously $\dot{Y} = F'(x)\dot{X} \in \mathbf{R}^{m \times p}$. The cost of this *vector forward* mode of automatic differentiation is given by

$$\mathrm{OPS}\{C \overset{\text{forw}}{\longmapsto} \dot{Y}\} \sim pl \sim p\,\mathrm{OPS}\{x \overset{\text{prog}}{\longmapsto} y\}. \tag{3}$$

If the columns of $\dot{X}$ are Cartesian basis vectors $e_j \in \mathbf{R}^n$ the corresponding columns of the resulting $\dot{Y}$ are the jth columns of the Jacobian. Hence by setting $\dot{X} = I$ with $p = n$ we may compute the whole Jacobian at a temporal complexity proportional to nl. Fortunately, in many applications the whole Jacobian is either not needed at all or due to its sparsity pattern it may be reconstructed from its *compression* $\dot{Y} = F'(x)\dot{X}$ for a suitable *seed matrix* $\dot{X}$. As in the case of difference quotients this matrix may be chosen according to the Curtis–Powell–Reid [6] or the Newsam–Ramsdell [12] approach with p usually close to the maximal number of nonzeros in any row of the Jacobian.

Bauer's Formula. Using the recurrence for the $\dot{v}_i$ given above one may also obtain an explicit expression for each individual partial derivative $\partial y_i / \partial x_j$. Namely, it is given by the sum over the products of all arc values $c_{\widetilde{ij}}$ along all paths connecting the minimal node v_{j-n} with the maximal node v_{l+i}. This formula due to F.L. Bauer [1] implies in particular that the ijth Jacobian entry vanishes identically exactly when there is no path connecting nodes $j-n$ and $l+i$ in the computational graph. In

general the number of distinct paths in the graph is very large and it represents exactly the lengths of the formulas obtained if one expresses each y_i directly in terms of all x_j that it depends on. Hence we may conclude

$$\mathrm{OPS}\{C \overset{\text{bauer}}{\longmapsto} F'\} \sim \mathrm{OPS}\{x \overset{\text{formul}}{\longmapsto} y\}.$$

In the two-stranded chain scenario considered above, both operations counts would be of order 2^l, which is obviously an unacceptable effort. Fortunately, vector forward and Bauer's formula are just two special choices amongst many ways for accumulating the Jacobian $F'(x)$ from the computational graph C. The most celebrated alternative is the reverse or backward mode of automatic differentiation.

Reverse Mode. Rather than propagating directional derivatives $\dot{v}_i$ forward through the computational graph one may also propagate adjoint quantities $\overline{v}_i$ backward. To define them properly one must perturb the original evaluation loop by rounding errors δ_i so that now

$$v_i = \delta_i + \varphi_i(v_j)_{j<i} \quad \text{for } i = 1-n, \ldots, l.$$

Then the resulting vector y is a function not only of x but also of the vector of small perturbations $(\delta_i)_{i=1-n,\ldots,l}$. Given any row vector of weights $\overline{y} = (\overline{v}_{l+i})_{i=1,\ldots,m}$ we obtain the sensitivities

$$\overline{v}_i \equiv \frac{\partial}{\partial \delta_i} \overline{y} y \Big|_{\delta_i = 0} \quad \text{for } 1-n \le i \le l,$$

where all other perturbations δ_j with $j \neq i$ are set to zero during the differentiation. The adjoint components $\overline{v}_{j-n} = \overline{x}_j$ form the row vector $\overline{x} = \overline{y} F'(x) \in \mathbf{R}^n$, which is simply the gradient of the linear combination $\overline{y}F(x)$. In the optimization context this scalar valued function is usually a Lagrangian, whose gradient and Hessian figure prominently in the first and second order optimality conditions. The amazing thing is that as a consequence of the chain rule such gradients can be computed at the same cost as tangents by using the backward recurrence

$$\overline{v}_j = \sum_{i \succ j} \overline{v}_i c_{ij} \quad \text{for } j = l, \ldots, 1-n. \qquad (4)$$

Just like in the forward scalar recurrence (2), each elemental partial $c_{ij} \neq 0$ occurs exactly once and we may amortize costs by bundling several $\overline{y}$ into an adjoint seed matrix $\overline{Y} \in \mathbf{R}^{q \times m}$. This vector reverse mode yields the matrix $\overline{X} = \overline{Y} F'(x) \in \mathbf{R}^{q \times n}$ at the cost

$$\mathrm{OPS}\{C \overset{\text{rev}}{\longmapsto} \overline{X}\} \sim ql \sim q\,\mathrm{OPS}\{x \overset{\text{prog}}{\longmapsto} y\}.$$

Again the whole Jacobian is obtained directly if we seed $\overline{Y} = I$ with $q = m$. Hence we find by comparison with (3) as a rule of thumb that the reverse mode is preferable if $m \ll n$, i.e., if there are not nearly as many dependents as independents. In classical NLPs we may think of m as the number of active constraints plus one, which is often much smaller then n, the number of variables. In unconstrained optimization we have $m = 1$ so that the gradient of the objective F can be computed with essentially the same effort as F itself. In the sparse case we may now employ column rather than row compression with q roughly equal to the maximal number of nonzeros in any column of $F'(x)$.

For suitable seeds $\overline{Y}$ the column compression $\overline{X} = \overline{Y} F'(x)$ allows the reconstruction of the complete Jacobian $F'(x)$. Furthermore, row and column compression can be combined yielding for example Jacobians with arrow head structure at the cost of roughly $p + q = 3$ function evaluations. In that case one may use

$$\dot{X} \equiv \begin{pmatrix} 1 & 1 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}^\top \quad \text{and} \quad \overline{Y} = e_m^\top.$$

Then $\overline{X} = \overline{Y} F'(x)$ is the last row of the arrowhead matrix $F'(x)$ and the two columns of $\dot{Y} = F'(x)\dot{X}$ contain all other nonzero entries. For pure row or column compression dense rows or columns always force $p = n$ or $q = m$, respectively. Hence the combination of forward and reverse differentiation offers the potential for great savings. In either case projections and restrictions of the Jacobian to subspaces of the vector functions domain and range can be built into the differentiation process, which is part of the goal-orientation we alluded to before.

Second Order Adjoints. Rather than separately propagating some first derivatives forward, others reverse, and then combining the results to compute Jacobian matrices efficiently, one may compose these two fundamental modes to compute second derivatives like Hessians of Lagrangians. More

specifically, we obtain by directional differentiation of the adjoint relation $\overline{x} = \overline{y}F'(x)$ the second order adjoint

$$\dot{\overline{x}} = \overline{y}F''(x)\dot{x} \in \mathbf{R}^n.$$

Here we have assumed that the adjoint vector $\overline{y}$ is constant. We also have taken liberties with matrix vector notation by suggesting that the $m \times n \times n$ derivative tensor $F''(x)$ can be multiplied by the row vector $\overline{y} \in \mathbf{R}^m$ from the left and the column vector $\dot{x} \in \mathbf{R}^n$ from the right yielding a row vector $\dot{\overline{x}}$ of dimension n. In an optimization context $\overline{y}$ should be thought of as a vector of Lagrange multipliers and $\dot{x}$ as a feasible direction. By composing the complexity bounds for the reverse and the forward mode one obtains the estimates

$$\mathrm{OPS}\{x \overset{\mathrm{prog}}{\longmapsto} y\} \sim \mathrm{OPS}\{x, \dot{x} \overset{\mathrm{forw}}{\longmapsto} \dot{y}\}$$

$$\sim \mathrm{OPS}\{x, \overline{y} \overset{\mathrm{rev}}{\longmapsto} \overline{x}\} \sim \mathrm{OPS}\{x, \dot{x}, \overline{y} \overset{\mathrm{ad}}{\longmapsto} \dot{\overline{x}}\}.$$

Here, ad represents reverse differentiation followed by forward differentiation or vise versa. The former interpretation is a little easier to implement and involves only one forward and one backward sweep through the computational graph.

Operations Counts and Overheads. From a practical point of view one would of course like to know the proportionality factors in the relations above. If one counts just multiplication operations then $\dot{y}$ and $\overline{x}$ are at worst 3 times as expensive as y, and $\dot{\overline{x}}$ is at most 9 times as expensive. A nice intuitive example is the calculation of the determinant y of a $\sqrt{n} \times \sqrt{n}$ matrix whose entries form the variable vector x. Then we have $m = 1$ and

$$\mathrm{OPS}\{x \mapsto y\} = \frac{1}{3}\sqrt{n}^3 + O(n)$$

multiplications if one uses an LU factorization. Then it can be seen that $\overline{y} = 1/y$ makes $\overline{x}$ the transpose of the inverse matrix and the resulting cost estimate of $\sqrt{n}^3 + O(n)$ multiplications conforms exactly with that for the usual substitution procedure.

However, these operations count ratios are no reliable indications of actual runtimes, which depend very strongly on the computing platform, the particular problem an hand, and the characteristics of the AD tool. Implementations of the vector forward mode like ADIFOR [3] that generate compilable source codes can easily compete with divided differences, i.e. compute p directional derivatives in the form $\dot{Y} = F'(x)\dot{X}$ at the cost of about p function evaluations. For sizeable $p \approx 10$ they are usually faster than divided differences, unless the roughly p-fold increase in storage results in too much paging onto disk. The reverse mode is an entirely different ball-game since most intermediate values v_i and some control flow hints need to be first saved and later retrieved, which can easily make the calculation of adjoints memory bound. This memory access overhead can be partially amortized in the vector reverse mode, which yields a bundle $\overline{X} = \overline{Y}F'(x)$ of q gradient vectors. For example in multicriteria optimization one may well have $q \approx 10$ objectives or soft constraints, whose gradients are needed simultaneously.

Worst-Case Optimality. Counting only multiplications we obtain for Jacobians $F' \in \mathbf{R}^{m \times n}$ the complexity bound

$$\mathrm{OPS}\{x \overset{\mathrm{ad}}{\mapsto} F'\} \leq 3 \min(n, m)\, \mathrm{OPS}\{x \overset{\mathrm{prog}}{\longmapsto} y\}.$$

Here, n and m can be reduced to the maximal number of nonzero entries in the rows and columns of the Jacobian, respectively.

Similarly, we have for the one-sided projection of the Lagrangian Hessian

$$H(x, \overline{y}) \equiv \overline{y}F'' \equiv \sum_{i=1}^{m} \overline{y}_i \nabla^2 F_i \in \mathbf{R}^{n \times n}$$

onto the space spanned by the columns of $\dot{X}$:

$$\mathrm{OPS}\{x \overset{\mathrm{ad}}{\mapsto} H(x, \overline{y})\dot{X}\} \leq 9p\, \mathrm{OPS}\{x \overset{\mathrm{prog}}{\longmapsto} y\}.$$

As we already discussed for indefinite integrals there are certainly functions whose derivatives can be evaluated much cheaper than they themselves for example using a computer algebra package. Note that here again we have neglected the preparation effort, which may be very substantial for symbolic differentiation. Nevertheless, the estimates given above for AD are optimal in the sense that there are vector functions F defined by evaluation procedures of the form (1), for which no differentiation process imaginable can produce the Jacobian and projected Hessian significantly cheaper than the given cost bound divided by a small constant. Here, producing these matrices is

understood to mean calculating all its elements explicitly, which may or may not be actually required by the overall computation.

Consider, for example, the cubic vector function

$$F(x) = x + \frac{b(a^\top x)^3}{2} \quad \text{with } a, b \in \mathbf{R}^n.$$

Its Jacobian and projected Hessian are given by

$$F'(x) = I + b\left(a^\top x\right)^2 a^\top \in \mathbf{R}^{n \times n}$$

and

$$H(x, \overline{y})\dot{X} = 2a(\overline{y}b)(a^\top x)a^\top \dot{X} \in \mathbf{R}^{n \times p}.$$

For general a, b and $\dot{X}$, all entries of the matrices $F'(x)$ and $H(x, \overline{y})\dot{X}$ are distinct and depend nontrivially on x. Hence their explicit calculation by any method requires at least n^2 or np arithmetic operations, respectively. Since the evaluation of F itself can be performed using just $3n$ multiplications and a few additions, the operations count ratios given above cannot be improved by more than a constant. There are other, more meaningful examples [9] with the same property, namely that their Jacobians and projected Hessians are orders of magnitude more expensive than the vector function itself. At least this is true if we insist on representing them as rectangular arrays of reals. This does not contradict our earlier observation that gradients are cheap, because the components of $F(x)$ cannot be considered as independent scalar functions. Rather, their simultaneous evaluation may involve many common subexpressions, as is the case for our rank-one example. These appear to be less beneficial for the corresponding derivative evaluation, thus widening the gap between function and derivative complexities.

Expensive≡Redundant? The rank-one problem and similar examples for which explicit Jacobians or Hessians appear to be expensive have a property that one might call *redundancy*. Namely, as x varies over some open neighborhood in its domain, the Jacobian $F'(x)$ stays in a lower-dimensional manifold of the linear space of all matrices with its format and sparsity pattern. In other words, the nonzero entries of the Jacobian are not truly independent of each other so that computing them all and storing them separately may be wasteful. In the rank-one example the Jacobian $F'(x)$ is dense but belongs at all x to the one-dimensional affine variety $\{I + b\alpha a^\top : \alpha \in \mathbf{R}\}$. Note that the vectors $a, b \in \mathbf{R}^n$ are assumed to be dense and constant parameter vectors of the problem at hand. Their elements all play the role of elemental partials c_{ij} with the corresponding operation φ_i being multiplications. Hence accumulating the extremely sparse triangular matrix C, which involves only $O(n)$ nonzero entries, to the dense $n \times n$ array $F'(x)$ is almost certainly a bad idea, no matter what the ultimate purpose of the calculation. In particular, if one wishes to solve linear systems in the Jacobian, the inverse formula of Sherman–Morrison–Woodbury provides a way of computing the solution of rank-one perturbations to diagonal matrices with $O(n)$ effort. This formula may be seen as a very special case of embedding linear systems in F' into a much larger and sparse linear system involving C as demonstrated in [11] and [5].

As of now, all our examples for which the array representation of Jacobians and Hessians are orders of magnitude more expensive to evaluate than the underlying vector function exhibit this redundancy property. In other words, we know of no convincing example where vectors that one may actually wish to calculate as end products are necessarily orders of magnitude more expensive than the functions themselves. Especially for large problems it seems hard to imagine that array representations of the Jacobians and Hessians themselves are really something anybody would wish to look at rather than just use as auxiliary quantities within the overall calculation.

So evaluating complete derivative arrays is a bit like fitting a handle to a wooden crate that needs to be moved about frequently. If the crate is of small weight and size this job is easily performed using a few screws. If, on the other hand, the crate is large and heavy, fitting a handle is likely to require additional bracing and other reinforcements. Moreover, this effort is completely pointless since nobody can just pick up the crate by the handle anyhow and one might as well use a fork left in the first place.

Preaccumulation and Combinatorics. The temporal complexity for both the forward and the

reverse (vector) mode are proportional to the number of edges in the linearized computational graph. Hence one may try to reduce the number of edges by certain algebraic manipulations that leave the corresponding Jacobian, i.e., the linear mapping between $\dot{x}$ and $\dot{y} = F'(x)\dot{x}$ and equivalently also that between $\overline{y}$ and $\overline{x} = \overline{y}F'(x)$ unchanged. It can be easily checked that this is the case if given an index j one updates first

$$c_{ik} += c_{ij}c_{jk}$$

either for fixed $i \succ j$ and all $k \prec j$, or for fixed $k \prec j$ and all $i \succ j$, and then sets $c_{ij} = 0$ or $c_{jk} = 0$, respectively. In other words, either the edge (j,i) or the edge (k,j) is eliminated from the graph. This leads to fill-in by the creation of new arcs, unless all updated c_{ik} were already nonzero beforehand. Eliminating all edges (k,j) with $k \prec j$ or all edges (j,i) with $i \succ j$ is equivalent and amounts to eliminating the vertex j completely from the graph. After all intermediate vertices $1 \leq j \leq l$ are eliminated in some arbitrary order, the remaining edges c_{ij} directly connect independent variables with dependent variables and are therefore entries of the Jacobian $F'(x)$. Hence, one refers to the *accumulation* of the Jacobian F' if all intermediate nodes are eliminated and to *preaccumulation* if some of them remain so that the Jacobian is represented by a simplified graph.

As we have indicated in the section on goal oriented differentiation one would have to carefully look at the problem function and the overall computational task to decide how much preaccumulation should be performed. Moreover, there are $\widetilde{l}!$ different orders in which a particular set of $\widetilde{l} \leq l$ intermediate nodes can be eliminated and even many more different ways of eliminating the corresponding set of edges. So far there have only been few studies of heuristic criteria for finding efficient elimination orderings down to an appropriate preaccumulation level [9].

Summary. First and second derivative vectors of the form $\dot{y} = F'(x)\dot{x}$, $\overline{x} = \overline{y}F'(x)$ and $\dot{\overline{x}} = \overline{y}F''(x)\dot{x}$ can be evaluated for a fixed small multiple of the temporal complexity of the underlying relation $y = F(x)$. The calculation of the gradient $\overline{x}$ and the second order adjoint $\dot{\overline{x}}$ by the basic reverse method may require storage of order $l \equiv$ #intermediates. This possibly unacceptable amount can be reduced to order $\log(l)$ at a slight increase in the operations count (see [8]).

Jacobians and one-sided projected Hessians can be composed column by column or row by row from vectors of the kind $\dot{y}$, $\overline{x}$ and $\dot{\overline{x}}$. For sparse derivative matrices row and/or column compression using suitable seed matrices of type CPR or NR allow a substantial reduction of the computational effort. In some cases the nonzero entries of derivative matrices may be redundant, so that their calculation should be avoided, if the overall computational goal can be reached in some other way. The attempt to evaluate derivative array with absolutely minimal effort leads to hard combinatorial problems.

See also: **Complexity theory: Quadratic programming**; **Complexity of degeneracy**; **Complexity classes in optimization**; **Information-based complexity and information-based optimization**; **Fractional combinatorial optimization**; **Complexity theory**; **Computational complexity theory**; **Mixed integer nonlinear programming**; **Parallel computing: Complexity classes**; **Kolmogorov complexity**; ***NP*-complete problems and proof methodology**.

References

[1] BAUER, F.L.: 'Computational graphs and rounding error', *SIAM J. Numer. Anal.* **11** (1974), 87–96.

[2] BERZ, M., BISCHOF, CH., CORLISS, G., AND GRIEWANK, A. (eds.): *Computational differentiation: Techniques, applications, and tools*, SIAM, 1996.

[3] BISCHOF, CH., CARLE, A., CORLISS, G., GRIEWANK, A., AND HOVLAND, P.: 'ADIFOR: Generating derivative codes from Fortran programs', *Scientif. Program.* **1** (1992), 1–29.

[4] COLEMAN, T.F., AND MORÉ, J.J.: 'Estimation of sparse Jacobian matrices and graph coloring problems', *SIAM J. Numer. Anal.* **20** (1984), 187–209.

[5] COLEMAN, T.F., AND VERMA, A.: 'Structure and efficient Jacobian calculation', in M. BERZ, CH. BISCHOF, G. CORLISS, AND A. GRIEWANK (eds.): *Computational Differentiation: Techniques, Applications, and Tools*, SIAM, 1996, pp. 149–159.

[6] CURTIS, A.R., POWELL, M.J.D., AND REID, J.K.: 'On the estimation of sparse Jacobian matrices', *J. Inst. Math. Appl.* **13** (1974), 117–119.

[7] GRIEWANK, A.: 'The chain rule revisited in scientific computing, I–II', *SIAM News* (May-July 1991).

[8] GRIEWANK, A.: 'Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation', *Optim. Methods and Software* **1** (1992), 35–54.

[9] GRIEWANK, A.: *Evaluating derivatives, principles and techniques of algorithmic differentiation*, Vol. 19 of *Frontiers in Appl. Math.*, SIAM, 2000.

[10] GRIEWANK, A., AND CORLISS, G.F. (eds.): *Automatic differentiation of algorithms: Theory, implementation, and application*, SIAM, 1991.

[11] GRIEWANK, A., AND REESE, S.: 'On the calculation of Jacobian matrices by the Markowitz rule', in A. GRIEWANK AND G.F. CORLISS (eds.): *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, SIAM, 1991, pp. 126–135.

[12] NEWSAM, G.N., AND RAMSDELL, J.D.: 'Estimation of sparse Jacobian matrices', *SIAM J. Alg. Discrete Meth.* **4** (1983), 404–417.

Andreas Griewank
Inst. Sci. Comput. Dept. Math. Techn. Univ. Dresden
Germany
E-mail address: griewank@math.tu-dresden.de

MSC2000: 65D25, 68W30
Key words and phrases: automatic differentiation, divided differences, computer algebra, symbolic manipulation.

COMPLEXITY THEORY

Complexity theory poses the question: How much computing time is required to solve a problem, as a function of the size of the problem? A similar questions may be asked about other computing resources like memory space. In the context of optimization, the commonly-asked complexity question is how much computing time, as a function of m and n, is required to solve a certain class of mathematical programming problems with n variables and m constraints. This form of asymptotic complexity analysis was introduced by J. Hartmanis and R.E. Stearns [4].

Several different complexity theories have been developed to address this question. The best known complexity theory is based on Turing machines. Before defining this term, we start with a definition of 'problem'. Formally, a *problem* is a function F that takes as input an *instance* and produces as output a *result*. For example, in the context of *linear programming* the instance would be a triple $(A, \mathbf{b}, \mathbf{c})$ specifying a standard-form linear program LP:

$$\begin{cases} \min & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} & A\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}. \end{cases}$$

The value of $F(A, \mathbf{b}, \mathbf{c})$ is the optimal value of the LP instance, or perhaps the optimizer. The range of F must also include special output values to signify an infeasible instance, an unbounded instance, or an ill-formed instance, e.g., dimensions of A and $\mathbf{b}$ are incompatible. Thus, in the context of complexity theory, the word 'problem' refers not to a specific instance but to a class of instances.

For a Turing machine, all instances must be specified as finite-length strings of symbols where the symbols are chosen from a fixed, finite alphabet. For LP and other optimization problems, a reasonable alphabet would include the ten digits and delimiter marks like decimal points, commas, parentheses. A cardinality argument shows that this stipulation of finite string over finite alphabet precludes the consideration of problems with arbitrary real number data. Thus, Turing machine solution of linear programming is generally restricted to rational or integer data. Rational and integer data are essentially equivalent since one can transform rational to integer by multiplying by a common denominator.

A second limitation of the Turing machine model is that there is no simple way to specify a general objective function or constraint function of an optimization problem as part of the input. There is a generalization of the Turing machine definition to overcome this limitation (so-called 'oracle' Turing machines), but in this article we limit attention to conventional Turing machines. This limitation means that our Turing machine complexity analysis focuses on optimization problems with predefined classes of objective functions and constraints in which the only free parameters are numeric data, e.g., $(A, \mathbf{b}, \mathbf{c})$ in linear programming.

A *Turing machine* (TM) is a computational device equipped with an infinitely-long tape used for memory and a controller with a finite program. The tape contains an infinite number of cells numbered $0, 1, 2, \ldots$, and each cell is capable of holding

one symbol chosen from a finite alphabet. The alphabet of the tape is a superset of the alphabet used for the input. Initially the tape contains the input instance written one symbol per cell starting at the left end of the tape (cell 0). The remaining cells contain a special symbol meaning 'blank'.

The Turing machine controller has a tape head that is above one cell of the tape at any particular time. The controller is always in a *state* chosen from a finite list of states. Finally, the TM obeys a finite list of *transition rules*. Each transition rule has the form: 'if the current symbol under the head is x and the current state is y, then change the symbol to x', change the state to y' and move the tape head one cell in direction d', where d is either 'left' or 'right'. Thus, a TM is fully specified by its input alphabet, its tape alphabet, its list of states, and its list of transition rules. If, for any given combination of current symbol/current state, there is at most one applicable transition rule, the TM is said to be *deterministic* else it is said to be *nondeterministic*. In this article we consider deterministic TMs only.

An *execution* of a Turing machine consists of a sequence of *moves*. Initially, as mentioned above, the input is written on the tape, the head is at position 0, and the machine is in a specially designated state called the 'start' state. The applicable transition rule is selected and executed, meaning that cell 0 is rewritten and the head is moved. Each execution of a transition rule is called a 'move'. The Turing machine continues to make moves until it reaches another special state called the 'halt' state.

The Turing machine is said to *solve* problem F, if given an input instance x, the TM (eventually) writes $F(x)$ on its tape starting at position 0, followed by blanks, and then halts. If for some input the TM could ever execute an illegal operation, e.g., move left from cell 0, or enter a state/symbol combination before halting for which there is no applicable transition rule, then it does not solve F. Furthermore, we require that the Turing machine can correctly handle every possible finite string that can be written with the input alphabet. For incorrectly formatted strings, the Turing machine should output a special string indicating incorrect formatting.

The *running time* of a Turing machine for a given input instance is the number of moves required before it halts. The running time for the whole problem F is usually expressed as a function of the *size of the input*, that is, the number of symbols in the input string.

It can be proved using lengthy constructions that a Turing machine is capable of all the operations of an ordinary computer: it can simulate consecutively numbered memory cells each holding a separate integer or rational number that are individually addressable, it can multiply, divide, add, and subtract two such numbers, etc. For a more detailed treatment of Turing machines, see [5].

A Turing machine is said to solve problem F in *polynomial time* if its running time is no more than a polynomial function of the size of the input. Examples of optimization problems that can be solved in polynomial time include linear and convex quadratic programming.

A *decision problem* is a problem F in which the range of F consists of just two entries, 'YES' and 'NO'. Optimization problems can often be recast as decision problems. For instance, in the case of linear programming, the input instance consists of $(A, \mathbf{b}, \mathbf{c}, r)$, where r is a rational number, and the TM outputs 'YES' if the minimal solution to the LP problem is r or less, else it outputs 'NO'. For incorrectly formatted and infeasible problems, the TM also outputs 'NO'. The decision problem F partitions the input space into two sets of strings, 'YES'-instances and 'NO'-instances. A synonym for 'decision problem' is *language recognition problem*.

The set $\mathcal{P}$ is defined to be all decision problems that can be solved in polynomial time. This set includes linear programming (as recast in the previous paragraph) and many combinatorial optimization problems such as the minimum spanning tree problem and the shortest path problem (cf. also **Shortest path tree algorithms**).

Many interesting problems, such as *nonconvex quadratic programming* and *Boolean satisfiability*, are not known to be in $\mathcal{P}$, but are also not proven to lie outside of $\mathcal{P}$. To analyze these cases, we introduce a second complexity class called *NP*. A

decision problem F is said to lie in *NP* if there exists a polynomial time 'certificate-checking' machine M outputting 'YES' or 'NO', and polynomials $p(\cdot)$, $q(\cdot)$ with the following properties. For every 'YES'-instance x of F, there exists another string y, called the *certificate* of x, such that the size of y is no more than $p(\text{size}(x))$ such that the pair (x, y) (i.e., the string concatenation of x and y properly delimited) is a 'YES'-instance of M. On the other hand, for every 'NO'-instance x of F, and for every possible certificate y, M outputs 'NO' for the input pair (x, y). Finally, in both cases, M is required to run in time no more than $q(\text{size}(x) + \text{size}(y))$.

Notice this definition is asymmetric between 'YES'- and 'NO'-instances. Thus, it is not necessarily true that if problem F is in *NP*, then the problem $\overline{F}$ that results from complementing F's output (i.e., 'YES'-instances of F are 'NO'-instances of $\overline{F}$ and vice-versa) is still in *NP*. Indeed, the question of whether $F \in NP \Leftrightarrow \overline{F} \in NP$ is a well-known open question.

Another observation from the above definition is that $\mathcal{P} \subset NP$. In particular, if a decision problem F lies in $\mathcal{P}$, then it has polynomial time Turing machine T that distinguishes 'YES'-instances from 'NO'-instances. In this case, it is simple to design the certificate checking machine M needed for the definition of *NP*: in particular, M takes as input (x, y), it discards y, i.e., overwrites it with blank cells, and then switches to running T on x.

The most famous open question in complexity theory is whether this containment is actually equality, i.e., whether $\mathcal{P} = NP$. It turns out that the $\mathcal{P} = NP$ question hinges on *NP-complete problems*, the prototype of which is the satisfiability problem. The *satisfiability problem* is as follows. An instance is a Boolean formula with variables $x_1, \ldots, x_m$, conjunctions, disjunctions and complement operations. For example, $x_1 \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (x_2 \vee x_3)$ is a satisfiability instance. The decision problem is to determine whether there is an *assignment* of the variables, each one either 'TRUE' or 'FALSE', to make the entire formula true following the usual laws of boolean algebra. For example, the preceding formula is a 'YES'-instance because there is a satisfying assignment, namely x_1 ='TRUE', x_2 ='FALSE', x_3 ='TRUE'. It is easy to see that this problem is in *NP*: the certificate for a 'YES'-instance is the satisfying assignment. The certificate-checking machine M substitutes the satisfying assignment into the formula and verifies that the formula evaluates as 'TRUE'. Thus, every satisfiable formula has a certificate, but every nonsatisfiable instance is rejected by M no matter what certificate is given.

S.A. Cook [2] proved that every problem in *NP* is polynomially transformable to satisfiability. We say that decision problem F is *polynomially transformable* to F' if there exists a Turing machine T that takes as input an instance x of F and produces as output an instance x' of F', such that the running time of T is no more than a polynomial in the size of the input, and such that x is a 'YES'-instance of F if and only if x' is a 'YES'-instance of F'.

Cook's result means that given any decision problem F in *NP*, there is a Turing machine M depending on F that takes as input an instance x of F and proceeds as follows. In polynomial time, M constructs a Boolean formula x' from x such that x' is satisfiable if and only x is a 'YES'-instance of F. The construction of M is as as follows. The Boolean formula x' simulates the action of the certificate-checking machine in on x. The actual entries of the certificate are represented by unknown Boolean variables, as are the entries on the tape of M after the first move. The formula is composed of clauses that require the Turing machine to obey all transition rules and to end up halting with 'YES' as the output.

Thus, Cook's theorem implies that if there were a polynomial time algorithm for satisfiability, then there would be a polynomial time algorithm for every other problem in *NP*. Thus, the famous open question 'is $\mathcal{P} = NP$?' is now reduced to the (apparently) simpler question 'is there a polynomial time algorithm for the satisfiability problem?'

It is not yet (1999) known whether the answer to either question in the last paragraph is 'yes'. But many in the field suspect that the answer is 'no', i.e., many believe that there is no polynomial time algorithm for satisfiability. If indeed it is proved some day that no such polynomial time algorithm exists, we would say that satisfiability is

intractable.

A decision problem F is said to be *NP-complete* if it has these two properties, namely

1) F is in *NP*; and
2) every problem in *NP* can be polynomially transformed to F.

Cook's result can be restated as: satisfiability is *NP*-complete. Furthermore, since polynomial transformations can be composed, any problem F in *NP* to which any known *NP*-complete problem F' can be transformed must itself be *NP*-complete. After Cook's result was announced, R.M. Karp [6] showed that many well-known combinatorial problems, such as the *Hamiltonian cycle problem* ('given an undirected graph, is there a cycle containing each vertex exactly once?') and the *max-clique problem* ('given an undirected graph and an integer k, is there a set of k vertices that are all mutually connected by edges?'), are *NP*-complete. By 1979, already thousands of problems were known to be *NP*-complete and many were catalogued in [3]. A proof that a problem is *NP*-complete is regarded as strong evidence of the problem's intractability.

Although the first batch of *NP*-completeness proofs applied to combinatorial problems, many continuous optimization problems are also known to be *NP*-complete; see **NP-complete problems and proof methodology**.

A generalization of '*NP*-complete' is the notion of '*NP*-hard'. A problem F is said to be *NP-hard* if satisfiability (or any other *NP*-complete problem) can be polynomially transformed to F. Thus, an *NP*-hard problem does not necessarily lie in *NP*. Indeed, the term *NP*-hard is often used to describe problems that are not even decision problems.

The Turing machine is not the only model of complexity used in the literature. In fact, some feel that the TM is inadequate for modeling continuous optimization problems. Most continuous optimization problem are based on computation with real numbers, but true real number computation is not possible with a TM. One model of real number computation is the *information-based model*. In this model, an algorithm is composed of operations on real numbers. Operations on real numbers are often counted as cost-free in this model, and the only costly operation is the evaluation of the functions defining the objective and constraints of the optimization problem. The objective functions and constraints are considered external black-box subroutines that take as input a vector and return as output the value of the function and possibly derivative values. In information-based complexity, a parameter $\epsilon > 0$ that specifies the desired degree of accuracy in the solution is always part of the input, since the information-based model rarely permits any problem to be solved exactly. This information-based model was used to analyze the *ellipsoid method* by its inventors D.B. Yudin and A.S. Nemirovsky [7]. It has also been used to analyze complexity of *local optimization* by S.A. Vavasis [12]. This model has also been used extensively to analyze other numerical algorithms not related to optimization, e.g., quadrature and other linear problems [9].

A second model of computation is a *real number model* in which each operation is unit cost, and in which there is no concept of external black-box function evaluation. In this model it is possible to develop real number analogs of complexity classes $\mathcal{P}$ and *NP*, and also a reasonable definition for *NP*-complete; see [1]. This model can be used to analyze linear programming and other problems specified by a finite number of real parameters. The complexity of linear programming problem in this model is not fully understood (see [13] and [10]).

For a more detailed look at complexity in optimization up to 1991, see [11]. For a more recent collection of papers on this topic, see [8].

See also: **Complexity theory: Quadratic programming**; **Complexity of degeneracy**; **Complexity classes in optimization**; **Information-based complexity and information-based optimization**; **Fractional combinatorial optimization**; **Complexity of gradients, Jacobians, and Hessians**; **Computational complexity theory**; **Mixed integer nonlinear programming**; **Parallel computing: Complexity classes**; **Kolmogorov complexity**; ***NP*-complete problems and proof methodology**.

References

[1] Blum, L., Cucker, F., Shub, M., and Smale, S.:

Complexity and real computation, Springer, 1998.

[2] COOK, S.A.: 'The complexity of theorem-proving procedures': *Proc. 3rd Annual ACM Symp. Theory of Computing*, 1971, pp. 151–158.

[3] GAREY, M.R., AND JOHNSON, D.S.: *Computers and intractability: A guide to the theory of NP-completeness*, Freeman, 1979.

[4] HARTMANIS, J., AND STEARNS, R.E.: 'On the computational complexity of algorithms', *Trans. Amer. Math. Soc.* **117** (1965), 285–306.

[5] HOPCROFT, J., AND ULLMAN, J.: *Introduction to automata theory, languages and computation*, Addison-Wesley, 1979.

[6] KARP, R.M.: 'Reducibility among combinatorial problems', in R.E. MILLER AND J.W. THATCHER (eds.): *Complexity of Computer Computations*, Plenum, 1972, pp. 85–103.

[7] NEMIROVSKY, A.S., AND YUDIN, D.B.: *Problem complexity and method efficiency in optimization*, Wiley, 1983. (Translated from the Russian.)

[8] PARDALOS, P.M.: *Complexity in numerical optimization*, World Sci., 1993.

[9] TRAUB, J.F., WASILKOWSKI, G.W., AND WOŹNIAKOWSKI, H.: *Information-based complexity*, Acad. Press, 1988.

[10] TRAUB, J.F., AND WOŹNIAKOWSKI, H.: 'Complexity of linear programming', *Oper. Res. Lett.* **1** (1982), 59–62.

[11] VAVASIS, S.A.: *Nonlinear optimization: Complexity issues*, Oxford Univ. Press, 1991.

[12] VAVASIS, S.A.: 'Black-box complexity of local minimization', *SIAM J. Optim.* **3** (1993), 60–80.

[13] VAVASIS, S.A., AND YE, Y.: 'A primal-dual interior point method whose running time depends only on the constraint matrix', *Math. Program.* **74** (1996), 79–120.

Stephen A. Vavasis
Cornell Univ.
Ithaca, NY, USA
E-mail address: vavasis@cs.cornell.edu

MSC2000: 90C60

Key words and phrases: complexity, Turing machine, *NP*-hard, *NP*-complete, polynomial time, information-based complexity, real number model, decision problem.

COMPLEXITY THEORY: QUADRATIC PROGRAMMING

Nowhere in optimization is the dichotomy between convex and nonconvex programming more apparent than in complexity issues for quadratic programming. *Quadratic programming*, abbreviated QP, refers to minimizing a quadratic function $q(\mathbf{x}) = \mathbf{x}^\top H\mathbf{x}/2 + \mathbf{c}^\top \mathbf{x}$ subject to linear constraints $A\mathbf{x} \geq \mathbf{b}$. The problem is thus specified by the four-tuple $(H, A, \mathbf{b}, \mathbf{c})$ where H is a symmetric $n \times n$ matrix, A is an $m \times n$ matrix, $\mathbf{b}$ is an m-vector and $\mathbf{c}$ is an n-vector. Minimizing a quadratic function subject to convex quadratic constraints is also an interesting problem and is considered at the end of this article. The quadratic function $q(\mathbf{x})$ is said to be *convex* if the matrix H is positive semidefinite. A special case of a convex function is when $H = 0$, in which case the problem is now called *linear programming*.

Convex quadratic programming inherits all the desirable attributes of the general convex programming. In particular, there is no local minimizer other than the global minimizer(s). Furthermore, general convex programming techniques like the *ellipsoid method* and *interior point methods* can be applied.

With either the ellipsoid or interior point method, convex quadratic programming can be solved in *polynomial time*. In more detail, assume that $(H, A, \mathbf{b}, \mathbf{c})$ contain all integer data so that the problem is finitely represented for a *Turing machine* (see **Complexity theory**). Let L denote the length of the input data, that is, the total number of digits to write $(H, A, \mathbf{b}, \mathbf{c})$. Assume $L \geq m^2$ since H has m^2 entries. Then M.K. Kozlov et al. [9] showed that the ellipsoid algorithm of A.S. Nemirovsky and D.B. Yudin [14] can solve a convex QP instance in time $O(m^2 L)$ iterations, where each iteration requires $O(m^2)$ arithmetic operations on integers, each of which has at most $O(L)$ digits. This result built on an analogous result for the LP case by L.G. Hačijan (also spelled L.G. Khachiyan) [4]. Thus, the total running time of this algorithm is polynomial in the size of the input. Note that the the global minimizer for quadratic programming (either convex or nonconvex), if it exists, can be written down with $O(nL)$ digits, and hence computing the true global minimizer in a Turing machine setting is possible.

Later, S. Kapoor and P.M. Vaidya [6] and Y. Ye and E. Tse [29] proved that an interior point method can solve convex quadratic programming in polynomial time under similar assumptions. This result built on the earlier result for the LP case by N.K. Karmarkar [7]. The best known running time for an interior point method for convex

QP is $O(m^{1/2}L)$ iterations, each iteration requiring $O(m^3)$ arithmetic operations on integers each of which has at most $O(L)$ digits and is based on work by J. Renegar [17].

The running times of both the ellipsoid and interior point algorithms are 'weakly' polynomial, meaning that the number of arithmetic operations is bounded by a polynomial in L rather than by a polynomial in m and n. In contrast, polynomial time algorithms for other problems like solving a system of linear equations or finding a minimum flow a network are *strongly polynomial time*, meaning that the number of operations is bounded by a polynomial in the combinatorial dimension of the input data. A well-known open (1999) question asks whether there is a strongly polynomial time algorithm for convex QP (or, more specifically, for LP). A strongly polynomial algorithm would involve a number of arithmetic operations polynomially bounded in m, n, in which each operation involves integers with a number of digits bounded by a polynomial in L. Some progress related to this question is as follows. If the dimension n is restricted to a small integer, then QP can be solved in time linear in m. This result is due to I. Adler and R. Shamir [1] and builds on [10] and [2]. Since the constant of proportionality (or perhaps an additive term) is exponential in n, this algorithm is not so useful except when $n \ll m$. An example would be quadratic programming arising from a geometric problem, such as finding the point in a 3D polyhedron closest to the origin.

In the case of linear programming, a modified ellipsoid algorithm has a number of operations depending only on L_A, where L_A is the number of digits in A (i.e., the number of operations no longer depends on $\mathbf{b}$ or $\mathbf{c}$), a result due to É. Tardos [19] and extended in [25]. Finally, some special cases of quadratic programming are known to be solvable in strongly polynomial time such as the *convex quadratic knapsack problem* [5] which is:

$$\begin{cases} \min & q_1(x_1) + \cdots + q_n(x_n) \\ \text{s.t.} & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \\ & \mathbf{b}^\top \mathbf{x} = \gamma. \end{cases} \tag{1}$$

Here $q_1, \ldots, q_n$ are convex quadratic functions of one variables (each specified by a quadratic and a linear coefficient) $\mathbf{l}$, $\mathbf{u}$, $\mathbf{b}$, γ are also part of the problem data.

Nonconvex quadratic programming is much harder than convex quadratic programming. If H is not positive semidefinite, then the QP instance is said to be *nonconvex*. A special case of nonconvex problems is when H is negative semidefinite, in which case the problem is said to be *concave quadratic programming*. When H is neither positive nor negative semidefinite, the problem is *indefinite*. Nonconvex quadratic programming was shown to be *NP*-hard by S. Sahni [18]. If the problem is posed as a decision problem, then it lies in *NP* (and is therefore *NP*-complete), a result due to S.A. Vavasis [20]. (See **Complexity theory** or **NP-complete problems and proof methodology** for the definitions of *NP*-complete and *NP*-hard.) Even the problem of finding a *local minimizer* is known to be *NP*-hard, a result due to K.G. Murty and S.N. Kabadi [13].

Many restricted versions of the problem are still *NP*-hard. The nonconvex quadratic knapsack problem, that is, (1) with general (not necessarily convex) quadratic functions $q_1, \ldots, q_n$, is *NP*-hard [18]. QP with only *box constraints*, that is, minimize $q(\mathbf{x})$ subject to $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$, is also *NP*-hard. Similarly, minimizing $q(\mathbf{x})$ subject to *simplicial constraints*, that is, constraints $\mathbf{x} \geq \mathbf{0}$ and $x_1 + \cdots + x_n = 1$, is *NP*-hard as proved in [15] using a theorem of T.S. Motzkin and E.G. Straus [12]. The simplicial case is interesting because minimizing either a concave or convex quadratic function on a simplex can be solved in polynomial time. P.M. Pardalos and Vavasis [16] showed that quadratic programming in which H has a single negative eigenvalue (i.e., H is 'almost' positive semidefinite) is *NP*-hard.

The hardness results have motivated a search for approximation algorithms to nonconvex quadratic programming problems. Vavasis [24], [22] proposed approximation algorithms for concave and indefinite QP in which the complexity depends exponentially on the number of negative eigenvalues of H. An additional result is a fully polynomial time approximation scheme for the indefinite knapsack problem. Ye [28] gave a constant-factor polynomial time approximation scheme for indefinite qua-

dratic programming with box constraints.

Because computing even a local minimum of a quadratic programming instance is hard, several researchers have looked at approximations and special cases for the local minimization problem. J.J. Moré and Vavasis [11] proved that a local minimizer for the concave knapsack problem can be found in polynomial time; this result was extended to the indefinite case in [23]. Ye [28] gave a polynomial time algorithm to find an approximate *KKT point* of general nonconvex QP.

So far we have considered only linear constraints. A *convex quadratic constraint*, also called an *ellipsoidal constraint*, is a constraint of the form $(\mathbf{x} - \mathbf{c})^\top A(\mathbf{x} - \mathbf{c}) \leq 1$, where A is a symmetric positive semidefinite matrix. The problem of minimizing a nonconvex quadratic function subject to a single ellipsoidal constraint is called the *trust region problem* and has received extensive attention in the literature because algorithms to solve this problem are often used as subroutines by general-purpose optimization algorithms. A polynomial time algorithm for the trust region problem was proposed independently by Ye [27] and Karmarkar [8]. The sense in which this algorithm is 'polynomial time' is weaker than the analogous claim for QP because in the trust region case, the optimizer $\mathbf{x}$ cannot be written in a finite number of digits even if the input data is all integer (because the solution may be irrational). But Vavasis and R. Zippel [26] showed nonetheless that this algorithm leads to a proof that the associated decision problem lies in $\mathcal{P}$. The trust region problem is thus one of the very few nonconvex optimization problem solvable in polynomial time. M. Fu, Z.-Q. Luo and Ye [3] have considered generalizing this result to more than one ellipsoidal constraint, although the results are not as strong as the single-constraint case.

All of the pre-1991 material in this article is covered in more depth by [21].

See also: **Complexity theory; Complexity of degeneracy; Complexity classes in optimization; Information-based complexity and information-based optimization; Fractional combinatorial optimization; Complexity of gradients, Jacobians, and Hessians; Computational complexity theory; Mixed integer nonlinear programming; Parallel computing: Complexity classes; Kolmogorov complexity; *NP*-complete problems and proof methodology; Quadratic programming with bound constraints; Quadratic knapsack; Quadratic programming over an ellipsoid; Quadratic fractional programming: Dinkelbach method; Quadratic assignment problem; Standard quadratic optimization problems: Theory; Standard quadratic optimization problems: Algorithms; Standard quadratic optimization problems: Applications.**

References

[1] ADLER, I., AND SHAMIR, R.: 'A randomization scheme for speeding up algorithms for linear and convex quadratic programming problems with a high constraints-to-variables ratio', *Math. Program.* **61** (1993), 39–52.

[2] CLARKSON, K.L.: 'Las Vegas algorithms for linear and integer programming when the dimension is small', *J. ACM* **42**, no. 2 (1995), 488–499.

[3] FU, M., LUO, Z.-Q., AND YE, Y.: 'Approximation algorithms for quadratic programming', *Working Paper Dept. Management Sci. Univ. Iowa* (1996).

[4] HAČIJAN, L.G.: 'A polynomial algorithm in linear programming', *Soviet Math. Dokl.* **20** (1979), 191–194. (*Dokl. Akad. Nauk SSSR* **244** (1979), 1093–1096.)

[5] HELGASON, R., KENNINGTON, J., AND LALL, H.: 'A polynomially bounded algorithm for a singly constrained quadratic program', *Math. Program.* **18** (1980), 338–343.

[6] KAPOOR, S., AND VAIDYA, P.M.: 'Fast algorithms for convex quadratic programming and multicommodity flows': *Proc. 18th Annual ACM Symp. Theory of Computing*, 1986, pp. 147–159.

[7] KARMARKAR, N.: 'A new polynomial-time algorithm for linear programming', *Combinatorica* **4** (1984), 373–395.

[8] KARMARKAR, N.: 'An interior-point approach to NP-complete problems', *Preprint* (1989).

[9] KOZLOV, M.K., TARASOV, S.P., AND HAČIJAN, L.G.: 'Polynomial solvability of convex quadratic programming', *Soviet Math. Dokl.* **20** (1979), 1108–1111. (*Dokl. Akad. Nauk SSSR* **248** (1979), 1049–1051.)

[10] MEGIDDO, N.: 'Linear programming in linear time when the dimension is fixed', *J. ACM* **31** (1984), 114–127.

[11] MORÉ, J.J., AND VAVASIS, S.A.: 'On the solution of concave knapsack problems', *Math. Program.* **49** (1991), 397–411.

[12] MOTZKIN, T.S., AND STRAUS, E.G.: 'Maxima for graphs and a new proof of a theorem of Turán', *Canad. J. Math.* **17** (1965), 533–540.

[13] MURTY, K.G., AND KABADI, S.N.: 'Some NP-complete problems in quadratic and nonlinear programming', *Math. Program.* **39** (1987), 117–129.
[14] NEMIROVSKY, A.S., , AND YUDIN, D.B.: *Problem complexity and method efficiency in optimization*, Wiley, 1983. (Translated from the Russian.)
[15] PARDALOS, P.M., HAN, C., AND YE, Y.: 'Algorithms for the solution of quadratic knapsack problems', *Linear Alg. & Its Appl.* **152** (1991), 69–91.
[16] PARDALOS, P.M., AND VAVASIS, S.A.: 'Quadratic programming with one negative eigenvalue is NP-hard', *J. Global Optim.* **1** (1991), 15–22.
[17] RENEGAR, J.: 'A polynomial-time algorithm based on Newton's method for linear programming', *Math. Program.* **40** (1988), 59–94.
[18] SAHNI, S.: 'Computationally related problems', *SIAM J. Comput.* **3** (1974), 262–279.
[19] TARDOS, E.: 'A strongly polynomial algorithm to solve combinatorial linear programs', *Oper. Res.* **34** (1986), 250–256.
[20] VAVASIS, S.A.: 'Quadratic programming is in NP', *Inform. Process. Lett.* **36** (1990), 73–77.
[21] VAVASIS, S.A.: *Nonlinear optimization: Complexity issues*, Oxford Univ. Press, 1991.
[22] VAVASIS, S.A.: 'Approximation algorithms for indefinite quadratic programming', *Math. Program.* **57** (1992), 279–311.
[23] VAVASIS, S.A.: 'Local minima for indefinite quadratic knapsack problems', *Math. Program.* **54** (1992), 127–153.
[24] VAVASIS, S.A.: 'On approximation algorithms for concave quadratic programming', in C.A. FLOUDAS AND P.M. PARDALOS (eds.): *Recent Advances in Global Optimization*, Princeton Univ. Press, 1992, pp. 3–18.
[25] VAVASIS, S.A., AND YE, Y.: 'A primal-dual interior point method whose running time depends only on the constraint matrix', *Math. Program.* **74** (1996), 79–120.
[26] VAVASIS, S.A., AND ZIPPEL, R.: 'Proving polynomial-time for sphere-constrained quadratic programming', *Techn. Report Dept. Computer Sci. Cornell Univ.* **90-1182** (1990).
[27] YE, Y.: 'On affine scaling algorithms for nonconvex quadratic programming', *Math. Program.* **56** (1992), 285–300.
[28] YE, Y.: 'Approximating quadratic programming with bound constraints', *Working Paper Dept. Management Sci. Univ. Iowa* (1997).
[29] YE, Y., AND TSE, E.: 'An extension of Karmarkar's projective algorithm for convex quadratic programming', *Math. Program.* **44** (1989), 157–179.

Stephen A. Vavasis
Cornell Univ.
Ithaca, NY, USA
E-mail address: vavasis@cs.cornell.edu

MSC2000: 90C60

Key words and phrases: quadratic programming, complexity, *NP*-hard, *NP*-complete, local minimization, trust region problem, polynomial time, strongly polynomial time, knapsack problem, simplicial constraints, box constraints, ellipsoid method, interior point methods, approximation algorithms.

COMPOSITE NONSMOOTH OPTIMIZATION, *CNSO*

By composite nonsmooth optimization (CNSO) we mean a class of optimization problems involving composite functions of the form $f(x) := g(F(x))$, where $F\colon \mathbf{R}^n \to \mathbf{R}^m$ is a (differentiable) smooth map and $g\colon \mathbf{R}^m \to \mathbf{R}$ is a nonsmooth function. The function g is often a nonsmooth convex function. Problems of CNSO occur when solving nonlinear equations $F_i(x) = 0$, $i = 1, \ldots, m$, by minimizing the norm $\|F(x)\|$. Similar problems arise when finding a feasible point of a system of nonlinear inequalities $F_i(x) \leq 0$, $i = 1, \ldots, m$, by minimizing $\|F(x)^+\|$ where $F_i^+ = \max(F_i, 0)$. Composite functions f also appear in the form of an exact penalty function when solving a nonlinear programming problem. Another type of CNSO problem which frequently arises in (electrical) engineering [4] is to minimize the *max-function* $\max_i F_i(x)$, where the maximum is taken over some (finite) set. All these examples can be cast within the structure of CNSO. Moreover, CNSO provides a unified framework in which to study theoretical properties and convergence behavior of various numerical methods for constrained optimization problems. There have been many contributors to the study of CNSO problems both in finite and infinite dimensions. (See for example [2], [6], [7], [13], [15], [16], [10], [11], [19].) In this article we only discuss different forms of composite model problems in finite dimensions and provide a brief account of their first and *second order Lagrangian theory of CNSO problems*. The implications for numerical optimization are not discussed here. For details on this see, for instance, [1], [6].

Real-Valued CNSO. Consider the problem

$$\text{(P)} \quad \min_{x \in \mathbf{R}^n} g(F(x)).$$

Notably, A.D. Ioffe [7], [8], [9] provided the theoretical foundation for CNSO problems in the case

where the function g is sublinear (convex plus positively homogeneous). Then J.V. Burke [2] extended the theory to the case where g is convex. A fundamental local dualization technique plays a significant role in the development of first - and second order *Lagrangian theory* for (P). To see the dualization result, let us define the *Lagrangian* of (P) as

$$L(x, y^*) = \langle y^*, F(x)\rangle - g^*(y^*),$$

where g^* is the Fenchel conjugate of g [14]. Let

$$L_0(z) = \{y^*\colon y^* \in \partial g(F(z)),\ y^*F'(z) = 0\}$$

and let

$$L_{\eta\epsilon}(z) = \{y^*\colon y^* \in \partial_\epsilon g(F(z)),\ \|y^*F'(z)\| \le \eta\},$$

where $F'(z)$ is the derivative of F at z, $\partial g(y)$ is the *convex subdifferential* of g at y, $\epsilon > 0$, $\eta > 0$ and $\partial_\epsilon g(F(z))$ is the *ϵ-subdifferential* of g at $F(z)$. The set $L_0(z)$ is the set of Lagrange multipliers for (P) at z (see [2], [11]). Define

$$\phi_{\eta\epsilon}(x) := \max_{y^* \in L_{\eta\epsilon}(z)} L(x, y^*).$$

A general form of the *Ioffe–Burke local dualization* result [11] states that if g is a lower semicontinuous convex function and F is a locally Lipschitzian and (Gâteaux) differentiable function then the following statements are equivalent:

i) $g(F(x))$ attains a local minimum at z.

ii) $L_0(z) \neq \emptyset$ and $\phi_{\eta\epsilon}$ attains a local minimum at z, for *any* $\eta > 0$, $\epsilon > 0$.

iii) $L_0(z) \neq \emptyset$ and $\phi_{\eta\epsilon}$ attains a local minimum at z, for *some* $\eta > 0$, $\epsilon > 0$.

These conditions also provide first order Lagrangian conditions for (P). Moreover, this local *dualization* result and a generalized Taylor expansion of V. Jeyakumar and X.Q. Yang [11] yield second order optimality conditions for (P). If g is a lower semicontinuous convex function and F is a differentiable map with locally Lipschitzian derivative F' (i.e. $C^{1,1}$) then a necessary condition for $a \in \mathbf{R}^n$ to be a local minimizer of (P) is

$$\max_{y^* \in L_0(a)} L^{\circ\circ}(a, y^*; u, u) \ge 0, \quad \forall u \in \overline{K(a)}.$$

On the other hand if $a \in \mathbf{R}^n$, $L_0(a) \neq \emptyset$ and

$$\max_{y^* \in L_0(a)} -L^{\circ\circ}(a, y^*; u, -u) > 0, \quad \forall u \in D(a),$$

then a is a strict local minimizer of order 2 for (P), i.e., there exist $\epsilon > 0$, $\rho > 0$ such that whenever $\|x - a\| < \rho$, $f(x) \ge f(a) + \epsilon\|x - a\|^2$. Here

$$K(a) = \left\{ u \in \mathbf{R}^n\colon \begin{array}{c} \exists t > 0, \\ g(F(a) + tF'(a)u) \\ \le g(F(a)) \end{array} \right\},$$

$D(a) = \{u \in \mathbf{R}^n\colon f'(a; u) \le 0\}$, and the directional derivative of f at a is given by $f'(a; d) = g'(F(a); F'(a)d)$. The *generalized second order directional derivative* of L at a in the directions $(u, v) \in \mathbf{R}^n \times \mathbf{R}^n$, $L^{\circ\circ}(a; u, v)$, is defined by

$$\limsup_{y \to a,\ s \to 0} \frac{\langle \nabla L(y + su), v\rangle - \langle \nabla L(y), v\rangle}{s}.$$

Special cases of these optimality conditions under twice continuously differentiability hypothesis can be found in [2], [9]. Composite problems where the map F is $C^{1,1}$, but is not necessarily twice continuously differentiable are discussed in [19].

Extended Real-Valued CNSO. A composite problem form which has greater versatility than the traditional form (P) is the following nonfinite valued problem [15], [16]

$$\text{(PE)} \quad \begin{cases} \min & g(F(x)) \\ \text{s.t.} & x \in \mathbf{R}^n, \\ & F(x) \in \operatorname{dom}(g), \end{cases}$$

where $g\colon \mathbf{R}^m \to \mathbf{R} \cup \{+\infty\}$ is a convex function and $F\colon \mathbf{R}^n \to \mathbf{R}^m$ is a smooth map. For instance, constrained CNSO problems of the form,

$$\begin{cases} \min & g_0(F_0(x)) \\ \text{s.t.} & x \in C, \\ & g_j(F_j(x)) \le 0, \quad j = 1, \ldots, m, \end{cases}$$

studied in [10], [17], can be re-written in the form of (PE) [11]. Here C is a closed convex subset of $\mathbf{R}^n$, g_j, $j = 0, \ldots, m$, are locally Lipschitz functions and F_j, $j = 0, \ldots, m$, are differentiable functions. Optimality conditions for (PE) can be derived by reducing (PE) to a real-valued minimization problem as it was shown in [3]. This requires a regularity condition known as a constraint qualification in the nonlinear programming literature. The following regularity condition, introduced in [15] as a *basic constraint qualification*, permits one to establish a reduction theorem. If

$g\colon \mathbf{R}^m \to \mathbf{R} \cup \{+\infty\}$ is a lower semicontinuous convex function and if $F\colon \mathbf{R}^n \to \mathbf{R}^m$ is locally Lipschitzian then the function $f(x) := g(F(x))$ is said to satisfy the basic constraint qualification at a point $x \in \operatorname{dom}(f)$ if the only point $w \in N(F(x)|\operatorname{dom}(g))$ for which $0 \in w^\top \partial F(x)$ is $w = 0$, where $N(F(x)|\operatorname{dom}(g))$ is the normal cone to $\operatorname{dom}(g)$ at $F(x)$ and $\partial F(x)$ is the generalized Jacobian of F at x [5]. The basic constraint qualification is equivalent to the *Mangasarian–Fromovitz constraint qualification* for the standard nonlinear programming problem with inequality and equality constraints (see [15]). The *Burke–Poliquin reduction* result gives us the following second order conditions for (PE). For problem (PE), suppose that $F(a) \in \operatorname{dom}(g)$, g is lower semicontinuous convex and F is $C^{1,1}$. Then the following statements (i) and (ii) hold.

i) If a is a local minimizer of (PE) at which the basic constraint qualification holds, then

$$\max_{y^* \in L_0(a)} L^{\circ\circ}(a, y^*; u, u) \geq 0, \quad \forall u \in \overline{K(a)}.$$

ii) If $L_0(a) \neq \emptyset$ and

$$\max_{y^* \in L_0(a)} -L^{\circ\circ}(a, y^*; u, -u) > 0, \ \forall u \in D(a),$$

then a is a strict local minimizer of order 2 for (PE).

With the aid of a representation condition, second order conditions can also be obtained for a *global minimizer* of (PE) in the case where F is twice strictly differentiable. This was shown in [19]. The problems (PE) have also been extensively studied by R.T. Rockafellar [15], [16] in the case where F is twice continuously differentiable and g is a proper convex function that is *piecewise linear quadratic* in the sense that the $\operatorname{dom}(g)$ is expressible as the union of finitely many polyhedral sets, relative to each of which g is given by the formula that is quadratic (or affine).

Multi-objective CNSO. Nonsmooth vector optimization problems (cf. **Vector optimization**) where the functions involved are compositions of convex functions and smooth functions arise in various applications. The following model problem was examined in [12]:

$$\text{(MP)} \quad \begin{cases} \text{V} - \min & (f_1(F_1(x)), \ldots, , f_p(F_p(x))) \\ \text{s.t.} & x \in C, \\ & g_j(G_j(x)) \leq 0, \\ & \quad j = 1, \ldots, m, \end{cases}$$

where C is a convex subset of $\mathbf{R}^n$, f_i, g_j are real valued convex functions on $\mathbf{R}^n$, F_i, G_j are locally Lipschitz and differentiable functions from $\mathbf{R}^n$ into $\mathbf{R}^n$. Note here that 'V-min' stands for vector minimization. This model is broad and flexible enough to cover many common types of vector optimization problems. In particular, this model includes the penalty representation of the standard vector nonlinear programming problems, examined in [18], and many vector approximation problems. By employing the Clarke subdifferential, first order Lagrangian optimality and duality results can be discussed as it was shown in [12]. Second order optimality conditions for a special case of the problem (MP) are discussed in [21], [19]

See also: **Nonsmooth and smoothing methods for nonlinear complementarity problems and variational inequalities; Solving hemivariational inequalities by nonsmooth optimization methods; Nonconvex-nonsmooth calculus of variations.**

References

[1] BURKE, J.V.: 'Descent methods for composite nondifferentiable optimization problems', *Math. Program.* **33** (1985), 260–279.

[2] BURKE, J.V.: 'Second-order necessary and sufficient conditions for convex composite NDO', *Math. Program.* **38** (1987), 287–302.

[3] BURKE, J.V., AND POLIQUIN, R.A.: 'Optimality conditions for nonfinite valued convex composite functions', *Math. Program. B* **57** (1992), 103–120.

[4] CHARALAMBOUS, C.: 'Acceleration of the least p^{th}- algorithm for minimax optimization with engineering applications', *Math. Program.* **17** (1979), 270–297.

[5] CLARKE, F.H.: *Optimization and nonsmooth analysis*, Wiley, 1983.

[6] FLETCHER, R.: *Practical methods of optimization*, Wiley, 1987.

[7] IOFFE, A.D.: 'Necessary and sufficient conditions for a local minimum, 1: A reduction theorem and first-order conditions', *SIAM J. Control Optim.* **17** (1979), 245–250.

[8] IOFFE, A.D.: 'Necessary and sufficient conditions for a local minimum, 2: Conditions of Levitin–Miljutin–Osmolovskii type', *SIAM J. Control Optim.* **17** (1979), 251–265.

[9] IOFFE, A.D.: 'Necessary and sufficient conditions for a local minimum, 3: second order conditions and augmented duality', *SIAM J. Control Optim.* **17** (1979), 266–288.

[10] JEYAKUMAR, V.: 'Composite nonsmooth programming with Gâteaux differentiability', *SIAM J. Optim.* **1** (1991), 30–41.

[11] JEYAKUMAR, V., AND YANG, X.Q.: 'Convex composite multi-objective nonsmooth programming', *Math. Program.* **59** (1993), 325–343.

[12] JEYAKUMAR, V., AND YANG, X.Q.: 'Convex composite minimization with $C^{1,1}$ functions', *J. Optim. Th. Appl.* **86** (1995), 631–648.

[13] PENOT, J.P.: 'Optimality conditions in mathematical programming and composite optimization', *Math. Program.* **67** (1994), 225–245.

[14] ROCKAFELLAR, R.T.: *Convex analysis*, Princeton Univ. Press, 1970.

[15] ROCKAFELLAR, R.T.: 'First- and second-order epi-differentiability in nonlinear programming', *Trans. Amer. Math. Soc.* **307** (1988), 75–108.

[16] ROCKAFELLAR, R.T.: 'Second-order optimality conditions in nonlinear programming obtained by way of epi-derivatives', *Math. Oper. Res.* **14** (1989), 462–484.

[17] STUDNIARSKI, M., AND JEYAKUMAR, V.: 'A generalized mean-value theorem and optimality conditions in composite nonsmooth minimization', *Nonlinear Anal. Th. Methods Appl.* **24** (1995), 883–894.

[18] WHITE, D.J.: 'Multi-objective programming and penalty functions', *J. Optim. Th. Appl.* **43** (1984), 583–599.

[19] YANG, X.Q.: 'Generalized second-order directional derivatives and optimality conditions', *PhD Thesis Univ. New South Wales, Australia* (1994).

[20] YANG, X.Q.: 'Second-order global optimality conditions for convex composite optimization', *Math. Program.* **81** (1998), 327–347.

[21] YANG, X.Q., AND JEYAKUMAR, V.: 'First and second-order optimality conditions for convex composite multi-objective optimization', *J. Optim. Th. Appl.* **95** (1997), 209–224.

V. Jeyakumar
School of Math., Univ. New South Wales
Sydney 2052, Australia
E-mail address: jeya@maths.unsw.edu.au

MSC2000: 46A20, 90C30, 52A01

Key words and phrases: nonsmooth analysis, convex composite programming, optimality conditions, nonsmooth optimization, vector optimization.

COMPUTATIONAL COMPLEXITY THEORY

Many problems that arise in operations research and related fields are *combinatorial* in nature: problems where we seek the optimum from a very large but finite number of solutions. Sometimes such problems can be solved quickly and efficiently, but often the best solution procedures available are slow and tedious. It therefore becomes important to assess how well a proposed procedure will perform.

The theory of *computational complexity* addresses this issue. Complexity theory is a comparatively young field, with seminal papers dating from 1971–1972 ([1], [5]). Today, it is a wide field encompassing many subfields. For a formal treatment, see [6]. As we shall see, the theory partitions all realistic problems into two groups: the 'easy' and the 'hard' to solve, depending on how complex (hence how fast or slow) the computational procedure for that problem is. The theory defines still other classes, but all but the most artificial mathematical constructs fall into these two. Each of them can be further subdivided in various ways, but these refinements are beyond our scope. It should be noted that we have not here used the accepted terminology, which is introduced below.

Definitions. A *problem* is a well-defined question to which an unambiguous answer exists. *Solving the problem* means answering the question. The problem is stated in terms of several *parameters*, numerical quantities which are left unspecified but are understood to be predetermined. They make up the data of the problem. An *instance* of a problem gives specified values to each parameter. A *combinatorial optimization problem*, whether *maximization* or *minimization*, has for each instance a finite number of candidates from which the answer, or optimal solution, is selected. The choice is based on a real-valued objective function which assigns a value to each candidate solution. A *decision problem* or *recognition problem* has only two possible answers, YES or NO.

EXAMPLE 1 For example, consider the problem of solving a given system of linear equations. Stated as a question, it becomes: 'what is the solution to $\mathbf{Ax} = \mathbf{b}$?' with parameters m, n, $a_{i,j}$, b_i, x_j where $i = 1,\ldots,m$, $j = 1,\ldots,n$. An instance might be: 'What is the solution to $7x_1 - 3x_2 = 16$ and $2x_1 + 5x_2 = 9$?' with parameters $m = 2$, $n = 2$ etc.

This is neither an optimization problem nor a decision problem. An example of an optimization problem is a linear program, which asks: 'what is the greatest value of $\mathbf{cx}$ subject to $\mathbf{Ax} \leq \mathbf{b}$?' To make this a combinatorial optimization problem, we might make the variable $\mathbf{x}$ bounded and integer-valued so that the number of candidate solutions is finite. A decision problem is: 'does there exist a solution to the linear program with $\mathbf{cx} \geq k$?' □

To develop *complexity theory*, it is convenient to state all problems as decision problems. An optimization (say, maximization) problem can always be replaced by a sequence of problems of determining the existence of solutions with values exceeding $k_1, k_2, \ldots$. An *algorithm* is a step-by-step procedure which provides a solution to a given problem; that is, to all instances of the problem. We are interested in how fast an algorithm is. We now introduce a measure of algorithmic speed: the time complexity function.

The Nature of the Time Complexity Function. Complexity theory does not measure the speed of an algorithm directly; that would depend on the speed of the computer being used and other extraneous factors. Rather, it considers the rate of growth of the solution time as a function of the instance size. Since different instances of the same size may require dramatically different solution times, we use the 'worst case' or longest time that any instance of that size requires. This maximal time needed to solve a problem instance, as a function of its size, is called the *time complexity function* (TCF) or simply the *complexity of the algorithm*. When we speak of the complexity of the problem, we mean the complexity of the most efficient algorithm (known or unknown) that solves it.

We need to clarify what we mean by the 'time required' and the 'size of an instance'. First, note that we always think of solving problems using a computer. Thus, an algorithm is a piece of computer code. Similarly, the *size of a problem instance* is technically the number of characters needed to specify the data, or the length of the input needed by the program. For a decision problem, an algorithm receives as input any string of characters, and produces as output either YES or NO or 'this string is not a problem instance'. An algorithm *solves the instance or string in time m* if it requires m basic operations to reach one of the three conclusions and stop.

In order to avoid detailed consideration of the exact input length (are binary or alphanumeric characters used? what encoding scheme is used?), as well as avoiding precise measurement of solution times, the theory requires no more than *orders of magnitude* of these measurements. Recall, we are only concerned with the rate of increase of solution time as instances grow. For example, we may ask how much longer it takes if we double the instance size. As long as we enter data consistently, an instance that is twice as big as another under one data entry scheme remains twice as big under another. (For a rigorous proof and other technical issues, see [4]). Indeed, it is customary to use as a surrogate for instance size, any number that is roughly proportional to the true value. We shall use the symbol n, $n = 1, 2, \ldots$, to represent the size of a problem instance. In summary, for a decision problem Π:

DEFINITION 2 The *time complexity function* (TCF) of algorithm A is:

$$T_A(n) = \begin{cases} \text{maximal time for } A \\ \text{to solve any string of length } n. \end{cases}$$

□

In what follows, the big oh notation ($\mathcal{O}$) introduced in [3] will be used when expressing the time complexity function. We say that, for two real-valued functions f and g, '$f(n)$ is $\mathcal{O}(g(n))$', or '$f(n)$ is of the same order as $g(n)$', if $|f(n)| \leq k \cdot |g(n)|$ for all $n \geq 0$ and some $k > 0$.

Polynomial versus Exponential Algorithms. An *efficient, polynomially bounded, polynomial time algorithm*, or simply a *polynomial algorithm*, is one which solves a problem instance in time bounded by a power of the instance size. Formally:

DEFINITION 3 An algorithm A is *polynomial time* if there exists a polynomial p such that

$$T_A(n) \leq p(n), \quad \forall n \in \mathbf{Z}^+ \equiv \{1, 2, \ldots\}. \quad \square$$

More specifically, an algorithm is *polynomial of degree c*, or *has complexity* $\mathcal{O}(n^c)$, or *runs in* $\mathcal{O}(n^c)$ *time* if, for some $k > 0$, the algorithm never takes longer than kn^c (the TCF) to solve an instance of size n.

DEFINITION 4 The collection P comprises all problems for which a polynomial time algorithm exists. □

Problems which belong to P are the ones we referred to earlier as 'easy'. All other algorithms are called *exponential time* or just *exponential*, and problems for which nothing quicker exists are 'hard'. Although not all algorithms in this class have TCF's that are technically exponential functions, we may think of a typical one as running in $\mathcal{O}(c^{p(n)})$ for some polynomial $p(n)$. Other examples of exponential rates of growth are n^n and $n!$.

The terms 'hard' and 'easy' are somewhat misleading, even though exponential TCFs clearly lead to far more rapid growth in solution times. Suppose an 'easy' problem has an algorithm with running time bounded by, say kn^5. Such a TCF may not be exponential, but it may well be considered pretty rapidly growing. Furthermore, some algorithms take a long time to solve even small problems (large k), and hence are unsatisfactory in practice even if the time grows slowly. On the other hand, an algorithm for which the TCF is exponential is not always useless in practice. Recall, the concept of the TCF is a worst case estimate, so complexity is only an upper bound on the amount of time required by an algorithm. This is a conservative measure and usually useful, but it is too pessimistic for some popular algorithms. The simplex algorithm for linear programming, for example, has a TCF that is $\mathcal{O}(2^n)$, but it has been shown that for the average case the complexity is only $\mathcal{O}(n)$. Thus, the algorithm is actually very fast for most problems encountered.

Despite these caveats, exponential algorithms generally have running times that tend to increase at an exponential rate and often seem to 'explode' when a certain problem size is exceeded. Polynomial time algorithms usually turn out to be of low degree ($\mathcal{O}(n^3)$ or better), run pretty efficiently, and are considered desirable.

Reducibility. A problem can be placed in P as soon as a polynomial time algorithm is found for it. Sometimes, rather than finding such an algorithm, we may place it in P by showing that it is 'equivalent' to another problem which is already known to be in P. We explain what we mean by equivalence between problems with the following definitions.

DEFINITION 5 A problem Π' is *reducible* or *transformable* to a problem Π ($\Pi' \propto \Pi$) if, for any instance I' of Π', an instance I of Π can be constructed in polynomially bounded time, such that the solution to I is sufficient to find the solution to I' in polynomial time. □

We call the construction of the I that corresponds to I' a *polynomial transformation* of I' into I.

DEFINITION 6 Two problems are *equivalent* if each is reducible (or simply *reduces*) to the other. □

Since reduction, and hence equivalence, are clearly transitive properties, we can define *equivalence classes of problems*, where all problems in the same equivalence class are reducible (or equivalent) to each other. Consider polynomial problems. Clearly, for two equivalent problems, if one is known to be polynomial, the other must be, too. Also, if two problems are each known to be polynomial, they are equivalent. This is because any problem $\Pi' \in P$ is reducible to any other problem $\Pi \in P$, or indeed to any $\Pi \notin P$, in the following trivial sense. For any instance I' of Π', we can pick any instance of Π, ignore its solution, and find the solution to I' directly. We conclude that P is an equivalence class.

We state a third simple result for polynomial problems as a theorem.

THEOREM 7 If $\Pi \in P$, then $\Pi' \propto \Pi \Rightarrow \Pi' \in P$. □

Given any instance I' of Π', one can find an instance I of Π by applying a polynomial time transformation to I'. Since $\Pi \in P$, there is a polynomial time algorithm that solves I. Hence, using

the transformation followed by the algorithm, I' can be solved in polynomial time.

Classification of Hard Problems. In practice, we do not usually use reduction to show a problem is polynomial. We are more likely to start optimistically looking for an *efficient algorithm* directly, which may be easier than seeking another problem known to be polynomial, for which we can find an appropriate transformation. But suppose we cannot find either an efficient algorithm or a suitable transformation. We begin to suspect that our problem is not 'easy' (i.e., is not a member of P). How can we establish that it is in fact 'hard'? We start by defining a larger class of problems, which includes P and also all the difficult problems we ever encounter. To describe it, consider any combinatorial decision problem. For a typical instance, there may be a very large number of possible solutions which may have to be searched. Picture a candidate solution as a set of values assigned to the variables $\mathbf{x} = (x_1, \ldots, x_n)$. The question may be 'for a given vector $\mathbf{c}$ is there a solution $\mathbf{x}$ such that $\mathbf{cx} \leq B$?' and the algorithm may search the solutions until it finds one satisfying the inequality (whereupon it stops with the answer YES) or exhausts all solutions (and stops at NO).

This may well be a big job. But suppose we are told 'the answer is YES, and here is a solution $\mathbf{x}$ that satisfies the inequality'. We feel we must at least *verify* this, but that is trivial. Intuitively, even for the hardest problems, the amount of work to check that a given candidate solution confirms the answer YES should be small, even for very large instances. We will now define our 'hard' problems as those which, though hard to solve, are easy to verify, where as usual 'easy' means taking a time which grows only polynomially with instance size. To formalize this, let:

$$V_A(n) = \begin{cases} \text{maximal time for } A \\ \text{to verify that a given solution} \\ \text{establishes the answer YES} \\ \text{for any instance of length } n. \end{cases}$$

DEFINITION 8 An algorithm $\tilde{A}$ is *nondeterministic polynomial time* if there exists a polynomial p such that for every input of length n with answer YES, $V_{\tilde{A}}(n) \leq p(n)$. □

DEFINITION 9 The collection NP comprises all problems for which a nondeterministic polynomial algorithm exists. □

It may be noted that a problem in NP is solvable by searching a decision tree of polynomially bounded depth, since verifying a solution is equivalent to tracing one path through the tree. From this, it is easy to see that $P \subseteq NP$. Strangely, complexity theorists have been unable to show that $P \subset NP$; it remains possible that all the problems in NP could actually be solved by polynomial algorithms, so that $P = NP$. However, since so many brilliant researchers have worked on so many difficult problems in NP for so many years without success, this is regarded as being very unlikely. Assuming $P \neq NP$, as we shall hereafter, it can be shown that the problems in NP include an infinite number of equivalence classes, which can be ranked in order of increasing difficulty; where an equivalence class $\mathcal{C}$ is 'more difficult' than another class $\mathcal{C}'$ if, for every problem $\Pi \in \mathcal{C}$ and every $\Pi' \in \mathcal{C}'$, $\Pi' \propto \Pi$ but $\Pi \not\propto \Pi'$. There also exist problems that cannot be compared: neither $\Pi \propto \Pi'$ nor $\Pi' \propto \Pi$.

Fortunately, however, all problems that arise naturally have always been found to lie in one of two equivalence classes: the 'easy' problems in P, and the 'hard' ones, which we now define.

The class of *NP-hard problems* (NPH) is a collection of problems with the property that every problem in NP can be reduced to the problems in this class. More formally,

DEFINITION 10

$$NPH = \{\Pi : \forall \Pi' \in NP : \Pi' \propto \Pi\}.$$

□

Thus each problem in NPH is at least as hard as any problem in NP. We know that some problems in NPH are themselves in NP, though some are not. Those that are include the toughest problems in NP, and form the class of *NP-complete problems* (NPC). That is,

DEFINITION 11

$$NPC = \left\{ \Pi: \begin{array}{c} (\Pi \in NP) \\ \text{and} \\ (\forall \Pi' \in NP:\ \Pi' \propto \Pi) \end{array} \right\}.$$

□

The problems in *NPC* form an equivalence class. This is so because all problems in *NP* reduce to them, hence, since they are all in *NP*, they reduce to each other. The class *NPC* includes the most difficult problems in *NP*. As we mentioned earlier, by a surprising but happy chance, all the problems we ever encounter outside the most abstract mathematics turn out to belong to either *P* or *NPC*.

Using Reduction to Establish Complexity. When tackling a new problem Π, we naturally wonder whether it belongs to *P* or *NPC*. As we said above, to show that the problem belongs to *P*, we usually try to find a polynomial time algorithm, though we could seek to reduce it to a problem known to be polynomial. If we are unable to show that the problem is in *P*, the next step generally is to attempt to show that it lies in *NPC*; if we can do so, we are justified in not developing an efficient algorithm. To do this, clearly no direct algorithmic development is possible, and only a reduction argument will do. This is based on the following theorem, which should be clear enough to require no proof.

THEOREM 12 $\forall \Pi, \Pi' \in NP$: $(\Pi' \in NPC)$ and $(\Pi' \propto \Pi)$ imply $\Pi \in NPC$ □

Thus, we need to find a problem $\Pi' \in NPC$ and show $\Pi' \propto \Pi$, thereby demonstrating that Π is at least as hard as any problem in *NPC*. To facilitate this, we need a list of problems known to be in *NPC*. Several hundred are listed in [2] in a dozen categories such as graph theory, mathematical programming, sequencing and scheduling, number theory, etc., and more are being added all the time. Even given an ample selection, a good deal of tenacity and ingenuity are needed to pick one with appropriate similarities to ours and to fill in the precise details of the transformation.

Of course, to build up the membership in *NPC* using Theorem 12, we need other problems that have already been show to belong to that class. To begin this process, at least one problem needs to be in *NPC*. It was S.A. Cook [1] who showed that the satisfiability problem is *NP*-complete, using direct arguments that did not involve reduction. This very important result is called *Cook's theorem*. For a proof, see [2].

As a simple illustration of reduction, we show that the traveling salesperson decision problem (TSP) is in *NPC*. To do so, we first select a closely related problem, the hamiltonian circuit problem (HCP), which we assume has already been shown to be *NP*-complete. We then find a reduction of HCP to TSP. The problems are defined as follows.

DEFINITION 13 (*TSP, traveling salesperson problem*).

Instance:

- a positive integer n;
- a finite set $C = \{c_1, \ldots, c_n\}$ of 'cities';
- 'distances', $d_{ij} \in \mathbf{Z}^+$. $\forall i, j:\ c_i, c_j \in C$;
- a bound $B \in \mathbf{Z}^+$.

Question: Does there exist a *tour* (i.e., a closed path that visits every city exactly once), of length no greater than B? □

DEFINITION 14 (*HCP, Hamiltonian circuit problem*).

Instance: A graph $G = (V, E)$, where V is the set of m vertices, and E the set of edges.

Question: Does G contain a *Hamiltonian circuit*, i.e., a tour that traverses all vertices exactly once? □

EXAMPLE 15 To show: HCP $\propto$ TSP.

In TSP, we have a complete graph and seek the shortest tour, whereas in HCP, given an arbitrary graph we require any tour. Thus, given the challenge of showing that the traveling salesperson problem (or the decision version of it) is *NP*-complete, we have found a similar problem whose membership in *NPC* is already established. We may still be unable to find a polynomial transformation from HCP, in which case another problem must be sought. A transformation of Π' to Π is a way of computing each parameter of Π in terms of the parameters of Π'. In this case, the reduction is relatively simple. The parameters of HCP are:

- m = cardinality V;

- $E = \left\{(i,j): \begin{array}{c} G \text{ contains an arc} \\ \text{between vertices } i,j \end{array}\right\}$.

The parameters of TSP are computed as follows:

- $n = m$;
- $d_{ij} = \begin{cases} 1 & (i,j) \in E, \\ N & \text{otherwise}; \end{cases}$
- $B = m$.

Here, N can be any number larger than 1; say, 2. Clearly, the shortest possible tour in TSP has length m, and this only occurs when arcs in E are used exclusively; that is, when a tour in HCP exists.

To complete the reduction, we need to show that the transformation can be performed in polynomial time. For that, given a pair of nodes in TSP, we need to check if an arc exists in HC, and this requires time

$$\mathcal{O}\left[\frac{m(m-1)}{2}\right] = \mathcal{O}(m^2).$$

□

See also: **Complexity theory: Quadratic programming**; **Complexity of degeneracy**; **Complexity classes in optimization**; **Information-based complexity and information-based optimization**; **Fractional combinatorial optimization**; **Complexity of gradients, Jacobians, and Hessians**; **Complexity theory**; **Mixed integer nonlinear programming**; **Parallel computing: Complexity classes**; **Kolmogorov complexity**.

References

[1] COOK, S.A.: 'The complexity of theorem proving procedures': *Proc. 3rd Annual ACM Symposium on Theory of Computing*, ACM, 1971, pp. 151–158.

[2] GAREY, M.R., AND JOHNSON, D.S.: *Computers and intractability*, Freeman, 1979.

[3] HARDY, G.H., AND WRIGHT, E.M.: *An introduction to the theory of numbers*, Clarendon Press, 1979.

[4] HOPCROFT, J.E., AND ULLMAN, J.D.: *Introduction to automata theory, languages, and computation*, Addison-Wesley, 1979.

[5] KARP, R.M.: 'Reducibility among combinatorial problems', in R.E. MILLER AND J.W. THATCHER (eds.): *Complexity of Computer Computations*, Plenum, 1972, pp. 85–103.

[6] PAPADIMITRIOU, C.H.: *Computational complexity*, Addison-Wesley, 1994.

[7] PINEDO, M.: *Scheduling: Theory, algorithms and systems*, Prentice-Hall, 1995.

Hamilton Emmons
Dept. OR and Operations Management
Case Western Reserve Univ.
10900 Euclid Avenue, Cleveland, Ohio 44106, USA
E-mail address: hxe@po.cwru.edu

Sanatan Rai
Dept. OR and Operations Management
Case Western Reserve Univ.
10900 Euclid Avenue, Cleveland, Ohio 44106, USA
E-mail address: sxr57@po.cwru.edu

MSC2000: 90C60

Key words and phrases: computational complexity, complexity theory, combinatorial optimization, decision problem, recognition problem, time complexity function, efficient algorithm, polynomial algorithm, exponential algorithm, reducibility, nondeterministic polynomial algorithm, *NP*-hard problem, *NP*-complete problem.

CONCAVE PROGRAMMING

Concave programming constitutes one of the most fundamental and intensely-studied problem classes in deterministic nonconvex optimization. There are at least three reasons for this. First, many of the mathematical properties of concave programming are intriguingly attractive. Some are even identical to properties of linear programming. Second, concave programming has a remarkably broad range of direct and indirect applications. Third, the algorithmic ideas used in concave programming have played and continue to play an active and often fundamental role in the development of solution procedures for other types of nonconvex programming problems.

The *concave programming*, or *concave minimization*, problem (CMP) can be written

$$\begin{cases} \text{globmin} & f(x), \\ \text{s.t.} & x \in D, \end{cases}$$

where D is a nonempty, closed convex set in $\mathbf{R}^n$ and f is a real-valued, concave function defined on some open convex set A in $\mathbf{R}^n$ that contains D. The goal in CMP is to find the global minimum value that f achieves over D, and, if this value is not equal to $-\infty$, to find, if it exists, at least one point in D that achieves this value. In many applications, D is compact and A equals all of $\mathbf{R}^n$. CMP invariably contains many points in D that

are local, but not global, minimizers of f over D. For this reason, CMP is an example of a *(multi-extremal) global optimization* problem [7].

The application of standard algorithms designed for solving constrained convex programming problems generally will fail to solve CMP. Even instances of CMP with relatively simple components can apparently present very significant solution challenges. For example, B. Kalantari [8] has shown that in problems involving the minimization of concave quadratic functions over rectangles in $\mathbf{R}^n$, an exponential number of extreme point local minima can exist. Additionally, P.M. Pardalos and G. Schnitger [13] have shown that minimizing a concave quadratic function over a hypercube is an *NP*-hard problem.

Although CMP is more difficult to solve than a convex programming problem, it possesses some highly interesting, special mathematical properties. A number of these properties have been exploited by researchers to create successful algorithms for solving the problem.

For instance, if D contains at least one extreme point, and CMP has at least one global optimal solution, then there must exist a global optimal solution which is an extreme point of D [14]. This is perhaps the most important and striking property of concave minimization problems. As a result of this property, just as in linear programming, if CMP has a global optimal solution, then one can confine the search for such a solution to the set of extreme points of D, provided that this set is nonempty. This property holds, as in linear programming, even when D is unbounded. A number of algorithms for CMP are based upon this property.

Another highly important property for CMP is that if D is a compact set, then CMP must have a global optimal solution which is an extreme point of D. This is perhaps the most widely-known theoretical result in concave minimization [1]. Like the property stated in the previous paragraph, it forms the basis for a number of important concave minimization algorithms.

For cases where D is a polyhedron, possibly unbounded, that contains at least one extreme point, it has been shown that either CMP has a global optimal solution which is an extreme point of D, or CMP must be unbounded and f must be unbounded from below over some extreme direction of D. Notice that the same property, remarkably, holds in the case of linear programming. This property is used by a large number of the algorithms designed to solve CMP when D is a nonempty polyhedron.

CMP displays a remarkable diversity of applications. Each application is either direct or indirect. By direct, we mean that the original model formulation takes the form of CMP immediately or, if not, with only relatively simple algebraic manipulations. The indirect applications involve problems whose direct formulations do not take the form of CMP, but existing theory can be used to reformulate these problems in the form of CMP.

Some of the oldest and most diverse direct applications of CMP belong to a class of problems called *fixed charge* problems. In these problems, the objective function f is *separable*, i.e., it is of the form

$$f(x) = \sum_{i=1}^{n} f_i(x_i).$$

For each $i = 1, \ldots, n$, in these problems f_i is a concave function on $\{x_i \in \mathbf{R}\colon x_i \geq 0\}$ of the form

$$f_i(x_i) = \begin{cases} 0 & \text{if } x_i = 0, \\ c_i + w_i(x_i) & \text{if } x_i > 0, \end{cases}$$

where $c_i > 0$ is the fixed *setup cost* of undertaking activity i at a positive level, and $w_i(x_i)$ is a continuous concave function on $\{x_i \in \mathbf{R}\colon x_i > 0\}$ that represents the *variable cost* of undertaking the activity at level x_i.

When the functions $w_i(x_i)$, $i = 1, \ldots, n$, are linear, the classical *linear fixed charge* problem is obtained. Some of the oldest applications of concave minimization involve solving problems of this type. Among these are applications to transportation planning, site selection, production lot sizing and network design. For cases where at least one of the functions $w_i(x_i)$, $i = 1, \ldots, n$, is piecewise linear, several types of applications have been reported. Included among these are problems involving price breaks, such as bid evaluation problems, certain inventory planning problems and various plant location problems. When $w_i(x_i)$ is a general

concave function for some $i = 1, \ldots, n$, applications involving *economies of scale*, for instance, can be solved.

More recently, CMP has been directly applied to a class of problems called *multiplicative programming* problems. These are problems of the form CMP where

$$f(x) = \prod_{j=1}^{p} f_j(x) \tag{1}$$

for some set of $p \geq 2$ functions f_j, $j = 1, \ldots, p$, that are each nonnegative over D. Notice that if $f_k(\bar{x}) = 0$ for some $k \in \{1, \ldots, p\}$ and some $\bar{x} \in D$, then it is easy to see that the global minimum value for CMP is 0, and $\bar{x}$ is a global optimal solution. Therefore, it is generally assumed in multiplicative programming problems that each function f_j is positive over D. Let us make this assumption henceforth.

The objective function (1) of a multiplicative program is generally not a concave function. But, when each function f_j, $j = 1, \ldots, p$, is a concave function on $\mathbf{R}^n$, some simple transformations of (1) yield concave functions over D. For instance, if, for each $x \in D$, we define w_1 and w_2 by

$$w_1(x) = \ln f(x) = \sum_{j=1}^{p} \ln f_j(x),$$

$$w_2(x) = [f(x)]^{1/p},$$

respectively, then, whenever each f_j, $j = 1, \ldots, p$, is a concave function on $\mathbf{R}^n$, both w_1 and w_2 are concave functions on D [3], [10]. Thus, by using w_1 or w_2, multiplicative programming problems in which f_j, $j = 1, \ldots, p$, are concave functions can be easily transformed to concave minimization problems.

Various applications of multiplicative programming problems with concave or linear functions f_j, $j = 1, \ldots, p$, in (1) have arisen, especially since the 1960s. For example, the linear case has been applied to the problem of optimizing value functions for multiple objective programming problems subject to linear or nonlinear constraints. For $p = 2$, the linear case has been used to help solve the modular design problem, to design integrated circuit chips and to select bond portfolios. The concave case has been used to analyze and solve a number of problems in microeconomics.

Subject to occasional restrictions, large classes of integer programming problems can be converted by various means into the form of CMP and solved as concave minimization problems. As a result, these integer programming problems, indirectly, are applications of concave minimization. The transformation processes used to accomplish the conversion, however, can be rather involved. They may also increase the size of the original problem [4], and they may call for choosing values for parameters that are difficult to determine [5], [9].

In particular, by using a certain general transformation process, any feasible linear integer or quadratic integer programming problem over a polyhedron with nonnegative, bounded variables can be converted into the form of CMP and solved as a concave minimization problem. By using more customized conversion processes, linear zero-one programs, quadratic assignment problems, and other special integer programming problems can also be transformed to the form of CMP and solved as concave minimization problems. The specialized transformations generally take advantage of some aspect of the original integer programming problem that the general processes ignore.

There are many other indirect applications of CMP, including *d.c. optimization*, *indefinite quadratic programming*, and *bilinear programming*, for instance. For further details and additional direct and indirect applications, see [1], [2], [6], [7], [11], [12].

To solve CMP, a large number of algorithms have been developed. Many of these rely on one or more of the following four approaches.

In the *cutting plane approach*, a local minimum for f over D is found. Subsequently, a hyperplane is constructed and used to cut off all points of D whose objective function values are not less than that of the local minimum. This yields a new closed convex set $D^1 \subset D$. This process is then repeated with D^1 in the role of D. By iterating this process, the portion of D remaining to be explored is progressively reduced. Termination occurs when it can be shown that $f(y) \geq f(x^k)$ for all $y \in D^k$, where

D^k is the portion of D remaining at iteration k, and x^k is the local minimum found through iteration k with the smallest objective function value.

In a typical *outer approximation* approach, D is assumed to be compact. To initiate the approach, a simple bounded polyhedron P^1 containing D whose vertices can be enumerated is constructed. A vertex v^1 of P^1 of minimum objective function value among all of the vertices of P^1 is found. This gives a lower bound $f(v^1)$ for the optimal value of CMP. If $v^1 \in D$, v^1 is a global optimal solution for CMP and termination occurs. Otherwise, a new bounded polyhedron $P^2 \subset P^1$ is constructed that contains D, and its vertices are enumerated. With P^2 in the role of P^1, the process is repeated. By repeating this process, a sequence of telescoping bounded polyhedra containing D is obtained. Termination occurs in the first iteration k where the vertex v^k found that minimizes f over all of the vertices of P^k lies in D.

In *inner approximation* (*polyhedral annexation*) approaches for CMP, D is assumed to be a bounded polyhedron. Typically, at each major iteration of an inner approximation algorithm, a local minimum extreme point solution $\overline{x}$ for CMP is available. A sequence of expanding inner approximating compact polyhedra for $(D \cap G)$ is constructed via a series of subiterations, where $G = \{x \in \mathbf{R}^n : f(x) \geq f(\overline{x})\}$. During this process, either an improved local minimum extreme point $\overline{\overline{x}}$ is found, or, after k subiterations, the algorithm detects that $D \subseteq P^k$, where P^k is the current inner approximation of $(D \cap G)$. In the former case, $\overline{\overline{x}}$ replaces $\overline{x}$ and a new major iteration begins. In the latter case, since $P^k \subseteq (D \cap G)$, it follows that $D \subseteq G$, and the algorithm therefore terminates with the global optimal solution $\overline{x}$.

In the *branch and bound* approaches for CMP, D is repeatedly subdivided into finer and finer partitions. A lower bound for f over each partition element is calculated. The lowest of these lower bounds at any step k of the process gives a global lower bound LB_k for f over D. At any stage, typically some feasible solutions for CMP have been detected. A feasible solution $\overline{y}$ with the smallest f value among all feasible solutions detected through any point in the algorithm is always available. This solution is called the *incumbent solution*. When, at some step k, the inequality $LB_k \geq f(\overline{y})$ holds for the first time, the algorithm stops and returns the global optimal solution $\overline{y}$.

Details concerning these and other solution approaches can be found in [1], [2], [6], [7].

See also: **Minimum concave transportation problems; Bilevel linear programming: Complexity, equivalence to minmax, concave programs.**

References

[1] Benson, H.P.: 'Concave minimization: Theory, applications and algorithms', in R. Horst and P.M. Pardalos (eds.): *Handbook global optimization*, Kluwer Acad. Publ., 1995, pp. 43–148.

[2] Benson, H.P.: 'Deterministic algorithms for constrained concave minimization: A unified critical survey', *Naval Res. Logist.* **73** (1996), 765–795.

[3] Benson, H.P., and Boger, G.M.: 'Multiplicative programming problems: Analysis and efficient point search heuristic', *J. Optim. Th. Appl.* **94** (1997), 487–510.

[4] Garfinkel, R.S., and Nemhauser, G.L.: *Integer programming*, Wiley, 1972.

[5] Giannessi, F., and Niccolucci, F.: 'Connections between nonlinear and integer programming problems', *Symp. Mat. Inst. Naz. di Alta Mat.* **19** (1976), 161–176.

[6] Horst, R., Pardalos, P.M., and Thoai, N.V.: *Introduction to global optimization*, Kluwer Acad. Publ., 1995.

[7] Horst, R., and Tuy, H.: *Global optimization: Deterministic approaches*, second revised ed., Springer, 1993.

[8] Kalantari, B.: 'Quadratic functions with exponential number of local maxima', *Oper. Res. Lett.* **5** (1986), 47–49.

[9] Kalantari, B., and Rosen, J.B.: 'Penalty for zero-one integer equivalent problems', *Math. Program.* **24** (1982), 229–232.

[10] Konno, H., and Kuno, T.: 'Multiplicative programming problems', in R. Horst and P.M. Pardalos (eds.): *Handbook Global Optim.*, Kluwer Acad. Publ., 1995, pp. 369–405.

[11] Pardalos, P.M.: 'On the passage from local to global in optimization', in J.R. Birge and K.G. Murty (eds.): *Mathematical Programming: State of the Art*, Braun-Brumfield, 1994, pp. 220–247.

[12] Pardalos, P.M., and Rosen, J.B.: *Constrained global optimization: Algorithms and applications*, Springer, 1987.

[13] Pardalos, P.M., and Schnitger, G.: 'Checking local optimality in constrained quadratic programming is NP-hard', *Oper. Res. Lett.* **7** (1988), 33–35.

[14] ROCKAFELLAR, R.T.: *Convex analysis*, Princeton Univ. Press, 1970.

Harold P. Benson
Dept. Decision and Information Sci. Univ. Florida
Gainesville, Florida 32611-7169, USA
E-mail address: benson@dale.cba.ufl.edu

MSC2000: 90C25
Key words and phrases: concave programming, multi-extremal global optimization.

CONJUGATE-GRADIENT METHODS

Conjugate-gradient methods (CG methods) are used to solve large-dimensional problems that arise in computational linear algebra and computational nonlinear optimization. These two subjects share a broad common frontier, and one of the most easily traversed crossing points is via the following simple observation: the problem of solving a system of linear equations $\mathbf{Ax} = \mathbf{b}$ for the unknown vector $\mathbf{x} \in \mathbf{R}^n$, where $\mathbf{A}$ is a positive definite, symmetric matrix and $\mathbf{b}$ is a given vector, is mathematically equivalent to finding the minimizing point of the strictly convex quadratic function

$$q(\mathbf{x}) = -\mathbf{b}^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \mathbf{Ax}.$$

The *linear CG method* for solving the system of linear equations is able to capitalize on this equivalent optimization formulation. It was developed in the pioneering 1952 paper of M.R. Hestenes and E.L. Stiefel [11] who, in turn, cite antecedents in the contributions of several other authors (see [9]). The method fell out of favor with numerical analysts during the 1960s because it did not compete with direct methods, in particular, Gaussian elimination, but it continued to be widely used in real-world applications by specialists in other areas. Interest in CG as an iterative method, downplaying its finite-termination properties, revived in the 1970s when the solution of *large scale linear systems* was coming to the forefront of academic research.

The *nonlinear CG method* extends the linear CG approach to the problem of minimizing a smooth, nonlinear function $f(\mathbf{x})$, $\mathbf{x} \in \mathbf{R}^n$, where n can be large. It was developed in another landmark article published in 1964 by R. Fletcher and C. Reeves [8]. Optimization techniques for this class of problems, which are inherently iterative in nature, form a direction of descent at an approximation to the solution (the current iterate), and search along this direction to obtain a new iterate with an improved function value. The *Fletcher-Reeves algorithm* combined a search direction derived from the *Hestenes-Stiefel approach* with an efficient line search procedure along this direction vector adapted from the 1959 variable-metric breakthrough algorithm of W.C. Davidon [6]. The resulting CG algorithm was a marked enhancement of the classical steepest descent method of A.-L. Cauchy.

Like steepest descent, the nonlinear CG method is storage-efficient and requires only a few n-vectors of computer storage beyond that needed to specify the problem itself. During the three decades after its discovery, a large number of storage-efficient *nonlinear CG-related algorithms* were proposed. In particular, a structural connection between conjugate-gradient and variable-metric techniques for defining search direction vectors provided the springboard for effective new families of variable-storage/limited-memory algorithms and affine-reduced-Hessian algorithms that occupy a middle ground.

These three classes of algorithms, namely, linear CG, nonlinear CG and nonlinear CG-related, will be discussed in the respective Sections below.

Notation and Preliminaries. Lowercase boldface letters, e.g., $\mathbf{x}$, denote vectors, and uppercase boldface letters, e.g., $\mathbf{A}$, denote symmetric, positive definite matrices. The *residual* at $\mathbf{x}$ of the linear system $\mathbf{Ax} = \mathbf{b}$ is $\mathbf{r} = \mathbf{Ax} - \mathbf{b}$. It equals the gradient vector $\mathbf{g} = -\mathbf{b} + \mathbf{Ax}$ of the strictly convex, quadratic form

$$q(\mathbf{x}) = -\mathbf{b}^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \mathbf{Ax}$$

at $\mathbf{x}$. The gradient vector of q vanishes only at the unique solution $\mathbf{A}^{-1}\mathbf{b}$ of the linear system.

Linear CG Algorithms. A basic CG algorithm for solving the system of linear equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a positive definite, symmetric matrix, is as follows:

	(Initialization)
0	$\mathbf{x}_1$ = arbitrary;
	$\mathbf{r}_1$ = residual of linear system at $\mathbf{x}_1$;
	$\mathbf{d}_1 = -\mathbf{r}_1$;
	(Iteration i)
1	$\mathbf{x}_{i+1}$ = unique minimizing point of q on halfline through $\mathbf{x}_i$ along direction $\mathbf{d}_i$;
2	$\mathbf{r}_{i+1}$ = residual of linear system at $\mathbf{x}_{i+1}$;
3	$\beta_i = \|\mathbf{r}_{i+1}\|^2/\|\mathbf{r}_i\|^2$;
4	$\mathbf{d}_{i+1} = -\mathbf{r}_{i+1} + \beta_i \mathbf{d}_i$.

In the computational linear algebra setting, the matrix $\mathbf{A}$ is provided exogenously. The residual $\mathbf{r}_{i+1}$, at step 2, is computed as $\mathbf{A}\mathbf{x}_{i+1} - \mathbf{b}$ or else obtained by updating $\mathbf{r}_i$. The direction $\mathbf{d}_i$ is always a descent direction for q at $\mathbf{x}_i$. At step 1, the minimizing point is computed as follows:

$$\alpha_i = -\frac{\mathbf{r}_i^\top \mathbf{d}_i}{\mathbf{d}_i^\top \mathbf{A}\mathbf{d}_i}; \quad \mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i.$$

There are numerous variants on this basic algorithm that seek to enhance convergence through problem preconditioning (transformation of variables), to improve algorithm stability and to solve related computational linear algebra problems. A contextual overview and further references can be found in [2].

Here our focus is optimization. In this setting, the residual at $\mathbf{x}_{i+1}$ is given its alternative interpretation and representation as the gradient vector $\mathbf{g}_{i+1}$ of q at $\mathbf{x}_{i+1}$, and this gradient is assumed to be provided exogenously. The minimizing point at step 1 is computed, alternatively, as follows:

$$\alpha_i = -\frac{\mathbf{g}_i^\top (\overline{\mathbf{x}}_i - \mathbf{x}_i)}{\mathbf{d}_i^\top (\overline{\mathbf{g}}_i - \mathbf{g}_i)},$$

where $\overline{\mathbf{x}}_i$ is any point on the ray through $\mathbf{x}_i$ in the direction $\mathbf{d}_i$ and $\overline{\mathbf{g}}_i$ is its corresponding gradient vector; $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i$. The expression for α_i is derived from the previous linear systems version and the relation $\mathbf{A}(\overline{\mathbf{x}}_i - \mathbf{x}_i) = \overline{\mathbf{g}}_i - \mathbf{g}_i$.

We will call the resulting optimization algorithm the *CG-standard for minimizing q*. It will provide an important guideline for defining CG algorithms in the subsequent discussion.

Nonlinear CG Algorithms. A nonlinear CG algorithm is used to find a minimizing point of the nonlinear function $f(\mathbf{x})$, $\mathbf{x} \in \mathbf{R}^n$, when n is large and/or computer storage is at a premium. A basic algorithm can be stated as follows:

	(Initialization)
0	$\mathbf{x}_1$ = arbitrary;
	$\mathbf{g}_1$ = gradient of f at $\mathbf{x}_1$;
	$\mathbf{d}_1 = -\mathbf{g}_1$;
	(Iteration i)
1	$\mathbf{x}_{i+1}$ = an improved iterate on halfline through $\mathbf{x}_i$ along direction $\mathbf{d}_i$;
2	$\mathbf{g}_{i+1}$ = gradient of f at $\mathbf{x}_{i+1}$;
3	$\beta_i = \|\mathbf{g}_{i+1}\|^2/\|\mathbf{g}_i\|^2$;
4	$\mathbf{d}_{i+1} = -\mathbf{g}_{i+1} + \beta_i \mathbf{d}_i$.

Note that this algorithm is closely patterned after the *CG-standard*. The improved iterate at step 1 is obtained by a *line search* procedure, which is normally based on quadratic or cubic polynomial fitting (suitably safeguarded). When $f = q$, such a line search procedure can immediately locate the minimizing point along the line of search, once it has gathered the requisite information to make an exact fit. In other words, when applied to the minimization of q, the foregoing nonlinear CG algorithm is able to replicate the CG-standard precisely. This property characterizes a nonlinear CG algorithm.

Considerable research has gone into alternative expressions for the quantity β_i. The four leading contenders are the *Fletcher–Reeves* (FR), *Hestenes–Stiefel* (HS), *Polyak–Polak–Ribière* (PPR) and *Dai–Yuan* (DY) choices (see [8], [11], [22], [20], [5]). These define β_i as follows:

$$\text{FR}: \quad \beta_i = \frac{\mathbf{g}_{i+1}^\top \mathbf{g}_{i+1}}{\mathbf{g}_i^\top \mathbf{g}_i}; \tag{1}$$

$$\text{HS}: \quad \beta_i = \frac{\mathbf{g}_{i+1}^\top \mathbf{y}_i}{\mathbf{d}_i^\top \mathbf{y}_i};$$

$$\text{PPR}: \quad \beta_i = \frac{\mathbf{g}_{i+1}^\top \mathbf{y}_i}{\mathbf{g}_i^\top \mathbf{g}_i};$$

$$\text{DY}: \quad \beta_i = \frac{\mathbf{g}_{i+1}^\top \mathbf{g}_{i+1}}{\mathbf{d}_i^\top \mathbf{y}_i}, \tag{2}$$

where $\mathbf{y}_i = \mathbf{g}_{i+1} - \mathbf{g}_i$ is the gradient change that corresponds to the step $\mathbf{s}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$.

When line searches are exact and the function is quadratic, the following relations hold:

$$\mathbf{g}_{i+1}^\top \mathbf{g}_{i+1} = \mathbf{g}_{i+1}^\top \mathbf{y}_i, \quad \mathbf{g}_i^\top \mathbf{g}_i = \mathbf{d}_i^\top \mathbf{y}_i. \tag{3}$$

Thus, the values of the scalar β_i are identical for all four choices, and each of the associated algorithms becomes the CG-standard. In general, however, they are applied to nonquadratics and use inex-

act line searches, resulting in four distinct, nonlinear CG algorithms that can exhibit behavior very different from one another.

The following generalization yields a *two-parameter family*:

$$\beta_i = \frac{\lambda_i(\mathbf{g}_{i+1}^\top \mathbf{g}_{i+1}) + (1-\lambda_i)(\mathbf{g}_{i+1}^\top \mathbf{y}_i)}{\mu_i(\mathbf{g}_i^\top \mathbf{g}_i) + (1-\mu_i)(\mathbf{d}_i^\top \mathbf{y}_i)}, \quad (4)$$

where $\lambda_i \in [0,1]$ and $\mu_i \in [0,1]$. For any choice of λ_i and μ_i in these ranges, the associated algorithm reduces to the CG-standard when f is quadratic and line searches are exact, which follows from (3). If the scalars λ_i and μ_i take only their extreme values, 0 or 1, then one obtains four possible combinations corresponding to (1)–(2). The above two-parameter family of nonlinear CG algorithms, which subsumes FR, HS, PPR and DY, and its subfamilies (defined, for example, by taking $\lambda_i \equiv 1$) are currently a topic of active research.

When the line search is sufficiently accurate, a nonlinear CG algorithm always produces a direction of descent at step 4. Suitable inexact line search termination conditions, in conjunction with different choices of β_i, have been extensively studied, both theoretically and computationally. A good overview of the theory and convergence analysis of nonlinear CG algorithms can be found in [19]. The nonlinear CG algorithm based on the PPR choice for β_i ([20], [22]) and a suitable restarting strategy ([23]) has emerged as the most efficient in practice. However, it is well known that no single nonlinear CG algorithm works well all the time. There is enormous variability in performance on different problems or even within different regions of the same problem.

Nonlinear CG-Related Algorithms. We informally characterize a *nonlinear CG-related algorithm* as follows:

- its computer storage requirements are 'similar' to those of an implemented nonlinear CG algorithm, for example, [23];
- its path traverses the iterates of the CG-standard when the function is a strictly convex quadratic, line searches are exact, and the same initialization is used.

In other words, it may use a few more n-vectors of computer storage than, say, the PPR nonlinear CG algorithm, and it is permitted to generate additional intermediate iterates and form search vectors in novel ways. A nonlinear CG-related algorithm does not have to imitate the 'structure' of the basic nonlinear CG algorithm of the previous Section. But the above requirement that its path must cover the iterates of the CG-standard of the first Section implies the following: if the candidate algorithm does not exhibit finite termination when applied to a quadratic q then it is not a CG-related algorithm.

Let us now briefly categorize the main lines of development.

Classical Alternatives to CG. These seek to enhance or accelerate the steepest descent algorithm of Cauchy more directly, without explicitly introducing notions of conjugacy. The year of publication of [8] was indeed a banner year for such developments. Two particularly noteworthy contributions, which coincidentally also appeared in 1964, were the *parallel-tangents* or *PARTAN* algorithm of B.V. Shah, R.J. Buehler and O. Kempthorne [24] and the *heavy ball algorithm* of B.T. Polyak [21]. For a modern description of the former, and its subsequently discovered CG-related properties, see [14]. The basic iteration of the latter algorithm is as follows:

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \mathbf{g}_i + \beta(\mathbf{x}_i - \mathbf{x}_{i-1}),$$

where α is a constant positive stepsize and β is a scalar with $0 < \beta < 1$. Although, strictly speaking, it is not CG-related in this form, the algorithm has CG-like rate of convergence properties on a quadratic under optimal choices of the algorithm parameters α and β (see [3]). The algorithms in [24] and [21] both used very simple steplength techniques to move from one iterate to the next, in contrast to the nonlinear CG implementation of [8]. The introduction of the line-search technique of Davidon [6] into either of these 1964 algorithms, as in [8], could have propelled them much more into the limelight at the time. More recently, the important contribution of Yu.E. Nesterov [17] on global rate of convergence is based on an algorithm

akin to PARTAN [24]. However, in general, classical alternatives to CG remain on the sidelines.

Nonlinear CG Variants. These are premised on retaining the algorithmic structure, and the conjugacy properties on quadratics, of the basic nonlinear CG algorithm of the above Section when the initial direction is not along the negative gradient and/or line searches are inexact. For instance, the *three-term-recurrence algorithm* (TTR) is able to simultaneously relax both CG-standard requirements. The overall iteration follows the basic algorithm of the previous Section, but with the computation of the search direction at steps 3 and 4 replaced as follows:

$$\beta_i = \frac{\mathbf{y}_i^\top \mathbf{y}_i}{\mathbf{y}_i^\top \mathbf{d}_i}, \quad \gamma_i = \frac{\mathbf{y}_{i-1}^\top \mathbf{y}_i}{\mathbf{y}_{i-1}^\top \mathbf{d}_{i-1}},$$

and

$$\mathbf{d}_{i+1} = -\mathbf{y}_i + \beta_i \mathbf{d}_i + \gamma_i \mathbf{d}_{i-1}.$$

Conjugacy of search directions is retained when the initial search direction is chosen to be an arbitrary direction of descent. If this initial direction is along the negative gradient and line searches are exact, then the TTR generates the same search directions and iterates as the CG-standard on a positive definite quadratic. A drawback of the TTR algorithm is that it does not guarantee a descent direction on more general functions even if line searches are exact. But, in practice, the direction almost always satisfies the condition $\mathbf{g}_{i+1}^\top \mathbf{d}_{i+1} < 0$.

For references to other nonlinear CG variants, see [10] and the survey articles in [1]. Despite theoretical advantages on quadratics, algorithms in this category, in practice, have not proved to be significantly superior to the PPR nonlinear CG algorithm of the above Section.

Variable-Storage/Limited-Memory Algorithms. These are premised on a key structural relationship between the nonlinear CG algorithm and the BFGS variable-metric algorithm, and its properties on quadratics, see [15], [4] and [12]. The most effective CG-related algorithm, to date, is the L-BFGS algorithm of J. Nocedal [18]. This is described in more detail in **Unconstrained nonlinear optimization: Newton–Cauchy framework** and is not repeated here. The algorithm has the property that it produces a descent direction under weak termination conditions on the line search. It has proved to be an efficient and versatile algorithm (it can exploit additional computer storage when available) that generally outperforms the PPR algorithm in practice.

For an overview of other variable-storage algorithms that draw on the *BFGS-CG relationship*, see the survey articles in [1].

Affine-Reduced-Hessian Algorithms. These make estimates of curvature, i.e., approximations to the Hessian or its inverse, in an affine subspace usually of low dimension and defined by the most recent gradient and one or more prior steps and/or gradients. For algorithms of this type and their CG-related properties, see [7], [16], [13] and references cited therein.

The foregoing principle, on which such algorithms are premised, has the conceptual advantage that it provides a *true continuum* between the nonlinear CG and full-storage variable-metric (and Newton) algorithms. But, practical affine-reduced–Hessian implementations are not yet widespread.

Conclusion. CG algorithms are among the simplest and most elegant algorithms of computational nonlinear optimization. They can be surprisingly effective in practice, and thus will always have an honored place in the repertoire. Nevertheless, the subject still lacks a comprehensive underlying theory, and many interesting algorithmic issues remain to be explored.

Some references cited in the present discussion are listed below, and other key references can be traced, in turn, through their bibliographies.

See also: **Local attractors for gradient-related descent iterations**; **Large scale trust region problems**; **Nonlinear least squares: Trust region methods**; **Nonlinear least squares: Newton-type methods**; **Broyden family of methods and the BFGS update**; **Unconstrained optimization in neural network training**; **Large scale unconstrained optimization**; **Unconstrained nonlinear optimization: Newton–Cauchy framework**.

References

[1] ADAMS, L., AND NAZARETH, J.L.: *Linear and nonlinear conjugate gradient-related methods*, SIAM, 1996.

[2] BARRETT, R., BERRY, M., CHAN, T.F., DEMMEL, J., DONATO, J., DONGARRA, J., EIJKHOUT, V., POZO, R., ROMINE, C., AND VORST, H. VAN DER: *Templates for the solution of linear systems*, SIAM, 1993.

[3] BERTSEKAS, D.P.: *Nonlinear programming*, second ed., Athena Sci., 1999.

[4] BUCKLEY, A.: 'Extending the relationship between the conjugate gradient and BFGS algorithms', *Math. Program.* **15** (1978), 343–348.

[5] DAI, Y.H.: 'Analyses of nonlinear conjugate gradient methods', *PhD Diss. Inst. Computational Math. Sci./Engin. Computing, Chinese Acad. Sci., Beijing, China* (1997).

[6] DAVIDON, W.C.: 'Variable metric method for minimization', *SIAM J. Optim.* **1** (1991), 1–17, (Original (with a different preface): Argonne Nat. Lab. Report ANL-5990 (Rev., Argonne, Illinois).

[7] FENELON, M.C.: 'Preconditioned conjugate-gradient-type methods for large-scale unconstrained optimization', *PhD Diss. Stanford Univ., Stanford, CA* (1981).

[8] FLETCHER, R., AND REEVES, C.: 'Function minimization by conjugate gradients', *Computer J.* **7** (1964), 149–154.

[9] GOLUB, G.H., AND O'LEARY, D.P.: 'Some history of the conjugate gradient and Lanczos algorithms 1948-1976', *SIAM Rev.* **31** (1989), 50–102.

[10] HESTENES, M.R.: *Conjugate direction methods in optimization*, Vol. 12 of *Appl. Math.*, Springer, 1980.

[11] HESTENES, M.R., AND STIEFEL, E.L.: 'Methods of conjugate gradients for solving linear systems', *J. Res. Nat. Bureau Standards (B)* **49** (1952), 409–436.

[12] KOLDA, T.G., O'LEARY, D.P., AND NAZARETH, J.L.: 'BFGS with update skipping and varying memory', *SIAM J. Optim.* **8** (1998), 1060–1083.

[13] LEONARD, M.W.: 'Reduced Hessian quasi-Newton methods for optimization', *PhD Diss. Univ. Calif., San Diego, CA* (1995).

[14] LUENBERGER, D.G.: *Linear and nonlinear programming*, second ed., Addison-Wesley, 1984.

[15] NAZARETH, J.L.: 'A relationship between the BFGS and conjugate gradient algorithms and its implications for new algorithms', *SIAM J. Numer. Anal.* **16** (1979), 794–800.

[16] NAZARETH, J.L.: 'The method of successive affine reduction for nonlinear minimization', *Math. Program.* **35** (1986), 97–109.

[17] NESTEROV, Y.E.: 'A method of solving a convex programming problem with convergence rate $O(1/k^2)$', *Soviet Math. Dokl.* **27** (1983), 372–376.

[18] NOCEDAL, J.: 'Updating quasi-Newton matrices with limited storage', *Math. Comput.* **35** (1980), 773–782.

[19] NOCEDAL, J.: 'Theory of algorithms for unconstrained optimization', *Acta Numer.* **1** (1992), 199–242.

[20] POLAK, E., AND RIBIÈRE, G.: 'Note sur la convergence de methode de directions conjuguees', *Revue Franc. Inform. Rech. Oper.* **16** (1969), 35–43.

[21] POLYAK, B.T.: 'Some methods of speeding up the convergence of iteration methods', *USSR Comput. Math. Math. Phys.* **4** (1964), 1–17.

[22] POLYAK, B.T.: 'The conjugate gradient method in extremal problems', *USSR Comput. Math. Math. Phys.* **9** (1969), 94–112.

[23] POWELL, M.J.D.: 'Restart procedures for the conjugate gradient method', *Math. Program.* **12** (1977), 241–254.

[24] SHAH, B.V., BUEHLER, R.J., AND KEMPTHORNE, O.: 'Some algorithms for minimizing a function of several variables', *J. SIAM* **12** (1964), 74–91.

J.L. Nazareth

Dept. Pure and Applied Math. Washington State Univ.
Pullman, Washington 99164-3113, USA
Dept. Applied Math. Univ. Washington
Seattle, Washington 98195, USA

E-mail address: nazareth@amath.washington.edu

MSC2000: 90C30

Key words and phrases: unconstrained minimization, conjugate gradients, linear CG method, nonlinear CG method, nonlinear CG-related algorithms, variable-storage, limited-memory, affine-reduced-Hessian, three-term-recurrence.

CONTINUOUS GLOBAL OPTIMIZATION: APPLICATIONS

Nonlinear Systems and Global Optimization. Man-made systems and processes can often be modeled to reasonable accuracy by postulating the exclusive use of continuous linear functions. For instance, one may think of the simplest production and distribution models known from the OR literature. (Models with integer variables will not be discussed here, even though they can be equivalently reformulated, to fit into the present framework.) If we attempt, however, the analysis of natural — physical, chemical, biological, environmental, or even economic, financial and societal — systems and their governing processes, then nonlinear functions start to play a significant role in the quantitative description. To illustrate this point, one may think of the most prominent (basic) function forms in physics: probably polynomials, power functions, the exponential-logarithmic pair and trigonometric functions come to mind first.

Clouds, water flows, rugged terrains, plants and animals — as well as many other natural objects — all possess visible nonlinearities. For sophisticated examples and general principles, one may think of discussions of nonlinear dynamics, chaos, self-organizing systems and the fractal nature of the Universe: consult, e.g., [3], [5], [14], [19], [25].

Prescriptive (control, management, optimization) models which attempt to describe and optimize the behavior of inherently nonlinear systems — as a rule — lead to nonlinear decision problems. Since nonlinear decision models frequently possess multiple local optima, the general relevance of global optimization (GO) becomes obvious. In this brief article we present a list of important and challenging GO applications. We also provide several illustrative references: these describe numerous further application areas.

The Continuous Global Optimization Model. We shall consider problems in the general form

$$\left\{ \begin{array}{ll} \min\limits_{x \in D} & f(x) \\ \\ \text{s.t.} & D := \left\{ x: \begin{array}{c} l \leq x \leq u; \\ g(x) \leq 0, \\ j = 1, \ldots, J \end{array} \right\}. \end{array} \right. \tag{1}$$

In (1) we use the following notation:

- x is a vector which represents decision alternatives in $\mathbf{R}^n$;
- $f(x)$ is a continuous objective function;
- D is a nonempty set of feasible decisions, defined by
  - $g(x)$, an m-vector of continuous constraint functions; and
  - l. u, explicit (finite, componentwise) n-vector bounds.

Explicit bounds on the constraint function values can also be imposed; however, such more specialized models are directly amenable to the form (1).

First of all, note that if all functions are continuous, then — by the classical theorem of Weierstrass — the optimal solution set to (1) is nonempty. At the same time, without further structural assumptions, (1) can be a very difficult global optimization problem. In other words — unless additional information is provided — there may well exist multiple (local) solutions of various quality to (1). Naturally, in most cases one would like to find the 'very best' (global) solution to the underlying decision problem, avoiding the 'traps' offered by local optima. To attain this objective, a considerable variety of GO models and solution approaches have been suggested: consult, e.g., [12].

Test Problems. Although our primary topic is real-world GO applications, one should at least mention several standardized test problem suites, since these often originate from real-world applications. For collections of (both convex and nonconvex) nonlinear programming test problems, consult, e.g., [11], [18]. See [6], [7], [13], [22] for collections of GO test problems. On the WWW, see [8], [1] and [21]; especially [21] provides numerous further links and pointers, including discussions of test and real-world problems.

Illustrative Applications. Since GO problems are literally ubiquitous in scientific, engineering and economic decision making, we shall only list a number of illustrative applications. All application areas will be listed simply in alphabetical order, by information source. (The reader will notice overlaps among the problems studied in different works.)

The test problem collection [6] presents application models from the following fields:

- chemical reactor networks;
- distillation column sequencing;
- heat exchanger network synthesis;
- indefinite quadratic programming;
- mechanical design;
- general nonlinear programming;
- phase and chemical reaction equilibrium;
- pooling and blending;
- quadratically constrained problems;
- reactor-separator-recycling systems;
- VLSI design.

The volume [7] significantly expands upon the above material, adding more specific classes of nonlinear programming models, combinatorial op-

timization problems, and dynamic models, as well as further practical examples (see later on).

The MINPACK-2 collection presented at [1] includes models related to the following types of problems:

- brain activity;
- Chebychev quadrature;
- chemical and phase equilibria;
- coating thickness standardization;
- combustion of propane;
- control systems (analysis and design);
- database optimization;
- design with composites;
- elastic-plastic torsion;
- enzyme reaction analysis;
- exponential data fitting;
- flow in a channel;
- flow in a driven cavity;
- Gaussian data fitting;
- Ginzburg–Landau problem;
- human heart dipole;
- hydrodynamic modeling;
- incompressible elastic rods;
- isomerization of alpha-pinene;
- Lennard–Jones clusters;
- minimal surfaces;
- pressure distribution;
- Ramsey graphs;
- solid fuel ignition;
- steady-state combustion;
- swirling flow between disks;
- thermistor resistance analysis;
- thin film design;
- VLSI design.

A detailed discussion of several GO case studies and applications is presented in [22]. These problems were analyzed by using LGO, an integrated model development environment to formulate and solve GO problems; consult also [23]. The current list of LGO applications includes, for instance, the following areas:

- bio-mechanical design;
- 'black box' (closed, confidential, etc.) system design and operation;
- combination of negotiated expert opinions (forecasts, votes, assessments, etc.);
- data classification, pattern recognition;
- dynamic population and resource management;
- extremal energy (point arrangement) problems, free and surface-constrained forms;
- inverse model fitting to observation data (calibration);
- multifacility location-allocation problems;
- nonlinear approximation, nonlinear regression, and other curve/surface fitting problems;
- optimized tuning of equipment and instruments in medical research and other areas;
- reactor maintenance policy analysis;
- resource allocation (in cutting, loading, scheduling, sequencing, etc. problems);
- risk analysis and control in various environmental management contexts;
- robotics design issues;
- robust product/mixture design;
- satisfiability problems;
- statistical modeling;
- systems of nonlinear equations and inequalities;
- therapy (dosage and schedule) optimization.

The WWW site [21] discusses, inter alia, the following application areas:

- bases for finite elements;
- boundary value problems;
- chemical engineering problems;
- chemical phase equilibria;
- complete pivoting example;
- distance geometry models;
- extreme forms;
- identification of dynamical systems with matrices depending linearly on parameters;
- indefinite quadratic programming models;
- minimax problems;

- nonlinear circuits;
- optimal control problems;
- optimal design;
- parameter identification with data of bounded error;
- PDE defect bounds;
- PDE solution by least squares;
- pole assignment;
- production planning;
- propagation of discrete dynamical systems;
- protein-folding problem;
- pseudospectrum;
- quadrature formulas;
- Runge–Kutta formulas;
- spherical designs (point configurations);
- stability of parameter matrices.

The collection of test problems [7] includes models from the following application areas:

- batch plant design under uncertainty;
- chemical reactor network synthesis;
- conformational problems in clusters of atoms and molecules;
- dynamic optimization problems in parameter estimation;
- homogeneous azeotropic separation system;
- network synthesis;
- optimal control problems;
- parameter estimation and data reconcilliation;
- phase and chemical reaction equilibrium;
- pooling/blending operations;
- pump network synthesis;
- robust stability analysis;
- trim loss minimization.

The article [4] reviews several significant applications of rigorous global optimization (based on the interval branch and bound approach). These applications include:

- currency trading;
- finite element analysis (in a high-tech engineering design context);
- gene prediction in genome therapeutics;
- magnetic resonance imaging (in a medical application);
- numerical mathematics (search for an approximate greatest common divisor of given polynomials);
- parameter estimation in signal processing;
- portfolio management;

One can immediately add here the application of interval techniques to an issue of paramount significance in numerical modeling:

- solving systems of nonlinear (and linear) equations; consult, e.g., [20]

The volume [24] also covers a broad range of applications from the following areas:

- agro-ecosystem management;
- analysis of nucleid acid sequences;
- assembly line design;
- cellular mobile network design;
- chemical process optimization;
- chemical product design;
- computational modeling of atomic and molecular structures;
- controller design for motors;
- electrical engineering design;
- feeding strategies in animal husbandry;
- financial modeling;
- laser equipment design;
- mechanical engineering design;
- radiotherapy equipment calibration;
- robotics design;
- satellite data analysis (interferometry problem);
- virus structure reconstruction;
- water resource distribution systems.

As the above lists illustrate, the application potentials of global optimization are indeed most diverse.

For additional literature on real-world applications, see, e.g., the following references:

- network problems, combinatorial optimization (knapsack, traveling salesman, flow-shop problems), batch process scheduling: [17];

- GO algorithms and their applications (primarily) in chemical engineering design: [9];
- contributions on decision support systems and techniques for solving GO problems, but also on molecular structures, queueing systems, image reconstruction, location analysis and process network synthesis: [2];
- multilevel optimization algorithms and their applications: [15];
- engineering applications of the finite element method: [16];
- a variety of applications, e.g., from the fields of environmental management, geometric design, robust product design, and parameter estimation: [10].

Numerous issues of the *Journal of Global Optimization* — as well as a large number of other professional OR/MS, natural science and engineering journals — also publish articles describing interesting GO applications.

See also: **DIRECT global optimization algorithm; Global optimization based on statistical models; Global optimization using space filling; αBB algorithm; Differential equations and global optimization; Global optimization methods for systems of nonlinear equations; Topology of global optimization; Continuous global optimization: Models, algorithms and software; Global optimization in binary star astronomy; Global optimization in the analysis and management of environmental systems; Interval global optimization; Mixed integer nonlinear programming; Forecasting.**

References

[1] ARGONNE NATIONAL LAB.: *MINPACK-2 test problem collection*, Argonne National Lab., 1993, see also the accompanying notes *Large-scale optimization: Model problems*, by B.M. Averick and J.J. Moré: www-c.mcs.anl/gov/gov/home/more/tprobs/html.

[2] BOMZE, I.M., CSENDES, T., HORST, R., AND PARDALOS, P.M. (eds.): *Developments in global optimization*, Kluwer Acad. Publ., 1997.

[3] CASTI, J.L.: *Searching for certainty*, Morrow, 1990.

[4] CORLISS, G.F., AND KEARFOTT, R.B.: 'Rigorous global search: Industrial applications', in T. CSENDES (ed.): *Developments in Reliable Computing*, Kluwer Acad. Publ., 1999, pp. 1–16.

[5] EIGEN, M., AND WINKLER, R.: *Das Spiel*, Piper, 1975.

[6] FLOUDAS, C.A., AND PARDALOS, P.M.: *A collection of test problems for constrained global optimization algorithms*, No. 455 in Lecture Notes Computer Sci. Springer, 1990.

[7] FLOUDAS, C.A., PARDALOS, P.M., ADJIMAN, C., ESPOSITO, W.R., GUMUS, Z.H., HARDING, S.T., KLEPEIS, J.L., MEYER, C.A., AND SCHWEIGER, C.A.: *Handbook of test problems in local and global optimization*, Kluwer Acad. Publ., 1999.

[8] GRAY, P., HART, W.E., PAINTON, L., PHILLIPS, C., TRAHAN, M., AND WAGNER, J.: *A survey of global optimization methods*, Sandia National Lab., 1999, http://www.cs.sandia.gov/opt/survey/main.html.

[9] GROSSMANN, I.E. (ed.): *Global optimization in engineering design*, Kluwer Acad. Publ., 1996.

[10] HENDRIX, E.M.T.: 'Global optimization at work', *PhD Thesis LU Wageningen, The Netherlands* (1998).

[11] HOCK, W., AND SCHITTKOWSKI, K. (eds.): *Test examples for nonlinear programming codes*, Vol. 187 of *Lecture Notes Economics and Math. Systems*, Springer, 1987.

[12] HORST, R., AND PARDALOS, P.M. (eds.): *Handbook of global optimization*, Kluwer Acad. Publ., 1995.

[13] JANSSON, C., AND KNÖPPEL, O.: 'A global minimization method: The multi-dimensional case', *Res. Report TUHH, Hamburg-Harburg, Germany* (1992).

[14] MANDELBROT, B.B.: *The fractal geometry of nature*, Freeman, 1983.

[15] MIGDALAS, A., PARDALOS, P.M., AND VÄRBRAND, P. (eds.): *Multilevel optimization: Algorithms and applications*, Kluwer Acad. Publ., 1997.

[16] MISTAKIDIS, E., AND STAVROULAKIS, G.E.: *Nonconvex optimization. Algorithms, heuristics and engineering applications of the F.E.M*, Kluwer Acad. Publ., 1997.

[17] MOCKUS, J., EDDY, W., MOCKUS, A., MOCKUS, L., AND REKLAITIS, G.: *Bayesian heuristic approach to discrete and global optimization*, Kluwer Acad. Publ., 1996.

[18] MORÉ, J.J., GARBOW, B.S., AND HILLSTRÖM, K.E.: 'Testing unconstrained optimization software', *ACM Trans. Math. Software* **7** (1981), 17–41.

[19] MURRAY, J.D.: *Mathematical biology*, Springer, 1983.

[20] NEUMAIER, A.: *Interval methods for systems of equations*, Cambridge Univ. Press, 1990.

[21] NEUMAIER, A.: 'Global optimization', *http://solon.cma.univie.ac.at/~neum/glopt.html* (1999).

[22] PINTER, J.D.: *Global optimization in action*, Kluwer Acad. Publ., 1996.

[23] PINTÉR, J.D.: *LGO–A model development system for continuous global optimization. User's guide*, Pintér Consulting Services, Halifax, NS, 1999.

[24] PINTÉR, J.D. (ed.): *Global optimization — Selected case studies*, Kluwer Acad. Publ., 2001.

[25] SCHROEDER, M.: *Fractals, chaos, power laws*, Freeman, 1991.

János D. Pintér
Pintér Consulting Services, Inc.,
and
Dalhousie Univ.
129 Glenforest Drive, Halifax, NS, Canada B3M 1J2
E-mail address: jdpinter@is.dal.ca
Web address: www.is.dal.ca/~jdpinter

MSC2000: 90C05
Key words and phrases: nonlinear decision models, multi-extremality, global optimization, scientific applications, engineering applications, economic applications.

CONTINUOUS GLOBAL OPTIMIZATION: MODELS, ALGORITHMS AND SOFTWARE

The Continuous Global Optimization Model. We shall consider the continuous global optimization problem (GOP) in the general form

$$\begin{cases} \min\limits_{x \in D} & f(x) \\ \text{s.t.} & D := \left\{ x: \begin{array}{c} l \le x \le u; \\ g(x) \le 0, \\ j = 1, \ldots, J \end{array} \right\}. \end{cases} \tag{1}$$

In (1) the following assumptions are used:

- x is a vector representing decision alternatives in $\mathbf{R}^n$;
- D is a nonempty set of feasible decisions, defined by
  - l, u: explicit (finite, componentwise) n-vector bounds of x, and
  - $g(x)$ is an m-vector of continuous constraint functions defined on $[l, u]$;
- $f(x)$ is a continuous objective function defined on D.

Explicit bounds on the constraint function values can also be imposed; however, such more specialized models are also directly amenable to the form (1).

Since the functions f and g are all continuous in D, the GOP (1) evidently has a nonempty globally optimal solution set X^*. At the same time, one can immediately realize that — in its full generality — instances of model (1) can pose a very significant numerical challenge. Since the usual convexity assumptions are absent, D may be disconnected and/or nonconvex, and the objective function f may also be multi-extremal. That is, the number of local (pseudo) solutions to (1) is typically unknown and it can be large; the quality of the various local and global solutions may differ significantly. To illustrate this point, see Fig. 1, which depicts a 'hilly landscape' (in fact, the surface plot of a relatively simple composition of trigonometric functions with embedded polynomial arguments, in just two variables). For instance, this function could be the objective in (1) defined on the corresponding interval feasible region $[l, u]$.

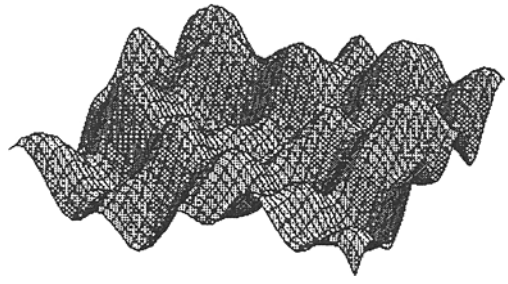


Fig. 1: A two-variable multi-extremal function.

To solve the GOP (1) — in a strict mathematical sense — means to find the complete set of globally optimal solutions X^*, and the associated global optimum value $f^* = f(x^*)$, $x^* \in X^*$. In most cases, at least in the realm of continuous GO, we need to replace this 'ambitious' objective by finding a verified estimate — upper and lower bounds — of f^*, and corresponding approximation(s) of points from the set X^*. Naturally, such estimates are to be determined on the basis of a finite number of algorithmically generated sample points from D, or from the embedding interval $[l, u]$.

For reasons of better analytical and numerical tractability, usually the following additional assumptions are made:

- D is a full-dimensional subset (a 'body') in $\mathbf{R}^n$;
- X^* is at most countable;
- g (i.e., each of its component functions) and f are Lipschitz-continuous on $[l, u]$.

Observe that the first assumption makes algorithmic search possible within the set D. With re-

spect to the second assumption, note that — in most practical contexts — the set of global optimizers consists only of a single point, or of several points. Finally, the Lipschitz assumption — i.e., that changes in function values are uniformly controlled by changes in their argument — is a sufficient condition for estimating f^* on the basis of a finite set of search points. We emphasize that the factual knowledge of the smallest suitable Lipschitz constant is not required — and in practice it is typically unknown indeed. The Lipschitz criterion is evidently met, e.g., by all continuously differentiable functions defined on $[l, u]$; however, their class is even broader.

Due to the very general model structure postulated above, classical (convexity-based) numerical approaches are, generally speaking, not directly applicable to solve GOPs: instead, truly global scope methodology is needed. In the past decades, a considerable variety of GO models and solution approaches have been proposed and analyzed. Below we shall provide a concise review, with a view towards software development. For detailed discussions, consult, e.g., the illustrative list of references.

Model Types. The most important GO model classes that have been extensively studied include the following. (Note that postulated properties of g — such as e.g., convexity — are required componentwise.)

- Bilinear and biconvex programming (f is bilinear or biconvex, D is convex).
- Combinatorial optimization (problems that have discrete decision variables in f and/or in g can be equivalently reformulated as GO problems in continuous variables).
- Concave minimization (f is concave, D is convex).
- Continuous global optimization (f and g are arbitrary continuous functions).
- Differential convex (DC) optimization (f and the components in g can all be explicitly represented, as the difference of two corresponding convex functions).
- Fractional programming (f is the ratio of two real functions, and g is convex).
- Linear and nonlinear complementarity problems (f is the scalar product of two vector functions, D is typically assumed to be convex).
- Lipschitz optimization (f and g are arbitrary Lipschitz-continuous functions).
- Minimax problems (f is some minimax objective, the maximum is considered over a discrete set or a convex set, D is convex).
- Multilevel optimization (e.g., models of noncooperative games, involving hierarchies of decision makers, the conflicting criteria are aggregated by f; D is usually assumed to be convex).
- Multi-objective programming (e.g., determination of the efficient set, when several conflicting objectives are to be optimized over the region D).
- Multiplicative programming (f is the product of several convex functions, and g is convex, or — more generally — also multiplicative).
- Network problems (f can be taken from several nonconvex function classes, and g is typically linear or convex).
- Parametric nonconvex programming (in these the feasible region D and/or the objective f may also depend also on a parameter vector).
- Quadratic optimization (f is an arbitrary — indefinite — quadratic function; g is linear or, in the more general case, is also made up by arbitrary quadratic functions).
- Reverse convex programming (at least one of the functions in g expresses a reverse convex constraint).
- Separable global optimization (f is an arbitrary nonlinear — in general, nonconvex — separable function, D is typically convex).
- Stochastic (nonconvex) models in which the functions f, g depend on random factors.
- Various other nonlinear programming problems, in absence of a verified convex structure: this broad category includes, e.g., models in which some of the functions f, g are de-

fined by complex 'black box' computational procedures.

Note that the problem classes listed are not necessarily distinct; in fact, several of them are hierarchically contained in the more general problem types listed. For detailed descriptions of most of these model types and their connections consult, e.g., [13], with numerous further references.

Observe also that in the list presented, there are specifically structured models (such as e.g., a concave minimization problem under linear or convex constraints), as well as far more general ones (such as e.g., differential convex, Lipschitz or continuous problems). Hence, one can reasonably expect that the most suitably tailored solution approaches will also vary to a considerable extent. Very general search strategies should work for most models — albeit their efficiency might be low for specialized problems. At the same time, strictly specialized solvers may not work at all for problem classes outside of their scope.

Several of the most important GO strategies are listed below, together with additional remarks and references. Again, the items of the list are not necessarily exclusive. Most GO software implementations are based upon one of these approaches, possibly combining ideas from several strategies.

Exact Methods.

Naive Approaches. These include the most well known passive (simultaneous) or direct (not fully adaptive) sequential GO strategies: uniform grid, space covering, and pure random searches. Note that such methods are obviously convergent under mild assumptions, but are — as a rule — impracticable in higher-dimensional problems. Consult corresponding chapters in [13], [24], [30].

Complete (Enumerative) Search Strategies. These are based upon an exhaustive (and typically streamlined) enumeration of all possible solutions. Applicable to combinatorial problems, as well as to certain 'well-structured' continuous GO problems such as, e.g., concave programming. See, e.g., [14].

Homotopy (Parameter Continuation), Trajectory Methods, and Related Approaches. These methods have the 'ambitious' objective of visiting all stationary points of the objective function: this, in turn, leads to the list of all — global as well as local — optima. This general approach includes differential equation model based, path following search strategies, as well as fixed-point methods and pivoting algorithms. See, for instance, [5] and [8].

Successive Approximation (Relaxation) Methods. The initial optimization problem is replaced by a sequence of relaxed subproblems that are easier to solve. Successive refinement of subproblems to approximate the initial problem; cutting planes and more general cuts, diverse minorant function constructions, nested optimization and decomposition strategies are also possible. Applicable to structured GO problems such as, e.g., concave minimization and DC problems ([14].)

Branch and Bound Algorithms. A variety of partition strategies have been proposed to solve GOPs. These are based upon adaptive partition, sampling, and subsequent lower and upper bounding procedures: these operations are applied iteratively to the collection of active (remaining 'candidate') subsets within the feasible set D. Their exhaustive search feature is similar in spirit to analogous integer programming methodology. Branch and bound subsumes many specific approaches, and allows for a range of implementations.

Branch and bound methods typically rely on some a priori structural knowledge about the problem. This information may relate, for instance to how rapidly each function can vary (e.g. the knowledge of a suitable 'overall' Lipschitz constant, for each function f and g); or to the availability of an analytic formulation — and guaranteed smoothness — of all functions (for instance, in interval arithmetic based methods).

The branch and bound methodology is applicable to broad classes of GO problems: e.g., in combinatorial optimization, concave minimization, reverse convex programs, DC programming, and Lipschitz optimization. For details, consult [12], [14], [15], [20], [24], [26].

Bayesian Search (Partition) Algorithms. These methods are based upon some postulated statis-

tical information, to enable a prior stochastic description of the function class modeled. During optimization, the problem instance characteristics are adaptively estimated and updated. Note that, typically only the corresponding one-dimensional model development is exact; furthermore, that in most practical cases 'myopic' approximate decisions govern the search procedure.

This general approach is applicable also to (merely) continuous GO problems. Theoretically, convergence to the optimal solution set is guaranteed only by generating an everywhere dense set of search points. One of the obvious challenges of using statistical methods is the choice and verification of an 'appropriate' statistical model, for the class of problems to which they are applied. Additionally, it seems to be difficult to implement rigorous and computationally efficient versions of these algorithms for higher-dimensional optimization problems. Note, however, that if one 'skips' the underlying Bayesian paradigm, then these methods can also be pragmatically viewed as adaptive partition algorithms, and — as such — they can be directly extended to higher dimensions: see [24]. For detailed expositions on Bayesian approaches, consult, e.g., [18], [19], [27].

Adaptive Stochastic Search Algorithms. This is another broad class of methods, based upon random sampling in the feasible set. In its basic form, it includes various random search strategies that are convergent, with probability one. Search strategy adjustments, clustering and deterministic solution refinement options, statistical stopping rules, etc. can also be added as enhancements.

The methodology is applicable to both discrete and continuous GO problems under very mild conditions. Consult, for instance, [2], [24], [30].

Heuristic Strategies.

'Globalized' Extensions of Local Search Methods. These are partially heuristic algorithms, yet often successful in practice. The essential idea is to apply a preliminary grid search or random search based global phase, followed by applying a local (convex programming) method. For instance, random multistart performs a local search from several points selected randomly from the search domain D. Note that even such sampling is not trivial, when D has a complicated shape, as being defined, e.g., by (merely) continuous nonlinear functions.

Frequently, sophisticated algorithm enhancements are added to this basic strategy. For instance, the clustering of sample points is aimed at selecting only a single point from each sampled 'basin' of f from which then a local search method is initiated. For more details, consult, for instance, [27].

Genetic Algorithms, Evolution Strategies. These methods 'mimic' biological evolution: namely, the process of natural selection and the 'survival of the fittest' principle. An adaptive search procedure based on a 'population' of candidate solution points is used. Iterations involve a competitive selection that drops the poorer solutions. The remaining pool of candidates with higher 'fitness value' are then 'recombined' with other solutions by swapping components with another; they can also be 'mutated' by making some smaller-scale change to a candidate. The recombination and mutation moves are applied sequentially; their aim is to generate new solutions that are biased towards subsets of D in which good — although not necessarily globally optimized — solutions have already been found.

Numerous variants of this general strategy, based on diverse evolution 'game rules', can be constructed. The different types of evolutionary search methods include approaches that are aimed at continuous GOPs, and also others that are targeted towards solving combinatorial problems. The latter group is often called genetic algorithms. For details, consult, e.g., [10], [17], [22], [29].

Simulated Annealing. These techniques are based upon the physical analogy of cooling crystal structures that spontaneously attempt to arrive at some stable (globally or locally minimal potential energy) equilibrium. This general principle is applicable to both discrete and continuous GO problems under mild structural requirements: consult, e.g., [1], [22], [28].

Tabu Search. In this general category of metaheuristics, the essential idea during search is to 'forbid' moves to points already visited in the (usu-

ally discrete) search neighborhood, at least for a number of upcoming steps. This way, one can temporarily accept new inferior solutions, in order to avoid (sub)paths already investigated. This approach can lead to exploring new regions of D, with the goal of finding a solution by 'globalized' search.

Tabu search has traditionally been applied to combinatorial optimization (e.g., scheduling, routing, traveling salesman) problems. The technique can be made — at least, in principle — directly applicable to continuous GOPs by a discrete approximation (encoding) of the problem, but other extensions are also possible. See [9], [22], [29].

Approximate Convex Global Underestimation. This heuristically attractive strategy attempts to estimate the (postulated) large scale, 'overall' convexity characteristics of the objective function f based on directed sampling in D. Applicable to smooth problems. See, e.g., [6].

Continuation Methods. These first transform the potential function into a more smooth ('simpler') function which has fewer local minimizers, and then attempt to trace the minimizers back to the original function. Again, this methodology is applicable to smooth problems. For theoretical background, see, for instance, [8].

Sequential Improvement of Local Optima. These methods usually operate on adaptively constructed auxiliary functions, to assist the search for gradually better optima. The general heuristic principle is realized by so-called tunneling, deflation, and filled function approaches; consult, for example, [16].

Global Optimization Software. In spite of significant theoretical advances in GO, software development and 'standardized' use lag behind. This can be expected due to the potential numerical difficulty of GOPs; recall Fig. 1. Even 'much simpler' problem instances — such as e.g., concave minimization, or indefinite quadratic programming — belong to the hardest (NP) class of mathematical programming problems.

As summarized above, there exist several broad classes of algorithmic GO approaches that possess strong theoretical convergence properties, and — at least in principle — are straightforward to implement. However, all such rigorous approaches involve a computational demand that increases exponentially as a function of problem size, even in case of the simplest GO problem instances. (Consult, for example, [13] for related discussions.) Therefore many practical GO strategies are completed by a 'traditional' local optimization phase. Global convergence, however, needs to be guaranteed by the global scope algorithm component: the latter — at least in theory — should be used in a complete, 'exhaustive' fashion. The above remarks indicate the basic inherent theoretical (and practical) difficulty of developing robust, yet efficient GO software.

Since the computational demand of rigorous strategies can be expected to be some exponential function of the problem dimensionality, GO problems in $\mathbf{R}^n$ (n being just $5, 10, 20, 50, 100, \ldots$) may have rapidly increasing — possibly straight enormous — numerical complexity. This is (and will remain) true, in spite of the fact that computational power seems to grow at an unbelievable pace: the so-called 'curse of dimensionality' is here to stay.

In 1996, a survey on continuous GO software was prepared for the newsletter of the Mathematical Programming Society [23]. Additional information has been collected from the Internet, from several GO books, and from the *Journal of Global Optimization.* Drawing on the responses of software developers and the additional information available, over 50 software products were annotated in that review. (In order to assist in obtaining further information, contact person(s), their e-mail addresses, ftp and/or WWW sites have also been listed.)

Most probably, by now the number of solvers aimed at GOPs is around one hundred (or even more). The general impression is, however, that many of these software products are still at an experimental development stage, and of dominantly 'academic' character, as opposed to 'industrial strength' tools. (Of course, it is not impossible that proprietary software products used by industry and private companies are not announced publicly.)

Below we shall list some key aspects that should be addressed by professional quality GO software development:

- well-specified hardware and software environments (supported development platforms);
- quality user guidance: clearly outlined model development procedure, sensible modeling and troubleshooting tips, user file templates, and (also) nontrivial numerical examples;
- fully functional, 'friendly' user interface;
- 'fool-proof' solver selection and execution procedures;
- good runtime communication and documentation: clear system output for all foreseeable program execution versions and situations, including proper error messages, and result file(s);
- visualization features which are especially desirable in nonlinear systems modeling, to avoid problem misrepresentation, and to assist in finding alternative models and solution procedures;
- reliable, high-quality user support;
- continuous product maintenance and development (since not only science progresses, but hardware devices, operating systems, as well as development platforms are in permanent change).

This tentative 'wish-list' of requirements indicates that although the task is not impossible, it is a challenge. As for an example, we refer to LGO — an integrated model development and solver system — that has been developed with a view towards the desiderata listed above. Details regarding LGO are described, e.g., in [24], [25].

Software Evaluation. In order to obtain information regarding the scope and usability of GO software, it needs to be thoroughly tested. This is a demanding task, when done properly. Consideration needs to be given to the selection of appropriate — nontrivial and practically meaningful — examples. Computational experiments should be carefully designed; and the results should be reported in sufficient details, to assure a fair and accurate assessment. For corresponding discussions and GO (or other) test problems, consult, e.g., [3], [4], [7], [21].

A GO software evaluation framework can be proposed, for instance, along the following guidelines.

Software Applicability Range (Solvable Model Types).

- objective function: concave, DC, Lipschitz, continuous, or some other (general or more special) function form;
- constraints: unconstrained problems, bound constrained problems, linear constraints, general nonlinear smooth constraints;
- additional information related to solvable model types and sizes, with corresponding expected runtimes (within given hardware and software environments).

GO Methodology Applied.

- summary (or more detailed) description of basic principles;
- adequate list of references;

Hardware and Software Requirements.

- supported hardware platforms;
- minimal hardware configuration needed;
- operating systems;
- programming languages and environments;
- compiler(s) needed;
- connectivity to other development environments;
- portability to other hardware and software platforms.

Test Results.

- test problem description, mathematical and/or coded form;
- real world background information (when applicable);
- best known results, with references;
- accuracy requirements, stopping criteria;
- hardware and software environment used in testing;

- standard timing (to facilitate comparisons among different platforms);
- time and computational demand, in order to find the (estimated) global optimum;
- comparative success rate;
- information regarding the reproducibility of results.

Additional Software Information.

- installation procedure;
- user interface features;
- academic and/or professional licenses; conditions of use;
- user support (manual, on-line help, example files, input and result handling, etc.);
- other points of interest.

Of course — at least from a practical point of view — the most meaningful test is to apply GO methods to problems that are of interest in the real world. For numerous existing and prospective GO applications, please consult the related articles **Global optimization in the analysis and management of environmental systems** and **Continuous global optimization: Applications**.

See also: **DIRECT global optimization algorithm**; **Global optimization based on statistical models**; **Global optimization using space filling**; **αBB algorithm**; **Differential equations and global optimization**; **Global optimization methods for systems of nonlinear equations**; **Topology of global optimization**; **Global optimization in binary star astronomy**; **Convex envelopes in optimization problems**; **Global optimization in generalized geometric programming**; **Global optimization of heat exchanger networks**; **Mixed integer linear programming: Heat exchanger network synthesis**; **Mixed integer linear programming: Mass and heat exchanger networks**; **MINLP: Heat exchanger network synthesis**; **MINLP: Mass and heat exchanger networks**; **Global optimization in batch design under uncertainty**; **Smooth nonlinear nonconvex optimization**; **Interval global optimization**; **Global optimization in phase and chemical reaction equilibrium**; **MINLP: Branch and bound global optimization algorithm**; **Optimization software**; **Modeling languages in optimization: A new paradigm**; **Large scale unconstrained optimization**.

References

[1] AARTS, E., AND LENSTRA, J.K. (eds.): *Local search in combinatorial optimization*, Wiley, 1997.

[2] BOENDER, C.G.E., AND ROMEIJN, E.: 'Stochastic methods', in R. HORST AND P.M. PARDALOS (eds.): *Handbook of Global of Optimization*, Kluwer Acad. Publ., 1995, pp. 829–869.

[3] BOMZE, I.M., CSENDES, T., HORST, R., AND PARDALOS, P.M. (eds.): *Developments in global optimization*, Kluwer Acad. Publ., 1997.

[4] DE LEONE, R., MURLI, A., PARDALOS, P.M., AND TORALDO, G. (eds.): *High performance software for nonlinear optimization: Status and perspectives*, Kluwer Acad. Publ., 1998.

[5] DIENER, I.: 'Trajectory methods in global optimization', in R. HORST AND P.M. PARDALOS (eds.): *Handbook of Global Optimization*, Kluwer Acad. Publ., 1995, pp. 649–668.

[6] DILL, K.A., PHILLIPS, A.T., AND ROSEN, J.B.: 'Molecular structure prediction by global optimization', in I.M. BOMZE, T. CSENDES, R. HORST, AND P.M. PARDALOS (eds.): *Developments in Global Optimization*, Kluwer Acad. Publ., 1997, pp. 217–234.

[7] FLOUDAS, C.A., PARDALOS, P.M., ADJIMAN, C., ESPOSITO, W.R., GUMUS, Z.H., HARDING, S.T., KLEPEIS, J.L., MEYER, C.A., AND SCHWEIGER, C.A.: *Handbook of test problems in local and global optimization*, Kluwer Acad. Publ., 1999.

[8] FORSTER, W.: 'Homotopy methods', in R. HORST AND P.M. PARDALOS (eds.): *Handbook of Global Optimization*, Kluwer Acad. Publ., 1995, pp. 669–750.

[9] GLOVER, F., AND LAGUNA, M.: *Tabu search*, Kluwer Acad. Publ., 1997.

[10] GOLDBERG, D.E.: *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley, 1989.

[11] GRAY, P., HART, W.E., PAINTON, L., PHILLIPS, C., TRAHAN, M., AND WAGNER, J.: 'A survey of global optimization methods', *Sandia National Laboratories* (1999), http://www.cs.sandia.gov/opt/survey/main.html.

[12] HANSEN, E.R.: *Global optimization using interval analysis*, M. Dekker, 1992.

[13] HORST, R., AND PARDALOS, P.M. (eds.): *Handbook of global optimization*, Kluwer Acad. Publ., 1995.

[14] HORST, R., AND TUY, H.: *Global optimization–deterministic approaches*, 3rd ed., Springer, 1996.

[15] KEARFOTT, R.B.: *Rigorous global search: Continuous problems*, Kluwer Acad. Publ., 1996.

[16] LEVY, A.V., AND GOMEZ, S.: 'The tunneling method applied to global optimization', *Numerical Optimization*, in P.T. BOGGS (ed.). SIAM, 1984, pp. 213–244.
[17] MICHALEWICZ, Z.: *Genetic algorithms + data structures = evolution programs*, third ed., Springer, 1996.
[18] MOCKUS, J.: *Bayesian approach to global optimization*, Kluwer Acad. Publ., 1989.
[19] MOCKUS, J., EDDY, W., MOCKUS, A., MOCKUS, L., AND REKLAITIS, G.: *Bayesian heuristic approach to discrete and global optimization*, Kluwer Acad. Publ., 1996.
[20] NEUMAIER, A.: *Interval methods for systems of equations*, Cambridge Univ. Press, 1990.
[21] NEUMAIER, A.: 'Global optimization', *http://solon.cma.univie.ac.at/~neum/glopt.html* (1999).
[22] OSMAN, I.H., AND KELLY, J.P. (eds.): *Meta-heuristics: Theory and applications*, Kluwer Acad. Publ., 1996.
[23] PINTÉR, J.D.: 'Continuous global optimization software: A brief review', *Optima* **52** (1996), 1–8, http://plato.la.asu.edu/gom.html.
[24] PINTÉR, J.D.: *Global optimization in action*, Kluwer Acad. Publ., 1996.
[25] PINTÉR, J.D.: 'A model development system for global optimization', in R. DE LEONE, A. MURLI, P.M. PARDALOS, AND G. TORALDO (eds.): *High Performance Software for Nonlinear Optimization: Status and Perspectives*, Kluwer Acad. Publ., 1998, pp. 301–314.
[26] RATSCHEK, H., AND ROKNE, J.G.: 'Interval methods', in R. HORST AND P.M. PARDALOS (eds.): *Handbook of Global Optimization*, Kluwer Acad. Publ., 1995, pp. 751–828.
[27] TÖRN, A.A., AND ŽILINSKAS, A.: *Global optimization*, No. 350 in Lecture Notes Computer Sci. Springer, 1989.
[28] VAN LAARHOVEN, P.J.M., AND AARTS, E.H.L.: *Simulated annealing: Theory and applications*, Kluwer Acad. Publ., 1987.
[29] VOSS, S., MARTELLO, S., OSMAN, I.H., AND ROUCAIROL, C. (eds.): *Meta-heuristics: Advances and trends in local search paradigms for optimization*, Kluwer Acad. Publ., 1999.
[30] ZHIGLJAVSKY, A.A.: *Theory of global random search*, Kluwer Acad. Publ., 1991.

János D. Pintér
Pintér Consulting Services, Inc.,
and
Dalhousie Univ.
129 Glenforest Drive, Halifax, NS, Canada B3M 1J2
E-mail address: jdpinter@is.dal.ca
Web address: www.is.dal.ca/~jdpinter

MSC2000: 90C05
Key words and phrases: nonlinear decision models, continuous global optimization, model types, solution strategies, software development and evaluation.

CONTRACTION-MAPPING

Statement of the Result. The method of *successive substitution*, *Richardson iteration*, or *direct iteration* seeks to find a *fixed point* of a map K, that is a point u^* such that

$$u^* = K(u^*).$$

Given an initial iterate u_0, the iteration is

$$u_{k+1} = K(u_k), \qquad \text{for} \quad k \geq 0. \tag{1}$$

Let X be a Banach space and let $D \subset X$ be closed. A map $K: D \to D$ is a *contraction* if

$$\|K(u) - K(v)\| \leq \alpha \|u - v\| \tag{2}$$

for some $\alpha \in (0,1)$ and all $u, v \in D$. The contraction mapping theorem, [3], [7], [13], [14], states that if K is a contraction on D then

- K has a unique fixed point u^* in D, and
- for any $u_0 \in D$ the sequence $\{u_k\}$ given by (1) converges to u^*.

The message of the contraction mapping theorem is that if one wishes to use direct iteration to solve a fixed point problem, then the fixed point map K must satisfy (2) for some D and relative to some choice of norm. The choice of norm need not be made explicitly, it is determined implicitly by the K itself. However, if there is no norm for which (2) holds, then another, more robust, method, such as Newton's method with a line search, must be used, or the problem must be reformulated.

One may wonder why a Newton-like method is not always better than a direct iteration. The answer is that the cost for a single iteration is very low for Richardson iteration. So, if the equation can be set up to make the contraction constant α in (2) small, successive substitution, while taking more iterations, can be more efficient than a Newton-like iteration, which has costs in linear algebra and derivative evaluation that are not incurred by successive substitution.

Affine Problems. An affine fixed point map has the form

$$K(u) = Mu + b$$

where M is a linear operator on the space X. The fixed point equation is

$$(I - M)u = b, \tag{3}$$

where I is the identity operator. The classical stationary iterative methods in numerical linear algebra, [8], [13], are typically analyzed in terms of affine fixed point problems, where M is called the iteration matrix. Multigrid methods, [2], [4], [9], [5], are also stationary iterative methods. We give an example of how multigrid methods are used later in this article.

The contraction condition (2) holds if

$$\|M\| \leq \alpha < 1. \tag{4}$$

In (4) the norm is the operator norm on X. M may be a well defined operator on more than one space and (4) may not hold in all of them. Similarly, if X is finite dimensional and all norms are equivalent, (4) may hold in one norm and not in another. It is known, [10], that (4) holds for some norm if and only if the spectral radius of M is < 1.

When (4) does not hold it is sometimes possible to form an approximate inverse *preconditioner* P so that direct iteration can be applied to the equivalent problem

$$u = (I - P(I - M))u - Pb. \tag{5}$$

In order to apply the contraction mapping theorem and direct iteration to (5) we require that

$$\|I - P(I - M)\| \leq \alpha < 1$$

in some norm. In this case we say that P is an *approximate inverse* for $I - M$. In the final section of this article we give an example of how approximate inverses can be built for discretizations of integral operators.

Nonlinear Problems. If the nonlinear fixed point map K is sufficiently smooth, then a Newton-like method may be used to solve

$$F(u) = u - K(u) = 0.$$

The transition from a current approximation u_c of u^* to an update u_+ is

$$u_+ = u_c - P(u_c - K(u_c)), \tag{6}$$

where

$$P \approx F'(u^*)^{-1} = (I - K'(u^*))^{-1}.$$

$P = F(u_c)^{-1}$ is Newton's method and $P = F'(u_0)^{-1}$ is the chord method.

It is easy to show [7], [13], [14] that if u is near u^* and P is an approximate inverse for $F'(u^*)$ then the preconditioned fixed point problem

$$u = u - P(u - K(u))$$

is a contraction on a neighborhood D of u^*. This is, in fact, one way to analyze the convergence of Newton's method. In this article our focus is on preconditioners that remain constant for several iterations and do not require computation of the derivative of K.

The point to remember is that, if the goal is to transform a given fixed point map into a contraction, preconditioning of nonlinear problems can be done by the same process (formation of an approximate inverse) as for linear problems.

Integral Equations Example. We close this article with the *Atkinson-Brakhage preconditioner* for integral operators [2], [4]. We will begin with the linear case, from which the nonlinear algorithm is a simple step. Let $\Omega \in \mathbf{R}^N$ be compact and let $k(x, y)$ be a continuous function on $\Omega \times \Omega$. We consider the affine fixed point problem

$$\begin{aligned} u(x) &= f(x) + (\mathbf{K}u)(x) \\ &= f(x) + \int_\Omega k(x, y)u(y)\, dy, \end{aligned}$$

where $f \in X = C(\Omega)$ is given and a solution $u^* \in X$ is sought. In this example $D = X$. We will assume that the linear operator $I - \mathbf{K}$ is nonsingular on X.

We consider a family of increasingly accurate quadrature rules, indexed with a level l, with weights $\{w_i^l\}_{i=1}^{N_l}$ and nodes $\{x_i^l\}_{i=1}^{N_l}$ that satisfy

$$\lim_{l \to \infty} \sum_{j=1}^{N_l} f(x_j^l) w_j^l = \int_\Omega f(x)\, dx$$

for all $f \in X$. The family of operators $\{\mathbf{K}_l\}$ defined by

$$\mathbf{K}_l u(x) = \sum_{j=1}^{N_l} k(x, x_j^l) u(y) w_j^l$$

converges strongly to $\mathbf{K}$, that is

$$\lim_{l \to \infty} \mathbf{K}_l u = \mathbf{K} u$$

for all $u \in X$. The family $\{\mathbf{K}_l\}$ is also *collectively compact*, [1]. This means that if $\mathbf{B}$ is a bounded subset of X, then

$$\cup_l \mathbf{K}_l(\mathbf{B})$$

is precompact in X. The direct consequences of the strong convergence and collective compactness are that $I - \mathbf{K}_l$ are nonsingular for l sufficiently large and

$$(I - \mathbf{K}_l)^{-1} \to (I - \mathbf{K})^{-1} \tag{7}$$

strongly in X. The Atkinson–Brakhage preconditioner is based on these results.

For $g \in X$ one can compute

$$v = (I - \mathbf{K}_l)^{-1} g$$

by solving the finite-dimensional linear system

$$v_i = g(x_i^l) + \sum_{j=1}^{N_l} k(x_i^l, x_j^l) v_j w_j^l \tag{8}$$

for the values $v(x_i^l) = v_i$ of v at the nodal points and then applying the *Nyström interpolation*

$$\begin{aligned} v(x) &= g(x) + \sum_{j=1}^{N_l} k(x, x_j^l) v_j w_j^l \\ &= g(x) + (\mathbf{K}_l v)(x) \end{aligned}$$

to recover $v(x)$ for all $x \in \Omega$. (8) can be solved at a cost of $O(N_l^3)$ floating point operations if direct methods for linear equations are used and for much less if iterative methods such as GMRES [15] are used. In that case, only $O(1)$ matrix-vector products are need to obtain a solution that is accurate to truncation error [6]. This is, up to a multiplicative factor, optimal. The Atkinson–Brakhage preconditioner can dramatically reduce this factor, however.

The results in [1] imply that

$$M_l = I + (I - \mathbf{K}_l)^{-1} \mathbf{K},$$

the Atkinson–Brakhage preconditioner, converges to $(I - K)^{-1}$ *in the operator norm.* Hence, for l sufficiently large (coarse mesh sufficiently fine) Richardson iteration can be applied to the system

$$u = u - M_l(I - \mathbf{K}_l)u - M_l f,$$

where $L \gg l$. Applying this idea for a sequence of grids or levels leads to the optimal form of the Atkinson–Brakhage iteration [11]. The algorithm uses a coarse mesh, which we index with $l = 0$, to build the preconditioner and then cycles through the grids sequentially until the solution at a desired fine ($l = L$) mesh is obtained. One example of this is a sequence of composite midpoint rule quadratures in which $N_{l+1} = 2N_l$. Then, [2], [11], if the coarse mesh is sufficiently fine, only one Richardson iteration at each level will be needed. The cost at each level is two matrix vector products at level l and a solve at level 0.

1) Solve $u_0 - \mathbf{K}_0 u_0 = f$; set $u = u_0$.
2) For $l = 1, \ldots, L$:
   a) Compute $r = u - \mathbf{K}_l u - f$;
   b) $u = u - M_0 r$.

Nonlinear problems can be solved with exactly the same idea. We will consider the special case of *Hammerstein equations*

$$u(x) = \mathbf{K}(u)(x) = \int_\Omega k(x, y, u(y))\, dy.$$

If we use a sequence of quadrature rules as in the linear case we can define

$$\mathbf{K}_l(u)(x) = \sum_{j=1}^{N_l} k(x, x_j^l, u(x_j^l)) w_j^l.$$

The nonlinear form of the Atkinson–Brakhage algorithm for Hammerstein equations simply uses the approximation

$$I + (I - \mathbf{K}_0'(u_0))^{-1} \mathbf{K}'(u) \approx (I - \mathbf{K}_l'(u))^{-1}$$

in a Newton-like iteration. One can see from the formal description below that little has changed from the linear case.

1) Solve $u_0 - \mathbf{K}_0(u_0) = 0$; set $u = u_0$.
2) For $l = 1, \ldots, L$:
   a) Compute $r = u - \mathbf{K}_l(u)$;
   b) $u = u - (I + (I - \mathbf{K}_l'(u_0))^{-1} \mathbf{K}'(u)) r$.

The Atkinson–Brakhage algorithm can, under some conditions, be further improved, [12] and the number of fine mesh operator-function products per level reduced to one. There is also no need to explicitly represent the operator as an integral operator with a kernel.

See also: **Global optimization methods for systems of nonlinear equations; Nonlinear systems of equations: Application to the enclosure of all azeotropes; Interval analysis: Systems of nonlinear equations; Nonlinear least squares: Newton-type methods.**

References

[1] ANSELONE, P.M.: *Collectively compact operator approximation theory*, Prentice-Hall, 1971.

[2] ATKINSON, K.E.: 'Iterative variants of the Nyström method for the numerical solution of integral equations', *Numer. Math.* **22** (1973), 17–31.

[3] BANACH, S.: 'Sur les opérations dans les ensembles abstraits et leur applications aux équations intégrales', *Fundam. Math.* **3** (1922), 133–181.

[4] BRAKHAGE, H.: 'Über die numerische Behandlung von Integralgleichungen nach der Quadraturformelmethode', *Numer. Math.* **2** (1960), 183–196.

[5] BRIGGS, W.: *A multigrid tutorial*, SIAM, 1987.

[6] CAMPBELL, S.L., IPSEN, I.C.F., KELLEY, C.T., MEYER, C.D., AND XUE, Z.Q.: 'Convergence estimates for solution of integral equations with GMRES', *J. Integral Eq. Appl.* **8** (1996), 19–34.

[7] DENNIS, J.E., AND SCHNABEL, R.B.: *Numerical methods for nonlinear equations and unconstrained optimization*, No. 16 in Classics Appl. Math. SIAM, 1996.

[8] GOLUB, G.H., AND VANLOAN, C.G.: *Matrix computations*, Johns Hopkins Univ. Press, 1983.

[9] HACKBUSCH, W.: *Multi-grid methods and applications*, Vol. 4 of *Comput. Math.*, Springer, 1985.

[10] ISAACSON, E., AND KELLER, H.B.: *Analysis of numerical methods*, Wiley, 1966.

[11] KELLEY, C.T.: 'Operator prolongation methods for nonlinear equations', in E.L. ALLGOWER AND K. GEORG (eds.): *Computational Solution of Nonlinear Systems of Equations*, Vol. 26 of *Lect. Appl. Math.*, Amer. Math. Soc., 1990, p. 359–388.

[12] KELLEY, C.T.: 'A fast multilevel algorithm for integral equations', *SIAM J. Numer. Anal.* **32** (1995), 501–513.

[13] KELLEY, C.T.: *Iterative methods for linear and nonlinear equations*, No. 16 in Frontiers in Appl. Math. SIAM, 1995.

[14] ORTEGA, J.M., AND RHEINBOLDT, W.C.: *Iterative solution of nonlinear equations in several variables*, Acad. Press, 1970.

[15] SAAD, Y., AND SCHULTZ, M.H.: 'GMRES a generalized minimal residual algorithm for solving nonsymmetric linear systems', *SIAM J. Sci. Statist. Comput.* **7** (1986), 856–869.

C.T. Kelley
Dept. Math. Center for Research in Sci.
North Carolina State Univ.
Raleigh, NC, USA
E-mail address: tim_kelley@ncsu.edu

MSC2000: 65H10, 65J15
Key words and phrases: nonlinear equations, linear equations, integral equations, iterative method, contraction mapping.

CONTROL VECTOR ITERATION, *CVI*

In solving optimal control problems involving nonlinear differential equations, some iterative procedure must be used to obtain the optimal control policy. As is true with any iterative procedure, one is concerned about the convergence rate and also about the reliability of obtaining the optimal control policy. Although from *Pontryagin's maximum principle* it is known that the minimum of the performance index corresponds to the minimum of the *Hamiltonian*, to obtain the minimum value for the Hamiltonian is not always straightforward. Here we outline a procedure that changes the control policy from iteration to iteration, improving the value of the performance index at each iteration, until the improvement is less than certain amount. Then the iteration procedure is stopped and the results are analyzed. Such a procedure is called *control vector iteration method* (CVI), or *iteration in the policy space*.

To illustrate the procedure, let us consider the *optimal control problem*, where the system is described by the differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \quad \text{with } \mathbf{x}(0) \text{ given}, \tag{1}$$

where $\mathbf{x}$ is an n-dimensional state vector and $\mathbf{u}$ is an r-dimensional control vector. The optimal control problem is to determine the control $\mathbf{u}$ in the time interval $0 \leq t < t_f$, so that the *performance index*

$$I = \int_0^{t_f} \phi(\mathbf{x}, \mathbf{u})\, dt \tag{2}$$

is minimized. We consider the case where the final time t_f is given. To carry out the minimization of

the performance index in (2) subject to the constraints in (1), we consider the *augmented performance index*

$$J = \int_0^{t_f} \left[\phi + \mathbf{z}^\top \left(\mathbf{f} - \frac{d\mathbf{x}}{dt} \right) \right] dt, \tag{3}$$

where the n-dimensional vector of *Lagrange multipliers* $\mathbf{z}$ is called the *adjoint vector*. The last term in (3) can be thought of as a penalty function to ensure that the state equation is satisfied throughout the given time interval. We introduce the Hamiltonian

$$H = \phi + \mathbf{z}^\top \mathbf{f} \tag{4}$$

and use integration by parts to simplify (3) to

$$J = \int_0^{t_f} \left(H + \frac{d\mathbf{z}}{dt}^\top \mathbf{x} \right) dt \tag{5}$$
$$-\mathbf{z}(t_f)\mathbf{x}(t_f) + \mathbf{z}^\top(0)\mathbf{x}(0).$$

The optimal control problem now reduces to the minimization of J.

To minimize J numerically, we assume that we have evaluated J at iteration j by using control policy denoted by $\mathbf{u}^{(j)}$. Now the problem is to determine the control policy $\mathbf{u}^{(j+1)}$ at the next iteration. Since the goal is to minimize J, obviously we want to make the change in J negative and numerically as large as possible. If we let $\delta\mathbf{u} = \mathbf{u}^{(j+1)} - \mathbf{u}^{(j)}$, the corresponding change in J is obtained by using Taylor series expansion up to the quadratic terms:

$$\delta J = \int_0^{t_f} \left[\left(\left(\frac{\partial H}{\partial \mathbf{x}} \right)^\top + \frac{d\mathbf{z}^\top}{dt} \right) \delta\mathbf{x} + \left(\frac{\partial H}{\partial \mathbf{u}} \right)^\top \delta\mathbf{u} \right] dt \tag{6}$$
$$+\frac{1}{2} \int_0^{t_f} \left[\delta\mathbf{x}^\top \frac{\partial^2 H}{\partial \mathbf{x}^2} \delta\mathbf{x} + 2\delta\mathbf{x}^\top \frac{\partial^2 H}{\partial \mathbf{x}\partial \mathbf{u}} \delta\mathbf{u} + \delta\mathbf{u}^\top \frac{\partial^2 H}{\partial \mathbf{u}^2} \delta\mathbf{u} \right] dt - \mathbf{z}^\top(t_f)\delta\mathbf{x}(t_f).$$

The necessary condition for minimum of J is that the first integral in (6) should be zero; i.e.,

$$\frac{d\mathbf{z}}{dt} = -\frac{\partial H}{\partial \mathbf{x}}, \quad \text{with } \mathbf{z}(t_f) = \mathbf{0}, \tag{7}$$

and

$$\frac{\partial H}{\partial \mathbf{u}} = \mathbf{0}. \tag{8}$$

In control vector iteration, we relax the necessary condition in (8) and choose $\delta\mathbf{u}$ to make δJ negative and in the limit (8) is satisfied. One approach is to choose

$$\delta\mathbf{u} = -\epsilon \frac{\partial H}{\partial \mathbf{u}}, \tag{9}$$

where ϵ is a positive parameter which may vary from iteration to iteration. This method is sometimes called *first variation method*, since the driving force for the change in the control policy is based only on the first term of the Taylor series expansion. The negative sign in (9) is required to minimize the Hamiltonian, as is required by Pontryagin's maximum principle. Numerous papers have been written on the determination of the *stepping parameter* ϵ [7].

Instead of arbitrarily determining the stepping parameter ϵ, one may solve the *accessory minimization problem*, where $\delta\mathbf{u}$ is chosen to minimize δJ given by (6) after the requirements for the adjoint are satisfied; i.e., it is required to find $\delta\mathbf{u}$ to minimize δJ given by

$$\delta J = \int_0^{t_f} \left[\left(\frac{\partial H}{\partial \mathbf{u}} \right)^\top \delta\mathbf{u} + \frac{1}{2}\delta\mathbf{x}^\top \frac{\partial^2 H}{\partial \mathbf{x}^2} \delta\mathbf{x} + \delta\mathbf{x}^\top \frac{\partial^2 H}{\partial \mathbf{x}\partial \mathbf{u}} \delta\mathbf{u} + \frac{1}{2}\delta\mathbf{u}^\top \frac{\partial^2 H}{\partial \mathbf{u}^2} \delta\mathbf{u} \right] dt, \tag{10}$$

subject to the differential equation

$$\frac{d\delta\mathbf{x}}{dt} = \left(\frac{\partial \mathbf{f}^\top}{\partial \mathbf{x}} \right)^\top \delta\mathbf{x} + \left(\frac{\partial \mathbf{f}^\top}{\partial \mathbf{u}} \right)^\top \delta\mathbf{u}, \quad \text{with } \delta\mathbf{x}(0) = \mathbf{0}. \tag{11}$$

The solution to this accessory minimization problem is straightforward, since (11) is linear and the performance index in (10) is almost quadratic, and can be easily done, as shown in [1, pp. 259–266] and [7]. The resulting equations, to be integrated backwards from $t = t_f$ to $t = 0$ with zero starting conditions, are

$$\frac{d\mathbf{J}}{dt} + \mathbf{J}\left(\frac{\partial \mathbf{f}^\top}{\partial \mathbf{x}} \right)^\top + \frac{\partial \mathbf{f}^\top}{\partial \mathbf{x}}\mathbf{J} + \frac{\partial^2 H}{\partial \mathbf{x}^2} - \mathbf{S}^\top \left(\frac{\partial^2 H}{\partial \mathbf{u}^2} \right)^{-1} \mathbf{S} = \mathbf{0}, \tag{12}$$

where the $(r \times n)$-matrix $\mathbf{S} = \partial^2 H/\partial \mathbf{u}\partial \mathbf{x} + \partial \mathbf{f}^\top/\partial \mathbf{x}\mathbf{J}$, and

$$\frac{d\mathbf{g}}{dt} - \mathbf{S}^\top \left(\frac{\partial^2 H}{\partial \mathbf{u}^2}\right)^{-1} \left(\frac{\partial H}{\partial \mathbf{u}} + \frac{\partial \mathbf{f}^\top}{\partial \mathbf{x}}\mathbf{g}\right) + \left(\frac{\partial \mathbf{f}^\top}{\partial \mathbf{x}}\right)\mathbf{g} = \mathbf{0}. \tag{13}$$

The control policy is then updated through the equation

$$\mathbf{u}^{(j+1)} = \mathbf{u}^{(j)} - \left(\frac{\partial^2 H}{\partial \mathbf{u}^2}\right)^{-1} \left(\frac{\partial H}{\partial \mathbf{u}} + \frac{\partial \mathbf{f}^\top}{\partial \mathbf{x}}\mathbf{g}\right) - \left(\frac{\partial^2 H}{\partial \mathbf{u}^2}\right)^{-1} \mathbf{S}(\mathbf{x}^{(j+1)} - \mathbf{x}^{(j)}). \tag{14}$$

This method of updating the control policy is called the *second variation method*. In (12) the $(n \times n)$-matrix $\mathbf{J}$ is symmetric, so the total number of differential equations to be integrated backwards is $n(n+1)/2+2n$. However, the convergence is quadratic if the initial control policy is close to the optimum.

To obtain good starting conditions, R. Luus and L. Lapidus [6] suggested the use of first variation method for the first few iterations and then to switch over to the second variation method.

One additional feature of the second variation method is that the control policy given in (14) is a function of the present state, so that the control policy is treated as being continuous and is not restricted to being piecewise constant over an integration time step, as is the case with the first variation method. As was shown in [1, pp. 316–317], for the linear six-plate gas absorber example, when the system equation is linear and the performance index is quadratic, the second variation method yields the optimal control policy in a single step.

However, the large number and complexity of equations required for obtaining the control policy and the instability of the method for very complex systems led to investigating different means of obtaining faster convergence with the first variation method. The effort was directed on the best means of obtaining the stepping parameter ϵ in (9). When ϵ is too large, overstepping occurs, and if ϵ is too small, the convergence rate is very small.

Numerous papers have been written on the determination of ϵ. Several methods were compared by S.N. Rao and Luus [8] in solving typical optimal control problems. Although they suggested a means of determining the 'best' method for performance indices that are almost quadratic, it is found that a very simple scheme is quite effective for a wide variety of optimal control problems. Instead of trying to get very fast convergence and risk instability, the emphasis is placed on the robustness. The strategy is to obtain the initial value for ϵ from the magnitude of $\partial H/\partial \mathbf{u}$, and then increasing ϵ when the iteration has been successful, and reducing its value if overstepping occurs. This type of approach was used in [2] in solving the optimal control of a pyrolysis problem. When the iteration was successful, the stepping parameter was increased by 10 percent, and when overstepping resulted, the stepping parameter was reduced to half its value. The algorithm for first variation method may be presented as follows:

1) Choose an initial control policy $\mathbf{u}^{(0)}$ and a value for ϵ; set the iteration index j to 0.
2) Integrate (1) from $t = 0$ to $t = t_f$ and evaluate the performance index in (2). Store the values of the state vector at the end of each integration time step.
3) Integrate the *adjoint equation* (7) from $t = t_f$ to $t = 0$, using for $\mathbf{x}$ the stored values of the state vector in Step 2. At each integration time step evaluate the gradient $\partial H/\partial \mathbf{u}$.
4) Choose a new control policy
$$\mathbf{u}^{(j+1)} = \mathbf{u}^{(j)} - \epsilon \frac{\partial H}{\partial \mathbf{u}}. \tag{15}$$
5) Integrate (1) from $t = 0$ to $t = t_f$ and evaluate the performance index in (2). Store the values of the state vector at the end of each integration time step. If the performance index is worse (i.e., overstepping has occurred), reduce ϵ to half its value and go to Step 4. If the performance index has been improved increase ϵ by a small factor, such as 1.10 and go to Step 3, and continue for a number of iterations, or terminate the iterations when the change in the performance index in an iteration is less than some criterion, and interpret the results.

Illustration of the First Variation Method. Let us consider the nonlinear continuous stirred tank *reactor* that has been used for optimal control studies in [1, pp. 308–318] and [6], and which was shown in [4] to exhibit multiplicity of solutions. The system is described by the two equations

$$\frac{dx_1}{dt} = -2(x_1 + 0.25) \tag{16}$$
$$+(x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right) - u(x_1 + 0.25),$$

$$\frac{dx_2}{dt} = 0.5 - x_2 \tag{17}$$
$$-(x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right),$$

with the initial state $x_1(0) = 0.09$ and $x_2(0) = 0.09$. The control u is a scalar quantity related to the valve opening of the coolant. The state variables x_1 and x_2 represent deviations from the steady state of dimensionless temperature and concentration, respectively. The performance index to be minimized is

$$I = \int_0^{t_f} (x_1^2 + x_2^2 + 0.1u^2)\, dt, \tag{18}$$

where the final time $t_f = 0.78$. The Hamiltonian is

$$H = z_1(-2(x_1 + 0.25) \tag{19}$$
$$+R - u(x_1 + 0.25))$$
$$+z_2(0.5 - x_2 - R) + x_1^2 + x_2^2 + 0.1u^2,$$

where $R = (x_2 + 0.5)\exp(25x_1/(x_1 + 2))$. The adjoint equations are

$$\frac{dz_1}{dt} = (u + 2)z_1 - 2x_1 + 50R\frac{(z_2 - z_1)}{(x_1 + 2)^2}, \tag{20}$$

$$\frac{dz_2}{dt} = -2x_2 + \frac{(z_2 - z_1)}{(x_2 + 0.5)}R + z_2, \tag{21}$$

and the gradient of the Hamiltonian is

$$\frac{\partial H}{\partial u} = 0.2u - (x_1 + 0.25)z_1. \tag{22}$$

To illustrate the computational aspects of CVI, the above algorithm was used with a Pentium-120 personal computer using WATCOM Fortran compiler version 9.5. The calculations were done in double precision. As found in [4], convergence to the local optimum was obtained when small values for the initial control policy were used, and the global optimum was obtained when large values were used as initial policy. As is seen in Table 1, when an integration time step of 0.0065 was used (allowing 120 piecewise constant steps), in spite of the large number of iterations, the optimal control policy can be obtained in less than 2 seconds of computer time. The iterations were stopped when the change in the performance index from iteration to iteration was less than 10^{-6}.

Initial policy $u^{(0)}$	Performance index	Number of iterations	CPU time s
1.0	0.244436	16	0.16
1.2	0.244436	17	0.17
1.4	0.244436	18	0.11
1.6	0.244436	18	0.16
1.8	0.244436	19	0.22
2.0	0.133128	143	1.49
2.2	0.133128	149	1.53
2.4	0.133128	149	1.54
2.6	0.133130	133	1.43
2.8	0.133129	142	1.37
3.0	0.133130	136	1.38

Table 1: Application of First Variation Method to CSTR.

The total computation time for making this run with 11 different initial control policies was 9.6 seconds on the Pentium-120 digital computer. When an integration time step of 0.00312 was used, the value of the performance index at the global optimum was improved to 0.133104. When a time step of 0.001 was used, giving 780 time steps, the optimal control policy yielded $I = 0.133097$. Even here the computation time for the 11 different initial conditions was only 31 seconds. With the use of piecewise linear control and only 20 time stages, a performance index of $I = 0.133101$ was obtained in [3] with *iterative dynamic programming* (IDP). To obtain this result with IDP, by using 5 randomly chosen points and 10 passes, each consisting of 20 iterations, took 13.4 seconds on a Pentium-120. The use of 15 time stages yielded $I = 0.133112$ and required 7.8 seconds. Therefore, computationally CVI is faster than IDP for this problem, but the present formulation does not allow piecewise linear control to be used in CVI.

Number of time stages P	Optimal I by CVI	Optimal I by IDP
20	0.13429	0.13416
30	0.13363	0.13357
40	0.13339	0.13336
60	0.13323	0.13321
80	0.13317	0.13316
120	0.13313	0.13313
240	0.13310	0.13310

Table 2: Effect of the number of time stages P on the optimal performance index.

The effect of the number of time stages for *piecewise constant control* is shown in Table 2, where CVI results are compared to those obtained by IDP in [5].

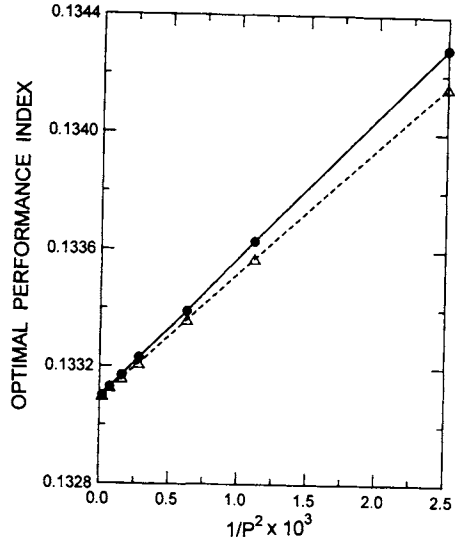


Fig. 1.: Linear variation of optimal performance index with P^{-2}; •—•—• CVI, Δ – –Δ – –Δ IDP.

As can be seen, the given *algorithm* gives results very close to those obtained by IDP, and the deviations decrease as the number of time stages increases, because the approximations introduced during the backward integration when the stored values for the state vector are used, and in the calculation of the gradient of the Hamiltonian in CVI become negligible as the time stages become very small. As is shown in Fig. 1, when the optimal value of the performance index is plotted against $1/P^2$, the extrapolated value, as $1/P^2$ approaches zero, gives the value obtained with the second variation method.

The first variation method is easy to program and will continue to be a very useful method of determining the optimal control of *nonlinear systems*.

See also: **Infinite horizon control and dynamic games; Optimization strategies for dynamic systems; Duality in optimal control with first order differential equations; Semi-infinite programming and control problems; Robust control: Schur stability of polytopes of polynomials; Robust control; Optimal control of a flexible arm; Dynamic programming: Continuous-time optimal control; Hamilton–Jacobi–Bellman equation; Dynamic programming: Optimal control applications; MINLP: Applications in the interaction of design and control; Multi-objective optimization: Interaction of design and control; Sequential quadratic programming: Interior point methods for distributed optimal control problems; Dynamic programming and Newton's method in unconstrained optimal control; Suboptimal control; Boundary condition iteration.**

References

[1] LAPIDUS, L., AND LUUS, R.: *Optimal control of engineering processes*, Blaisdell, 1967.

[2] LUUS, R.: 'On the optimization of oil shale pyrolysis', *Chem. Engin. Sci.* **33** (1978), 1403–1404.

[3] LUUS, R.: 'Application of iterative dynamic programming to very high dimensional systems', *Hungarian J. Industr. Chem.* **21** (1993), 243–250.

[4] LUUS, R., AND CORMACK, D.E.: 'Multiplicity of solutions resulting from the use of variational methods in optimal control problems', *Canad. J. Chem. Engin.* **50** (1972), 309–311.

[5] LUUS, R., AND GALLI, M.: 'Multiplicity of solutions in using dynamic programming for optimal control', *Hungarian J. Industr. Chem.* **19** (1991), 55–62.

[6] LUUS, R., AND LAPIDUS, L.: 'The control of nonlinear systems. Part II: Convergence by combined first and second variations', *AIChE J.* **13** (1967), 108–113.

[7] MERRIAM, C.W.: *Optimization theory and the design of feedback control systems*, McGraw-Hill, 1964, pp. 259–261.

[8] RAO, S.N., AND LUUS, R.: 'Evaluation and improvement of control vector iteration procedures for optimal control', *Canad. J. Chem. Engin.* **50** (1972), 777–784.

Rein Luus
Dept. Chemical Engin. Univ. Toronto
Toronto, ON M5S 3E5, Canada
E-mail address: luus@ecf.utoronto.ca

MSC2000: 93-XX

Key words and phrases: optimal control, control vector iteration, variation method, Pontryagin's maximum principle.

CONVEX ENVELOPES IN OPTIMIZATION PROBLEMS

Let $f\colon S \to \mathbf{R}$ be a *lower semicontinuous function*, where $S \subseteq \mathbf{R}^n$ is a nonempty convex subset. The convex envelope taken over S is a function $f_S\colon S \to \mathbf{R}$ such that

- f_S is a convex function defined over the set S;
- $f_S(x) \le f(x)$ for all $x \in S$;
- if h is any other convex function such that $h(x) \le f(x)$ for all $x \in S$, then $h(x) \le f_S(x)$ for all $x \in S$.

In other words, f_S is the pointwise supremum among any *convex underestimators* of f over S, and is uniquely determined. The following demonstrates the most fundamental properties shown by [6], [3]. Suppose that the minimum of f over S exists. Then,

$$\min\{f(x)\colon x \in S\} = \min\{f_S(x)\colon x \in S\}$$

and

$$\begin{aligned}&\{x^*\colon f(x^*) \le f(x),\ \forall x \in S\}\\ &\subseteq \{x^*\colon f_S(x^*) \le f_S(x),\ \forall x \in S\}.\end{aligned}$$

The properties indicate that an optimal solution of a nonconvex minimization problem could be obtained by minimizing the associated convex envelope. In general, however, finding the convex envelope is at least as difficult as solving the original one.

Several practical results have been proposed for special classes of objective functions and constraints. Suppose that the function f is concave and S is a polytope with vertices $v^0, \dots, v^K$. Then, the convex envelope f_S over S can be expressed as:

$$\begin{cases} f_S(x) = \min & \displaystyle\sum_{i=0}^{K} \alpha_i f(v^i)\\ \text{s.t.} & \displaystyle\sum_{i=0}^{K} \alpha_i v^i = x,\\ & \displaystyle\sum_{i=0}^{K} \alpha_i = 1,\\ & \alpha_i \ge 0, \quad i = 0, \dots, K. \end{cases}$$

Especially, if S is an n-simplex with vertices $v^0, \dots, v^n$, f_S is the affine function

$$f_S(x) = a^\top x + b,$$

which is uniquely determined by solving the following linear system

$$a^\top v^i + b = f(v^i), \quad i = 0, \dots, n.$$

The properties above have been used to solve concave minimization problems with linear constraints [6], [4].

The following property shown in [5], [1] is frequently used in the literature. For each $i = 1, \dots, p$, let $f^i\colon S_i \to \mathbf{R}$ be a continuous function, where $S_i \subseteq \mathbf{R}^{n_i}$, and let $n = n_1 + \cdots + n_p$. If

$$f(x) = \sum_{i=1}^{p} f^i(x^i)$$

and

$$S = S_1 \times \cdots \times S_p,$$

where $x^i \in \mathbf{R}^{n_i}$, $i = 1, \dots, p$, and $x = (x^1, \dots, x^p) \in \mathbf{R}^n$, then the convex envelope $f_S(x)$ can be expressed as:

$$f_S(x) = \sum_{i=1}^{p} f_S^i(x^i).$$

In particular, let $f(x) = \sum_{i=1}^{n} f_i(x_i)$ be a separable function, where $x = (x_1, \dots, x_n) \in \mathbf{R}^n$, and let $f_i(x_i)$ be concave for each $i = 1, \dots, n$. Then the convex envelope of $f(x)$ over the rectangle $R = \{x \in \mathbf{R}^n\colon a_i \le x_i \le b_i,\ i = 1, \dots, n\}$ can be the affine function, which is given by the sum of the linear functions below:

$$f_R(x) = \sum_{i=1}^{n} l_i(x_i),$$

where $l_i(x_i)$ meets $f_i(x_i)$ at both ends of the interval $a_i \le x_i \le b_i$ for each $i = 1, \dots, n$. B. Kalantari and J.B. Rosen [7] show an algorithm for the global

minimization of a quadratic concave function over a polytope. They exploit convex envelopes of separable functions over rectangles to generate lower bounds in a *branch and bound scheme*.

Also, convex envelopes of *bilinear functions* over rectangles have been proposed in [2]. Consider the following rectangles:

$$\Omega_i = \left\{ (x_i, y_i) \colon \begin{array}{c} l_i \le x_i \le L_i, \\ m_i \le y_i \le M_i \end{array} \right\},$$
$$i = 1, \ldots, n,$$

and let

$$f^i(x_i, y_i) = x_i y_i, \quad i = 1, \ldots, n,$$

be bilinear functions with two variables. It has been shown that for each $i = 1, \ldots, n$, the convex envelope of $f^i(x_i, y_i)$ over Ω_i is expressed as:

$$f^i_{\Omega_i}(x_i, y_i)$$
$$= \max\{m_i x_i + l_i y_i - l_i m_i, M_i x_i + L_i y_i - L_i M_i\}.$$

Moreover, it can be verified that $f^i_{\Omega_i}(x_i, y_i)$ agrees with $f^i(x_i, y_i)$ at the four extreme points of Ω_i. Thus, the convex envelope of the general bilinear function

$$f(x, y) = x^\top y = \sum_{i=1}^{n} f^i(x_i, y_i),$$

where $x^\top = (x_1, \ldots, x_n)$ and $y^\top = (y_1, \ldots, y_n)$ over $\Omega = \Omega_1 \times \cdots \times \Omega_n$ can be expressed as

$$f_\Omega(x, y) = \sum_{i=1}^{n} f^i_{\Omega_i}(x_i, y_i).$$

Another characterization of convex envelopes of bilinear functions over a special type of polytope, which includes a rectangle as a special case, is derived in [8].

See also: **αBB algorithm; Global optimization in generalized geometric programming; MINLP: Global optimization with αBB.**

References

[1] AL-KHAYYAL, F.A.: 'Jointly constrained bilinear programs and related problems: An overview', *Comput. Math. Appl.* **19**, no. 11 (1990), 53–62.

[2] AL-KHAYYAL, F.A., AND FALK, J.E.: 'Jointly constrained biconvex programming', *Math. Oper. Res.* **8**, no. 2 (1983), 273–286.

[3] BAZARAA, M.S., SHERALI, H.D., AND SHETTY, C.M.: *Nonlinear programming: Theory and algorithms*, Wiley, 1993.

[4] BENSON, H.P., AND SAYIN, S.: 'A finite concave minimization algorithm using branch and bound and neighbor generation', *J. Global Optim.* **5**, no. 1 (1994), 1–14.

[5] FALK, J.E.: 'Lagrange multipliers and nonconvex programs', *SIAM J. Control* **7** (1969), 534–545.

[6] FALK, J.E., AND HOFFMAN, K.: 'A successive underestimation method for concave minimization problems', *Math. Oper. Res.* **1**, no. 3 (1976), 251–259.

[7] KALANTARI, B., AND ROSEN, J.B.: 'An algorithm for global minimization of linearly constrained concave quadratic functions', *Math. Oper. Res.* **12**, no. 3 (1987), 544–561.

[8] SHERALI, H.D., AND ALAMEDDINE, A.: 'An explicit characterization of the convex envelope of a bivariate bilinear function over special polytopes', *Ann. Oper. Res.* **25**, no. 1-4 (1990), 197–209.

Yasutoshi Yajima
Tokyo Inst. Technol.
Tokyo, Japan
E-mail address: yasutosi@me.titech.ac.jp

MSC2000: 90C26
Key words and phrases: convex underestimator, nonconvex optimization.

CONVEX MAX-FUNCTIONS, *minimax*

Examples of the Problem. In a classic nonlinear program (NLP) a smooth objective function is minimized on a feasible set defined by finitely many smooth constraints. However, many optimization problems have an objective function that is not smooth but is of the max type, that is, defined as

$$f(x) := \max\{f_\ell(x) \colon \ell \in \mathcal{L}\}, \qquad (1)$$

where the functions $f_\ell \colon \mathbf{R}^n \to \mathbf{R}$ are themselves smooth.

When $\mathcal{L}$ is finite, the minimization of f is called a *finite minimax problem*. In a general *minimax problem*, the indices ℓ can range over an infinite compact set $\mathcal{L}$, see [6]. Those f_ℓ's realizing the maximum in (1) are called *active functions*. The corresponding indices form the *active index set*, defined as $\mathcal{L}_a(x) := \{l \in \mathcal{L} \colon f(x) = f_l(x)\}$.

There are numerous examples of optimization problems dealing with the minimization of convex max-functions:

- When processing empirical data $\{(t_\ell, p_\ell)\colon \ell = 1, \ldots, m\}$, consider the problem of selecting the coefficients x of a polynomial $p_x(t) = \sum_{j=0}^{n} x_{j+1} t^j$ that fits well the data. The quality of the approximation can be measured through the deviation $f_\ell(x) := |p_\ell - p_x(t_\ell)|$, defined for all ℓ. Depending on the nature of the problem, it can be interesting to minimize either the sum of squared deviations or the maximum deviation. The first case is a *least squares problem*, while the second is known as *Chebyshev best approximation*, and is a particular instance of (1), with index set $\mathcal{L} = \{\ell\colon \ell = 1, \ldots, m\}$.

- A more general case is finding the best approximation of a continuous function φ_0 on a compact interval $\mathcal{L}$. Instead of n powers t^j, n linearly independent functions $\varphi_j\colon \mathcal{L} \to \mathbf{R}$ are given. To find the linear combination $\sum_j x_j \varphi_j$ which best approximates φ_0 in the max-norm comes to solve (1), with $f_\ell(x) = |\sum_j x_j \varphi_j(\ell) - \varphi_0(\ell)|$. Because the problem has infinitely many constraints, it is also an example of semi-infinite programming.

- A basic problem in structural optimization, see [2], is to find the stiffest structure of a given volume that is able to carry loads varying on a given set. Optimization can be performed through the variation of sizing variables, like the thickness of bars in a truss; shape variables, as the splines defining the boundary of the body; or even the distribution and properties of the composite material used to make the structure itself. After discretization by finite elements, the design problem has an objective function as in (1). The f_ℓ's therein have usually the form $f_\ell(x) = (1/2)x^\top A_\ell x - b^\top x$, with A_ℓ denoting the stiffness matrix of the ℓth element of the grid.

- Some large scale mixed integer problems can be solved by *decomposition techniques* using Lagrangian relaxation. The idea is to coordinate, by means of a master program, the iterative resolution of problems of smaller dimension or complexity, called local problems. When applying price decomposition, the master is led to maximize a dual function, say $\theta(\lambda)$, on the space of multipliers Λ, using some iterative method. In production planning problems, the iterate λ^k sent to the local solvers can be interpreted as prices paid by the master. Let $\mathcal{L} = \prod_i \mathcal{L}^i$ denote the (decomposed) primal space and let $L(x, \lambda) = \sum_i L^i(x^i, \lambda)$ be the (decomposed) Lagrangian of the original problem. Each local unit i decides its corresponding optimal level of production by solving $\min_{x^i \in \mathcal{L}^i} L^i(x^i, \lambda^k)$. Having those local optimal levels, the master adjusts prices by updating λ^k in order to maximize the dual function θ, defined as the pointwise minimum of the Lagrangian: $\theta(\lambda) = \min_{x \in \mathcal{L}} L(x, \lambda) = \sum_i \min_{x^i \in \mathcal{L}^i} L^i(x^i, \lambda)$. An equivalent problem for the master is to minimize $f(\lambda) := -\theta(\lambda)$. This last formulation has an objective f that is a max-function as in (1), letting $f_\ell(\lambda) = -\sum_i L^i(x^i_\ell, \lambda)$, for each $x_\ell \in \mathcal{L}$.

- Solving a nonlinear program using *exact penalties* leads to the iterative minimization of penalized objective which are max-functions, with the max-operation involving the constraints.

- In game theory, consider a zero sum game, with two players whose strategy is to optimize their individual choice against the worst possible selection by the other player. The first player can choose his action over n possible moves, with probability distribution $x = (x_1, \ldots, x_n)$. To every possible move $j = 1, \ldots, m$ of the second player, corresponds a loss $a_{i,j}$ player I pays to player II. Let ℓ be a continuous index counting elements in the set of probability distributions of the second player: $\mathcal{L} = \{z \in \mathbf{R}^m\colon \sum_{j=1}^m z_j = 1,\ z_j \geq 0\}$. Calling $A = [a_{i,j}]$ the $n \times m$ loss-matrix, the average amount to be paid by player I is $f_\ell(x) = x^\top A z_\ell$. It follows that the first player needs to solve a problem like (1). The mini-maximization of the bivariate function $F(x, z) = x^\top A z$ is also called a *saddle-point problem*.

Continuity and Optimality Conditions. When taking the pointwise maximum in (1), some properties of the functions f_ℓ are transmitted to f. Such is the case of continuity and convexity, but not of differentiability.

More precisely, f is continuous when both the f_ℓ-s and its gradients ∇f_ℓ are continuous. When the underlying functions f_ℓ are convex, so is f.

Convexity implies that the max-function f is differentiable almost everywhere. Nevertheless, at those points where more than one underlying function f_l realizes the maximum, the gradient fails to exist. Typically, such is the case at $\overline{x}$, a minimizer of f. For instance, suppose that $\mathcal{L}_a(\overline{x}) = \{1, 2\}$, i.e., there are two active functions at $\overline{x}$: $f(\overline{x}) = f_1(\overline{x}) = f_2(\overline{x})$. Clearly, for $\nabla f(\overline{x})$ to exist, the unlikely equality $\nabla f(\overline{x}) = \nabla f_1(\overline{x}) = \nabla f_2(\overline{x})$ needs to hold. Moreover, the optimality condition for minimizing f on $\mathbf{R}^n$ would require all the involved gradients to be null.

Rather than a single gradient, it is possible to define a whole set of *subgradients* by making convex combinations of $\nabla f_1(\overline{x})$ and $\nabla f_2(\overline{x})$. For an arbitrary convex max-function f, at any given x, the set of subgradients is the so-called *subdifferential* of f at x. Its expression is given by the formula

$$\partial f(x) = \left\{ \sum_{l \in \mathcal{L}_a(x)} \alpha_l \nabla f_l(x) \colon \alpha \in \Delta \right\}, \qquad (2)$$

where Δ is the unit simplex

$$\Delta := \left\{ \alpha \in \mathbf{R}^{|\mathcal{L}_a(x)|} \colon \sum_{l \in \mathcal{L}_a(x)} \alpha_l = 1,\ \alpha_l \geq 0 \right\}. \qquad (3)$$

When $\mathcal{L}$ in (1) is an infinite compact set, (2) still holds, provided the application $\ell \mapsto f_\ell(x)$ is upper semicontinuous for each x, see [8, Chap. VI, §4.4].

Consider the constrained problem

$$\min_{x \in \Omega} f(x), \qquad (4)$$

where $\Omega \subset \mathbf{R}^n$ is a closed convex set and f is the function defined in (1). Assume the index set $\mathcal{L}$ is infinite and suppose (4) has a solution $\overline{x}$ such that $f(\overline{x}) = \max\{f_\ell(\overline{x}) \colon \ell \in \mathcal{L}\}$. Then it can be proved (see [6, Chap. VI, Thm. 3.3]) that (4) is equivalent to the finite minimax

$$\min_{x \in \Omega} \max_{i=1,\ldots,r} \{f_{\ell_i}(x) \colon \ell_i \in \mathcal{L}\}, \qquad (5)$$

with $r \leq n + 1$. The set $\{\ell_1, \ldots, \ell_r\}$ is called an *extremal basis* of (4).

When Ω satisfies a constraint qualification condition of Slater type, see, for instance, [7, Chap. III], a necessary optimality condition (OC) characterizing a minimizer $\overline{x}$ of (4) is

$$0 \in \partial f(\overline{x}) + N_\Omega(\overline{x}), \qquad (6)$$

where N is the normal cone of convex analysis. Because f is convex, the optimality condition is also sufficient.

The optimality condition (6) can be further specified when Ω is represented by a set of convex inequalities:

$$\Omega := \{x \in \mathbf{R}^n \colon c_j(x) \leq 0,\ j \in \mathcal{J}\}. \qquad (7)$$

Observe that Ω may contain an infinite number of constraints c_j, assumed to be smooth and convex. Using (1) together with (7), the following characterization of $\overline{x}$ results:

LEMMA 1 There exist $r \leq n + 1$ and $s \leq n$ such that for the index sets $\mathcal{L}_a(\overline{x}) := \{l_1, \ldots, l_r\} \subset \mathcal{L}$ and $\{j_1, \ldots, j_s\} \subset \mathcal{J}$ it holds

$$\sum_{i=1}^{r} \alpha_i \nabla f_{l_i}(\overline{x}) + \sum_{i=1}^{s} \mu_i \nabla c_{j_i}(\overline{x}) = 0, \qquad (8)$$

where the multipliers α and μ are positive and α is an element of the simplex Δ in $\mathbf{R}^{|\mathcal{L}|_a(\overline{x})}$ from (3). □

This characterization ensures the existence of an extremal basis of (4) near $\overline{x}$.

Algorithms of Minimization. Depending on the nature of the problem, several approaches have been proposed to solve (4):

- Reformulation as a NLP.
- Minimization of the nonsmooth max-function.
- Determination of a saddle point.
- Search of an extremal basis.

For example, the amount of available information can determine the method of resolution: if for any given x, all the active indices in (1) are known, then the full subdifferential (2) is available and a nonlinear programming technique can be applied.

Nonlinear Programming. An important feature of this approach is that smooth NLP techniques have a superlinear rate of convergence. The essential idea is first to write (4) as an NLP with an additional variable:

$$\begin{cases} \min\limits_{\substack{r\in\mathbf{R}\\ x\in\Omega}} & r \\ \text{s.t.} & r \geq f_\ell(x), \quad \ell \in \mathcal{L}, \end{cases} \tag{9}$$

and then solve the associated optimality conditions by using a Newton-like method, such as sequential quadratic programming (cf. also **Successive quadratic programming**) or interior point schemes, see [4, Parts III–IV] and [3, §§4.3–4.4].

Nonsmooth Optimization. Sometimes the explicit knowledge of all the active constraints in (9) can be difficult, if not impossible, to obtain; such is the case for structural optimization problems.

On the other hand, it is often possible to obtain a single subgradient almost for free when computing $f(x)$. Indeed, suppose that just one active index l in $\mathcal{L}_a(x)$ is known: $f(x) = f_l(x)$. Then, because of (2), $\nabla f_l(x) \in \partial f(x)$.

Algorithms from nonsmooth optimization, such as bundle methods [8, Chaps. XIV–XV], are designed to minimize a general convex function, possibly nondifferentiable, with the information furnished by an oracle that gives $f(x)$ and only one subgradient at x. Nonsmooth optimization techniques, specialized to a max-function like f in (1), have been successfully used in [12] and [9] to solve general minimax problems.

Although bundle methods are essentially first order methods, in recent years some proposals have been given that aim at obtaining better than linear convergence. They consist of a combination of bundle, proximal and quasi-Newton techniques ([10], [5], [11]).

Other Methods of Resolution. V.F. Demyanov and V.N. Malozemov treat (4) in an indirect way, by solving an infinite sequence of finite minimax problems. Keeping (5) in mind, the idea is to asymptotically identify an extremal basis by making successive approximations on a finite grid of the index set $\mathcal{L}$.

In game theory, rather than solving (4) by some 'mini-maximization' procedure, it can be more convenient to find a *saddle point*. That is, an equilibrium point satisfying $\min_x \max_{z_\ell} F(x, z_\ell) = \max_{z_\ell} \min_x F(x, z_\ell)$, with $F(x, z_\ell) := f_\ell(x)$. The determination of saddle points of F can be performed taking advantage of the extra structure of the problem. Some popular methods are Arrow–Hurwicz's and Uzawa's, see [1].

See also: **Lagrangian multipliers methods for convex programming.**

References

[1] Arrow, K.J., Hurwicz, L., and Uzawa, H.: *Studies in linear and nonlinear programming*, Stanford Univ. Press, 1958.

[2] Bendsøe, M.P.: *Optimization of structural topology, shape and material*, Springer, 1995.

[3] Bertsekas, D.P.: *Nonlinear programming*, Athena Sci., 1995.

[4] Bonnans, J.F., Gilbert, J.Ch., Lemaréchal, C., and Sagastizábal, C.: *Optimisation numérique: Aspects théoriques et pratiques*, Springer, 1997.

[5] Chen, X., and Fukushima, M.: 'Proximal quasi-Newton methods for nondifferentiable convex optimization', *Math. Program.* **85**, no. 2 (1999), 313–334.

[6] Dem'yanov, V.F., and Malozemov, V.N.: *Introduction to minimax*, Wiley, 1974.

[7] Hiriart-Urruty, J.-B.: *L'optimisation*, No. 3184 in Que sais-je? Press. Univ. France, 1996.

[8] Hiriart-Urruty, J.-B., and Lemaréchal, C.: *Convex analysis and minimization algorithms*, Vol. 305-306 of *Grundl. Math. Wiss*, Springer, 1993, (two volumes).

[9] Kiwiel, K.C.: 'A direct method of linearization for continuous minimax problems', *J. Optim. Th. Appl.*, no. 55 (1987), 271–287.

[10] Lemaréchal, C., and Sagastizábal, C.: 'Variable metric bundle methods: from conceptual to implementable forms', *Math. Program.* **76** (1997), 393–410.

[11] Mifflin, R., Sun, D.F., and Qi, L.Q.: 'Quasi-Newton bundle-type methods for nondifferentiable convex optimization', *SIAM J. Optim.* **8**, no. 2 (1998), 583–603.

[12] Panin, V.M.: 'Linearization method for continuous min-max problems', *Kibernetika*, no. 2 (1981), 75–78.

Claudia Sagastizábal
IMPA
Estrada dona Castorina 110
Jardim Botânico
Rio de Janeiro RJ 22460-320, Brazil
E-mail address: sagastiz@impa.br

MSC2000: 49K35, 49M27, 65K10, 90C25

Key words and phrases: minimax problem, convex optimization, max-function.

CONVEX-SIMPLEX ALGORITHM

W.I. Zangwill [9] first proposed the *convex-simplex algorithm* (CSA) for the following problem:

$$\min_{x \in S} f(x), \tag{1}$$

where $f(x)$ is a *pseudoconvex function* on $\mathbf{R}^n$. The set S is a nonempty *polyhedron*, i.e., $S = \{x \in \mathbf{R}^n : Ax = b, x \geq 0\}$, A is a $m \times n$ matrix, and b is a vector in $\mathbf{R}^m$. For simplicity, S is assumed to be bounded also.

CSA belongs to a class of algorithms called *feasible direction methods*. Given an initial feasible solution, algorithms in this class solve problem (1) by iteratively generating an improving feasible direction that leads to another feasible solution with an improved objective value. The name 'convex-simplex' is to indicate that the algorithm generates improving feasible directions in manner similar to the simplex algorithm for linear programs. When $f(x)$ is linear, the algorithm is identical to the simplex algorithm. In general, a vector d is an *improving feasible direction* at a point, x, feasible to problem (1) if the following (sufficient) conditions hold

a) $\nabla f(x)^\top d < 0$,

b) $Ad = 0$, and

c) $d_j \geq 0$ if $x_j = 0$.

It follows from the first order Taylor series expansion of $f(x)$ that

$$f(x + \lambda d) = f(x) + \lambda \nabla f(x)^\top d + \lambda \|d\| \alpha(x; \lambda d),$$

where $\lim_{\lambda \to 0} \alpha(x; \lambda d) = 0$. Via the above expansion, condition a) implies that $f(x + \lambda d) < f(x)$ for a sufficiently small $\lambda > 0$, i.e., d leads to an improvement in the objective function. The remaining two conditions guarantee that d can produce a point in S. In particular, condition b) yields the following:

$$A(x + \lambda d) = Ax + \lambda Ad = Ax = b.$$

This shows that $x + \lambda d$ is always feasible with respect to the equality constraint. Next, each component of $x + \lambda d$ can be written as

$$x_i + \lambda d_i = \begin{cases} x_i + \lambda d_i & \text{if } x_i > 0, \\ \lambda d_i & \text{if } x_i = 0. \end{cases}$$

When λ is a sufficiently small positive number, $x_i + \lambda d_i$ remains nonnegative in the first case. For the second case, it follows directly from condition c) that $\lambda d_i \geq 0$ for all $\lambda > 0$. Thus, $x + \lambda d \in S$ when λ is sufficiently small.

To describe how CSA generates an improving feasible direction, let a_j denote the jth column of A. Also, assume that every m columns of A are linearly independent and every extreme point of S has m strictly positive components. Under these assumptions, every feasible solution has at least m positive components and at most $(n - m)$ zero components. Given a feasible solution x, let $I(x)$ be the set of indices for the m largest components of x. Then, A can be partitioned into $[B, N]$, where $B = [a_j : j \in I(x)]$ and $N = [a_j : j \notin I(x)]$. Similarly, $x^\top$ can be partitioned into $[x_B^\top, x_N^\top]$ where $x_B^\top$, the *basic component*, corresponds to components of x belonging to $I(x)$, and $x_N^\top$, the *nonbasic component*, corresponds to components not in $I(x)$. By the above assumptions, $x_B^\top > 0$ and B is nonsingular.

Partitioning the direction vector, $d^\top$, into its basic and nonbasic components, i.e., $[d_B^\top, d_N^\top]$, produces the following sequence of relationships:

$$\begin{aligned} Ad &= 0, \\ Bd_B + Nd_N &= 0, \\ d_B &= -B^{-1}Nd_N. \end{aligned}$$

The last equality yields the following:

$$\begin{aligned} \nabla f(x)^\top d &= \nabla_B f(x)^\top d_B + \nabla_N f(x)^\top d_N \\ &= [\nabla_N f(x)^\top - \nabla_B f(x)^\top B^{-1} N] d_N \\ &= r_N^\top d_N = \sum_{j \notin I(x)} r_j d_j, \end{aligned}$$

where $r_N^\top \equiv \nabla_N f(x)^\top - \nabla_B f(x)^\top B^{-1} N$. In order for d to be an improving direction, $\nabla f(x)^\top d < 0$. There are several approaches to make this inner product negative. Each approach generates a different algorithm and all of which can be viewed as an extension or variant of the *reduced gradient algorithm* first proposed by P. Wolfe [8].

Like the simplex algorithm, CSA allows only one nonbasic component of d to be nonzero. In particular, let

$$j^+ = \operatorname*{argmax}_{j \notin I(x)} \{-r_j : r_j < 0\}$$

and

$$j^- = \underset{j \notin I(x)}{\operatorname{argmax}} \{x_j r_j : x_j > 0, r_j > 0\}.$$

Then, CSA chooses d_N as follows. If $-r_{j^+} \geq x_{j^-} r_{j^-}$, then $d_{j^+} = 1$ and $d_j = 0$ for the remaining nonbasic components. This makes $\nabla f(x)^\top d = r_{j^+} < 0$. Otherwise, $d_{j^-} = -1$ instead and $\nabla f(x)^\top d = -r_{j^-} < 0$. Given d_N, the basic component can be computed using the relationship $d_B = -B^{-1} N d_N$.

When $r_N \geq 0$ and $r_N^\top x_N = 0$, the indices j^+ and j^- are undefined in the above construction. When this occurs, x is globally optimal and d_N is usually set to zero to indicate that there is no improving feasible direction. To demonstrate, it is sufficient to show that there exist vectors $\mu^\top = [\mu_B^\top, \mu_N^\top] \geq 0$ and v (unrestricted) satisfying the following equations:

$$\nabla_B f(x) + B^\top v - \mu_B = 0,$$
$$\nabla_N f(x) + N^\top v - \mu_N = 0,$$
$$\mu_B^\top x_B = 0,$$
$$\mu_B^\top x_N = 0.$$

These equations are known as the *Karush–Kuhn–Tucker conditions* (see, e.g., [2]) and they are sufficient optimality conditions for problem (1). Letting $\mu_B = 0$, $\mu_N = \nabla_N f(x) - N(B^\top)^{-1} \nabla_N f(x)$, and $v = -(B^\top)^{-1} \nabla_B f(x)$ satisfies the first three conditions. Since $\mu_N = r_N$, the above assumptions concerning r_N imply that $\mu_N \geq 0$ and $\mu_N^\top x_N = 0$. Thus, the Karush–Kuhn–Tucker conditions hold and x must be globally optimal.

When $d_N \neq 0$, a better feasible point can be obtained from a solution to the following problem, typically called the *line search problem*:

$$\min_{0 \leq \lambda \leq \lambda_{\max}} f(x + \lambda d),$$

where $\lambda_{\max} = \min_j \{-x_j/d_j : d_j < 0\}$. This prevents components of $x + \lambda d$ from being negative. (If S is unbounded, then every component of d may be nonnegative and $\lambda_{\max} = \infty$.) Algorithms such as the bisection search, the golden section method, and an inexact line search technique (e.g., the *Armijo rule*, [1]) can efficiently solve the line search problem.

To summarize, CSA can be stated as follows:

0	Select $x^1 \in S$ and set $k = 1$.
1	Identify $I(x^k)$ and form the submatrices B and N. Compute $$r_N^\top = \nabla_N f(x^k)^\top - \nabla_B f(x^k)^\top B^{-1} N.$$
2	IF $r_N \geq 0$ and $r_N^\top x_N^k = 0$, THEN stop and x^k is an optimal solution. ELSE, let $$j^+ = \underset{j \notin I(x^k)}{\operatorname{argmax}} \{-r_j : r_j < 0\},$$ $$j^- = \underset{j \notin I(x^k)}{\operatorname{argmax}} \left\{x_j^k r_j : x_j^k > 0, r_j > 0\right\}.$$ Set $d_N^k = 0$. IF $-r_{j^+} \geq x_{j^-}^k r_{j^-}$, THEN set $d_{j^+}^k = 1$. ELSE, set $d_{j^-}^k = -1$ instead. Set $d_B^k = B^{-1} N d_N^k$. Go to Step 3.
3	Set $\lambda_{\max} = \min_j \{-x_j^k/d_j^k : d_j^k < 0\}$ and compute $$\lambda^k = \underset{0 \leq \lambda \leq \lambda_{\max}}{\operatorname{argmin}} f(x^k + \lambda d^k).$$ Then, set $x^{k+1} = x^k + \lambda^k d^k$ and $k = k + 1$. Return to Step 1.

The convex-simplex algorithm (CSA).

The convergence proof for CSA is the same as that of the reduced gradient algorithm and follows standard arguments in nonlinear programming. Although CSA behaves like the simplex algorithm, CSA converges slowly when compared to other algorithms. This is due in part to the restriction that only one nonbasic component of the improving feasible direction can be nonzero. To accelerate CSA, B.A. Murtagh and M.A. Saunder [5] used a second order approximation for the objective function and allowed several nonbasic components to be nonzero. The latter is often referred to as *superbasic variables*.

For other developments, S. Nguyen [6] and R.V. Helgason and J.L. Kennington [4] specialized CSA to nonlinear network flow problems. D.P. Rutenberg [7] (see also [3]) demonstrated that special techniques for solving linear programs with generalized network and generalized upper bounding structure also extend to CSA.

See also: **Sequential simplex method; Parametric linear programming: Cost simplex algorithm; Linear programming; Lemke method; Linear complementarity problem.**

References

[1] ARMIJO, L.: 'Minimization of functions having Lipschitz continuous first-partial derivatives', *Pacific J. Math.* **16** (1966), 1–3.

[2] BAZARAA, M.S., SHERALI, H.D., AND SHETTY, C.M.: *Nonlinear programming: Theory and algorithm*, Wiley, 1993.

[3] HSIA, W.S.: 'On Rutenberg's decomposition method', *Managem. Sci.* **21** (1974), 10–12.

[4] KENNINGTON, J.L., AND HELGASON, R.V.: *Algorithms for network programming*, Wiley, 1980.

[5] MURTAGH, B.A., AND SAUNDER, M.A.: 'Large-scale linearly constrained optimization', *Math. Program.* **14** (1978), 14–72.

[6] NGUYEN, S.: 'An algorithm for the traffic assignment problem', *Transport. Sci.* **8** (1974), 203–216.

[7] RUTENBERG, D.P.: 'Generalized networks, generalized upper bounding, and decomposition of the convex simplex method', *Managem. Sci.* **16** (1970), 388–401.

[8] WOLFE, P.: 'Methods of nonlinear programming', in R.L. GRAVES AND P. WOLFE (eds.): *Recent Advances in Mathematical Programming*, 1963.

[9] ZANGWILL, W.I.: 'The convex simplex method', *Managem. Sci.* **14** (1967), 221–283.

Siriphong Lawphongpanich
Naval Postgraduate School
Monterey, California, USA
E-mail address: SLawphon@nps.navy.mil

MSC2000: 90C30

Key words and phrases: basic component, convex-simplex algorithm, extreme point, feasible direction methods, first order Taylor series expansion, generalized networks, generalized upper bounding structure, golden section method, improving feasible direction, inexact line search technique, Karush–Kuhn–Tucker conditions, linearly independent, line search problem, linear program, nonbasic component, nonlinear network flow problems, nonlinear programming, nonsingular, polyhedron, pseudoconvex function, reduced gradient algorithm, second order approximation, simplex algorithm, superbasic variables.

COST APPROXIMATION ALGORITHMS, *CA algorithms*

The notion of cost approximation (CA) was created in the thesis [39], to describe the construction of the subproblem of a class of iterative methods in mathematical programming. In order to explain the notion of CA, we will consider the following conceptual problem (the full generality of the algorithm is explained in detail in [45] and in [37], [38], [40], [42], [43], [44], [46]):

$$\begin{cases} \min & T(x) := f(x) + u(x), \\ \text{s.t.} & x \in X, \end{cases} \tag{1}$$

where $X \subseteq \mathbf{R}^n$ is nonempty, closed and convex, $u\colon \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ is lower semicontinuous (l.s.c.), proper and convex, and $f\colon \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ is continuously differentiable (for short, in C^1) on $\operatorname{dom} u \cap X$, where dom denotes 'effective domain'. This problem is general enough to cover convex optimization ($f = 0$), unconstrained optimization ($f = 0$ and $X = \mathbf{R}^n$), and differentiable constrained optimization ($u = 0$). We note that if $\operatorname{int}(\operatorname{dom} u) \cap X$ is nonempty, then any locally optimal solution x^* satisfies the inclusion

$$-\nabla f(x^*) \in \partial u(x^*) + N_X(x^*), \tag{2}$$

where N_X is the normal cone operator for X and ∂u is the subdifferential mapping for u. Equivalently, by the definitions of these two operators,

$$\nabla f(x^*)^\top (x - x^*) + u(x) - u(x^*) \geq 0, \quad x \in X.$$

The CA algorithm was devised in order to place a number of existing algorithms for (1) in a common framework, thereby facilitating comparisons, for example, between their convergence properties. In short, the method works iteratively as follows. Note that, from (2), we seek a zero of the mapping $[\nabla f + \partial u + N_X]$. Given an iterate, $x^t \in \operatorname{dom} u \cap X$, this mapping is approximated by a monotone mapping, constructed so that a zero of which is easier to find. Such a point, y^t, is then utilized in the search for a new iterate, x^{t+1}, having the property that the value of some merit function for (1) is reduced sufficiently, for example through a line search in T along the direction of $d^t := y^t - x^t$.

Instances of the CA algorithm. To obtain a monotone approximating mapping, we introduce a monotone mapping $\Phi^t\colon \operatorname{dom} u \cap X \to \mathbf{R}^n$, which replaces the (possibly nonmonotone) mapping ∇f; by subtracting off the error at x^t, $[\Phi^t - \nabla f](x^t)$, from Φ^t, so that the resulting mapping becomes $[\Phi^t + \partial u + N_X] + [\nabla f - \Phi^t](x^t)$, the CA subproblem becomes the inclusion

$$[\Phi^t + \partial u + N_X](y^t) + [\nabla f - \Phi^t](x^t) \ni 0^n. \tag{3}$$

We immediately reach an interesting fixed-point characterization of the solutions to (2):

THEOREM 1 (Fixed-point, [45]) The point x^t solves (2) if and only if $y^t = x^t$ solves (3). □

This result is a natural starting point for devising stopping criteria for an algorithm.

Assume now that $\Phi^t \equiv \nabla\varphi^t$ for a convex function φ^t. We may then derive the inclusion equivalently as follows. At x^t, we replace f with the function φ^t, and subtract off the linearization of the error at x^t; the subproblem objective function then becomes

$$T_{\varphi^t}(y) := \varphi^t(y) + u(y) + [\nabla f(x^t) - \nabla\varphi^t(x^t)]^\top y.$$

It is straightforward to establish that (3) is the optimality conditions for the convex problem of minimizing T_{φ^t} over X.

Linearization Methods. Our first example instances utilize Taylor expansions of f to construct the approximations.

Let $u = 0$ and $X = \mathbf{R}^n$. Let $\Phi^t(y) := (1/\gamma_t)Q^t y$, where $\gamma_t > 0$ and Q^t is a symmetric and positive definite mapping in $\mathbf{R}^{n\times n}$. The inclusion (3) reduces to

$$\nabla f(x^t) + \frac{1}{\gamma_t}Q^t(y^t - x^t) = 0^n,$$

that is, $y^t = x^t - \gamma_t(Q^t)^{-1}\nabla f(x^t)$. The direction of $y^t - x^t$, $d^t := -\gamma_t(Q^t)^{-1}\nabla f(x^t)$, is the search direction of the class of *deflected gradient methods*, which includes the *steepest descent method* ($Q^t := I^n$, the identity matrix) and *quasi-Newton methods* (Q^t equals (an approximation of) $\nabla^2 f(x^t)$, if positive definite). (See further [47], [35], [50], [5].)

In the presence of constraints, this choice of Φ^t leads to $y^t = P_X^{Q^t}[x^t - \gamma_t(Q^t)^{-1}\nabla f(x^t)]$, where $P_X^{Q^t}[\cdot]$ denotes the projection onto X with respect to the norm $\|z\|_{Q^t} := \sqrt{z^\top Q^t z}$. Among the algorithms in this class we find the *gradient projection algorithm* ($Q^t := I^n$) and *Newton's method* ($Q^t := \nabla^2 f(x^t)$, $\gamma_t := 1$). (See [19], [27], [50], [5].)

A first order Taylor expansion of f is obtained from choosing $\varphi^t(y) := 0$; this results in $T_{\varphi^t}(y) = \nabla f(x^t)^\top y$ (if $u = 0$ is still assumed), which is the subproblem objective in the Frank–Wolfe algorithm ([17], [5]; cf. also **Frank–Wolfe algorithm**).

We next provide the first example of the very useful fact that the result of the cost approximation (in the above examples a linearization), leads to different approximations of the original problem, and ultimately to different algorithms, depending on which *representation* of the problem to one applies the cost approximation.

Consider the problem

$$\begin{cases} \min & f(x) \\ \text{s.t.} & g_i(x) = 0, \quad i = 1, \dots, \ell, \end{cases} \tag{4}$$

where f and g_i, $i = 1, \dots, \ell$, are functions in C^2. We may associate this problem with its first order optimality conditions, which in this special case is

$$F(x^*, \lambda^*) := \begin{pmatrix} \nabla_x L(x^*, \lambda^*) \\ -\nabla_\lambda L(x^*, \lambda^*) \end{pmatrix} = \begin{pmatrix} 0^n \\ 0^\ell \end{pmatrix}, \tag{5}$$

where $\lambda \in \mathbf{R}^\ell$ is the vector of Lagrange multipliers for the constraints in (4), and $L(x, \lambda) := f(x) + \lambda^\top g(x)$ is the associated Lagrangian function. We consider using Newton's method for this system, and therefore introduce a (primal-dual) mapping $\Phi\colon \mathbf{R}^{2(n+\ell)} \to \mathbf{R}^{n+\ell}$ of the form

$$\Phi((y,p),(x,\lambda)) := \nabla F(x,\lambda)\begin{pmatrix} y \\ p \end{pmatrix} = \begin{pmatrix} \nabla_x^2 L(x,\lambda) & \nabla g(x)^\top \\ -\nabla g(x) & 0 \end{pmatrix}\begin{pmatrix} y \\ p \end{pmatrix}.$$

The resulting CA subproblem in (y, p) can be written as the following linear system:

$$\nabla_x^2 L(x,\lambda)(y - x) + \nabla g(x)^\top p = 0^n,$$
$$\nabla g(x)(y - x) = -g(x);$$

this system constitutes the first order optimality conditions for (e.g., [4, Sec. 10.4])

$$\begin{cases} \min & f(x) + \nabla f(x)^\top(y - x) \\ & \quad + \frac{1}{2}(y - x)^\top \nabla_x^2 L(x,\lambda)(y - x) \\ \text{s.t.} & g(x) + \nabla g(x)^\top(y - x) = 0^\ell, \end{cases}$$

where we have added some fixed terms in the objective function for clarity. This is the generic subproblem of *sequential quadratic programming* (SQP) methods for the solution of (4); see, for example, [16], [5].

Regularization, Splitting and Proximal Point Methods. We assume now that $f := f_1 + f_2$, where f_1 is convex on $\operatorname{dom} u \cap X$, and rewrite the cost

mapping as

$$[\nabla f + \partial u + N_X]$$
$$= [\nabla \varphi^t + \nabla f_1 + \partial u + N_X] - [\nabla \varphi^t - \nabla f_2].$$

The CA subproblem is, as usual, derived by fixing the second term at x^t; the difference to the original setup is that we have here performed an *operator splitting* in the mapping ∇f to keep an additive part from being approximated. (Note that such a splitting can always be found by first choosing f_1 as a convex function, and then define $f_2 := f - f_1$. Note also that we can derive this subproblem from the original derivation by simply redefining $\varphi^t := \varphi^t + f_1$.) We shall proceed to derive a few algorithms from the literature.

Consider choosing $\varphi^t(y) = 1/(2\gamma_t)\|y - x^t\|^2$, $\gamma_t > 0$. If $f_2 = 0$, then we obtain the subproblem objective $T_{\varphi^t}(y) = T(y) + 1/(2\gamma_t)\|y - x^t\|^2$, which is the subproblem in the *proximal point algorithm* (e.g., [33], [34], [32], [51], [52]). This is the most classical algorithm among the regularization methods. More general choices of strictly convex functions φ^t are of course possible, leading for example to the class of regularization methods based on Bregman functions ([9], [14], [22]) and ψ-divergence functions ([54], [23]). If, on the other hand, $f_1 = 0$, then we obtain the gradient projection algorithm if also $u = 0$.

We can also construct algorithms in between these two extremes, yielding a true operator splitting. If both f_1 and f_2 are nonzero, choosing $\varphi^t = 0$ defines a *partial linearization* ([25]) of the original objective, wherein only f_2 is linearized. Letting $x = (x_1^\top, x_2^\top)^\top$, the choice $\varphi^t(y) = 1/(2\gamma_t)\|y_1 - x_1^t\|^2$ leads to the *partial proximal point algorithm* ([20], [7]); choosing $\varphi^t(y) = f(y_1, x_2^t)$ leads to a linearization of f in the variables x_2.

Several well-known methods can be derived either directly as CA algorithms, or as *inexact proximal point algorithms*. For example, the *Levenberg–Marquardt algorithm* ([49], [5]), which is a Newton-like algorithm wherein a scaled diagonal matrix is added to the Hessian matrix in order to make the resulting matrix positive definite, is the result of solving the proximal point subproblem with one iteration of a Newton algorithm. Further, the *extragradient algorithm* of [24] is the result of instead applying one iteration of the gradient projection algorithm to the proximal point subproblem.

The perhaps most well-known splitting algorithm is otherwise the class of *matrix splitting methods in quadratic programming* (e.g., [35], [36], [29], [28]). In a quadratic programming problem, we have

$$f(x) = \frac{1}{2}x^\top Ax + q^\top x,$$

where $A \in \mathbf{R}^{n\times n}$. A splitting (A_1^t, A_2^t) of this matrix is one for which $A = A_1^t + A_2^t$, and it is further termed regular if $A_1^t - A_2^t$ is positive definite. Matrix splitting methods correspond to choosing

$$f_1(x) = \frac{1}{2}x^\top A_1^t x,$$

and results in the CA subproblem mapping $y \mapsto A_1^t y + [A_2^t x^t + q]$, which obviously is monotone whenever A_1^t was chosen positive semidefinite.

Due to the fact that proximal point and splitting methods have dual interpretations as augmented Lagrangian algorithms ([51]), a large class of multiplier methods is included among the CA algorithms. See [45, Chapt. 3.2–3.3] for more details.

Perturbation Methods. All the above algorithms assume that

i) the mappings ∂u and N_X are left intact; and

ii) the CA subproblem has the fixed-point property of Theorem 1.

We here relax these assumptions, and are then able to derive subgradient algorithms as well as perturbed CA algorithms which include both regularization algorithms and exterior penalty algorithms.

Let $[\Phi^t + N_X] + [\nabla f + \partial u + \Phi^t]$ represent the original mapping, having moved ∂u to the second term. Then by letting any element $\xi_u(x^t) \in \partial u(x^t)$ represent this point-to-set mapping at x^t, we reach the subproblem mapping of the *auxiliary problem principle* of [12]. Further letting $\Phi^t(y) = (1/\gamma_t)[y - x^t]$ yields the subproblem in the classical *subgradient optimization* scheme ([48], [53]), where, assuming further that $f = 0$, $y^t := P_X[x^t - \gamma_t \xi_u(x^t)]$. (Typically, $\ell_t := 1$ is taken.)

Let again $[\Phi^t + \partial u + N_X] + [\nabla f + \Phi^t]$ represent the original problem mapping, but further let u be replaced by an *epiconvergent sequence* $\{u^t\}$

of l.s.c., proper and convex functions. An example of an epiconvergent sequence of convex functions is provided by convex exterior penalty functions. In this way, we can construct CA algorithm that approximate the objective function and simultaneously replace some of the constraints of the problem with exterior penalties. See [3], [13] for example methods of this type.

One important class of regularization methods takes as the subproblem mapping $[\Phi^t + \nabla f + \partial u + N_X]$, where Φ^t is usually taken to be strongly monotone (cf. (12)). This subproblem mapping evidently does not have the fixed-point property, as it is not identical to the original one at x^t unless $\Phi^t(x^t) = 0^n$ holds. In order to ensure convergence, we must therefore force the sequence $\{\Phi^t\}$ of mappings to tend to zero; this is typically done by constructing the sequence as $\Phi^t := (1/\gamma_t)\Phi$ for a fixed mapping Φ and for a sequence of $\gamma_t > 0$ constructed so that $\{\gamma_t\} \to \infty$ holds. For this class of algorithms, F. Browder [10] has established convergence to a unique limit point x^* which satisfies $-\Phi(x^*) \in N_{X^*}(x^*)$, where X^* is the solution set of (2). The origin of this class of methods is the work of A.N. Tikhonov [55] for *ill-posed problems*, that is, problems with multiple solutions. The classical regularization mapping is the scaled identity mapping, $\Phi^t(y) := (1/\gamma_t)[y]$, which leads to least squares (least norm) solutions. See further [56], [49].

Variational Inequality Problems. Consider the following extension of (2).

$$-F(x^*) \in \partial u(x^*) + N_X(x^*), \tag{6}$$

where $F\colon X \to \mathbf{R}^n$ is a continuous mapping on X. When $F = \nabla f$ we have the situation in (2), and also in the case when $F(x,y) = (\nabla_x \Pi(x,y)^\top, -\nabla_y \Pi(x,y)^\top)^\top$ holds for some saddle function Π on some convex product set $X \times Y$ (cf. (5)), the *variational inequality problem* (6) has a direct interpretation as the necessary optimality conditions for an optimization problem. In other cases, however, a merit function (or, objective function), for the problem (6) is not immediately available. We will derive a class of suitable merit functions below.

Given the convex function $\varphi\colon \operatorname{dom} u \cap X \to \mathbf{R}$ in C^1 on $\operatorname{dom} u \cap X$, we introduce the function

$$\psi(x) := \sup_{y \in X} L(y,x), \quad x \in \operatorname{dom} u \cap X, \tag{7}$$

where

$$L(y,x) := u(x) - u(y) + \varphi(x) - \varphi(y) + [F(x) - \nabla\varphi(x)]^\top (x-y). \tag{8}$$

We introduce the optimization problem

$$\min_{x \in X} \psi(x). \tag{9}$$

THEOREM 2 (Gap function, [45]) For any $x \in X$, $\psi(x) \geq 0$ holds. Further, $\psi(x) = 0$ if and only if x solves (6). Hence, the solution set of (6) (if nonempty) is identical to that of the optimization problem (9), and the optimal value is zero. □

The Theorem shows that the CA subproblem defines an auxiliary function ψ which measures the violation of (6), and which can be used (directly or indirectly) as a merit function in an algorithm.

To immediately illustrate the possible use of this result, let us consider the extension of Newton's method to the solution of (6). Let $x \in \operatorname{dom} u \cap X$, and consider the following cost approximating mapping: $y \mapsto \Phi(y,x) := \nabla F(x)(y-x)$. The CA subproblem then is the linearized variational inequality problem of finding $y \in \operatorname{dom} u \cap X$ such that

$$[F(x) + \nabla F(x)^\top (y-x)]^\top (z-y) + u(z) - u(y) \geq 0, \quad \forall z \in X. \tag{10}$$

Assuming that x is not a solution to (6), we are interested in utilizing the direction $d := y - x$ in a line search based on a merit function. We will utilize the *primal gap function* ([62], [2]) for this purpose, which corresponds to the choice $\varphi := 0$ in the definition of ψ. We denote the primal gap function by ψ_{p}. Let w be an arbitrary solution to its inner problem, that is, $\psi_{\mathrm{p}}(x) = u(x) - u(w) + F(x)^\top (x-w)$. The steplength is chosen such that the value of ψ_{p} decreases sufficiently; to show that this is possible, we use Danskin's theorem and the variational inequality (10) with $z = w$ to obtain (the maximum is taken over all w defining $\psi_{\mathrm{p}}(x)$)

$$\psi'_{\mathrm{p}}(x;d) := \max_w \left\{ \left[F(x) + \nabla F(x)^\top (x-w)\right]^\top d + u'(x;d) \right\}$$

$$\leq -\psi_{\mathrm{p}}(x) - d^{\top}\nabla F(x)^{\top}d,$$

which shows that d defines a direction of descent with respect to the merit function ψ_{p} at all points outside the solution set, whenever F is monotone and in C^1 on $\operatorname{dom} u \cap X$. (See also [30] for convergence rate results.) So, if Newton's method is supplied with a line search with respect to the primal gap function, it is globally convergent for the solution of variational inequality problems.

The merit function ψ and the optimization problem (9) cover several examples previously considered for the solution of (6).

The primal gap function, as typically all other gap functions, is nonconvex, and further also nondifferentiable in general. In order to utilize methods from differentiable optimization, we consider letting φ be strictly convex, whence the solution y^t to the inner problem (7) is unique. Under the additional assumption that $\operatorname{dom} u \cap X$ is bounded and that u is in C^1 on this set, ψ is in C^1 on $\operatorname{dom} u \cap X$. Among the known differentiable gap functions that are covered by this class of merit functions we find those of [1], [18], [26], [40], and [59], [60], [61], [31].

Descent Properties.

Optimization. Assume that x^t is not a solution to (2). We are interested in the conditions under which the direction of $d^t := y^t - x^t$ provides a descent direction for the merit function T. Let $d^t := \overline{y}^t - x^t$, where $\overline{y}^t$ is a possibly inexact solution to (3). Then, if $\Phi^t = \nabla\varphi^t$, the requirement is that

$$T_{\varphi^t}(\overline{y}^t) < T_{\varphi^t}(x^t), \tag{11}$$

that is, any improvement in the value of the subproblem objective over that at the current iterate is enough to provide a descent direction. To establish this result, one simply utilizes the convexity of φ^t and u and the formula for the directional derivative of T in the direction of d^t (see [45, Prop. 2.14.b]). We further note that (11) is possible to satisfy if and only if x^t is not a solution to (2); this result is in fact a special case of Theorem 1.

If Φ^t has stronger monotonicity properties, descent is also obtained when Φ^t is not necessarily a gradient mapping, and, further, if it is Lipschitz continuous then we can establish measures of the steepness of the search directions, extending the gradient relatedness conditions of unconstrained optimization. Let Φ^t be *strongly monotone* on $\operatorname{dom} u \cap X$, that is, for $x, y \in \operatorname{dom} u \cap X$,

$$[\Phi^t(x) - \Phi^t(y)]^{\top}(x - y) \geq m_{\Phi^t}\|x - y\|^2, \tag{12}$$

for some $m_{\Phi^t} > 0$. This can be used to establish that

$$T'(x^t; d^t) \leq -m_{\Phi^t}\left\|d^t\right\|^2.$$

If y^t is not an exact solution to (3), in the sense that for a vector $\overline{y}^t$, we satisfy a perturbation of (3) where its right-hand side 0^n is replaced by $r^t \neq 0^n$, then $d^t := \overline{y}^t - x^t$ is a descent direction for T at x^t if $\|r^t\| < m_{\Phi^t}\|d^t\|$.

Variational Inequality Problems. The requirements for obtaining a descent direction in the problem (6) are necessarily much stronger than in the problem (2), the reason being the much more complex form that the merit functions for (6) takes. (For example, the directional derivative of T at x in any direction d depends only on those quantities, while the directional derivative of ψ depends also on the argument y which defines its value at x.) Typically, monotonicity of the mapping F is required, as is evidenced in the above example of the Newton method. If further a differentiable merit function is used, the requirements are slightly strengthened, as the following example result shows.

THEOREM 3 (Descent properties, [60], [45]) Assume that X is bounded, u is finite on X and F is monotone and in C^1 on X. Let $\varphi\colon X \times X \to \mathbf{R}$ be a continuously differentiable function on $X \times X$ of the form $\varphi(y, x)$, strictly convex in y for each $x \in X$. Let $\alpha > 0$. Let $x \in X$, y be the unique vector in X satisfying

$$\psi_\alpha(x) := \max_{y \in X} L_\alpha(y, x),$$

where

$$L_\alpha(y, x) := u(x) - u(y) + \frac{1}{\alpha}[\varphi(x, x) - \varphi(y, x)] + \left[F(x) - \frac{1}{\alpha}\nabla_y\varphi(x, x)\right]^{\top}(x - y).$$

Then, with $d := y - x$, either d satisfies

$$\psi'_\alpha(x;d) \leq -\gamma\psi_\alpha(x), \quad \gamma \in (0,1),$$

or

$$\psi_\alpha(x) \leq -\frac{1}{\alpha(1-\gamma)}(\varphi(y,x) + \nabla_x\varphi(y,x)^\top d).$$

□

A descent algorithm is devised from this result as follows. For a given $x \in X$ and choice of $\alpha > 0$, the CA subproblem is solved with the scaled cost approximating, continuous and iteration-dependent function φ. If the resulting direction does not have the descent property, then the value of α is increased and the CA subproblem rescaled and resolved. Theorem 3 shows that a sufficient increase in the value of α will produce a descent direction unless x solves (6).

Steplength Rules. In order to establish convergence of the algorithm, the steplength taken in the direction of d^t must be such that the value of the merit function decreases sufficiently. An exact line search obviously works, but we will introduce simpler steplength rules that do not require a one-dimensional minimization to be performed.

The first is the *Armijo rule*. We assume temporarily that $u = 0$. Let $\alpha, \beta \in (0,1)$, and $\ell := \beta^{\bar{\imath}}$, where $\bar{\imath}$ is the smallest nonnegative integer i such that

$$f(x^t + \beta^i d^t) - f(x^t) \leq \alpha\beta^i \nabla f(x^t)^\top d^t. \qquad (13)$$

There exists a finite integer such that (13) is satisfied for any search direction $\overline{d}^t := \overline{y}^t - x^t$ satisfying (11), by the descent property and Taylor's formula (see [45, Lemma 2.24.b]).

In the case where $u \neq 0$, however, the situation becomes quite different, since $T := f + u$ is nondifferentiable. Simply replacing $\nabla f(x^t)^\top d^t$ with $T'(x^t;d^t)$ does not work. We can however use an overestimate of the predicted decrease $T'(x^t;d^t)$. Let $\alpha, \beta \in (0,1)$, and $\ell := \beta^{\bar{\imath}}$, where $\bar{\imath}$ is the smallest nonnegative integer i such that

$$T(x^t + \beta^i d^t) - T(x^t) \leq \alpha\beta^i [\nabla\varphi^t(x^t) - \nabla\varphi^t(y^t)]^\top d^t,$$

where now y^t necessarily is an exact solution to (3), and φ^t must further be strictly convex. We note that $T'(x^t;d^t) \leq [\nabla\varphi^t(x^t) - \nabla\varphi^t(y^t)]^\top d^t$ indeed holds, with equality in the case where $u = 0$ and $X = \mathbf{R}^n$ (see [45, Remark 2.28]).

To develop still simpler steplength rules, we further assume that ∇f is Lipschitz continuous, that is, that for $x, y \in \operatorname{dom} u \cap X$,

$$\|\nabla f(x) - \nabla f(y)\| \leq M_{\nabla f} \|x - y\|,$$

for some $M_{\nabla f} > 0$. The Lipschitz continuity assumption implies that for every $\ell \in [0,1]$,

$$\begin{aligned} &T(x^t + \ell d^t) - T(x^t) \\ &\leq \ell \left[\nabla\varphi^t(x^t) - \nabla\varphi^t(y^t)\right]^\top d^t \\ &\quad + \frac{M_{\nabla f}}{2}\ell^2 \left\|d^t\right\|^2; \end{aligned}$$

adding a strong convexity assumption on φ^t yields that

$$T(x^t + \ell d^t) - T(x^t) \leq \ell\left(-m_{\varphi^t} + \frac{M_{\nabla f}\ell}{2}\right)\left\|d^t\right\|^2.$$

This inequality can be used to validate the *relaxation step*, which takes

$$\ell_t \in \left(0, \frac{2m_{\varphi^t}}{M_{\nabla f}}\right) \cap [0,1], \qquad (14)$$

and the *divergent series steplength rule*,

$$[0,1] \supset \{\ell_t\} \to 0, \quad \sum_{t=0}^{\infty} \ell_t = \infty. \qquad (15)$$

In the case of (14), descent is guaranteed in each step, while in the case of (15), descent is guaranteed after a finite number of iterations.

Convergence Properties. Convergence of the CA algorithm can be established under many combinations of

i) the properties of the original problem mappings;

ii) the choice of forms and convexity properties of the cost approximating mappings;

iii) the choice of accuracy in the computations of the CA subproblem solutions;

iv) the choice of merit function; and

v) the choice of steplength rule.

A subset of the possible results is found in [45, Chapt. 5–9]. Evident from these results is that convergence relies on reaching a critical mass in the properties of the problem and algorithm, and that, given that this critical mass is reached, there

is a very large freedom-of-choice how this mass is distributed. So, for example, weaker properties in the monotonicity of the subproblem must be compensated both by stronger coercivity conditions on the merit function and by the use of more accurate subproblem solutions and steplength rules.

Decomposition CA Algorithms. Assume that $\operatorname{dom} u \cap X$ is a *Cartesian product set*, that is, for some finite index set $\mathcal{C}$ and positive integers n_i with $\sum_{i\in\mathcal{C}} n_i = n$,

$$X = \prod_{i\in\mathcal{C}} X_i, \quad X_i \subseteq \mathbf{R}^{n_i};$$

$$u(x) = \sum_{i\in\mathcal{C}} u_i(x_i), \quad u_i\colon \mathbf{R}^{n_i} \to \mathbf{R}\cup\{+\infty\}.$$

Such problems arise in applications of equilibrium programming, for example in traffic ([41]) and Nash equilibrium problems ([21]); of course, box constrained and unconstrained problems fit into this framework as well.

The main advantage of this problem structure is that one can devise several decomposition versions of the CA algorithm, wherein components of the original problem are updated upon in parallel or sequentially, independently of each other. With the right computer environment at hand, this can mean a dramatic increase in computing efficiency. We will look at three computing models for decomposition CA algorithm, and compare their convergence characteristics. In all three cases, decomposition is achieved by choosing the cost approximating mapping separable with respect to the partition of $\mathbf{R}^n$ defined by $\mathcal{C}$:

$$\Phi(x)^\top = [\Phi_1(x_1)^\top, \ldots, \Phi_{|\mathcal{C}|}(x_{|\mathcal{C}|})^\top]. \tag{16}$$

The individual subproblems, given x, then are to find y_i, $i \in \mathcal{C}$, such that

$$\Phi_i(y_i) + \partial u_i(y_i) + N_{X_i}(y_i) + F_i(x) - \Phi_i(x_i) \ni 0^{n_i};$$

if $\Phi_i \equiv \nabla\varphi_i$ for some convex function $\varphi_i\colon \operatorname{dom} u_i \cap X_i \to \mathbf{R}$ in C^1 on $\operatorname{dom} u_i \cap X_i$, then this is the optimality conditions for

$$\min_{y_i\in X_i} T_{\varphi_i}(y_i)$$
$$:= \varphi_i(y_i) + u_i(y_i) + [F_i(x) - \nabla\varphi_i(x_i)]^\top y_i.$$

Sequential Decomposition. The *sequential CA algorithm* proceeds as follows. Given an iterate $x^t \in \operatorname{dom} u \cap X$ at iteration t, choose an index $i_t \in \mathcal{C}$ and a cost approximating mapping $\Phi_{i_t}^t$, and solve the problem of finding $y_{i_t}^t \in \mathbf{R}^{n_{i_t}}$ such that $(i = i_t)$

$$\Phi_i^t(y_i^t) + \partial u_i(y_i^t) + N_{X_i}(y_i^t) + F_i(x^t) - \Phi_i^t(x_i^t) \ni 0^{n_i}.$$

Let $y_j^t := x_j^t$ for all $j \in \mathcal{C}\setminus\{i_t\}$ and $d^t := y^t - x^t$. The next iterate, x^{t+1}, is then defined by $x^{t+1} := x^t + \ell_t d^t$, that is,

$$x_j^{t+1} := \begin{cases} x_j^t + \ell_t(y_j^t - x_j^t), & j = i_t, \\ x_j^t, & j \neq i_t, \end{cases}$$

for some value of ℓ_t such that $x_{i_t}^t + \ell_t(y_{i_t}^t - x_{i_t}^t) \in \operatorname{dom} u_{i_t} \cap X_{i_t}$ and the value of a merit function ψ is reduced sufficiently.

Assume that F is the gradient of a function $f\colon \operatorname{dom} u \cap X \to \mathbf{R}$. Let the sequence $\{i_t\}$ be chosen according to the *cyclic rule*, that is, in iteration t,

$$i_t := t \pmod{|\mathcal{C}|} + 1.$$

Choose the cost approximating mapping $(i = i_t)$

$$y_i \mapsto \Phi_i^t(y_i) := \nabla_i f(x_{\neq i}^t, y_i), \quad y_i \in \operatorname{dom} u_i \cap X_i.$$

Note that this mapping is monotone whenever f is convex in x_i. Since $\Phi_i^t(x_i^t) = \nabla_i f(x^t)$, the CA subproblem is equivalent (under this convexity assumption) to finding

$$y_i^t \in \arg\min_{y_i\in X_i} \{f(x_{\neq i}^t, y_i) + u_i(y_i)\}.$$

An exact line search would produce $\ell_t := 1$, since y_i^t minimizes $f(x_{\neq i}, \cdot) + u_i$ over $\operatorname{dom} u_i \cap X_i$ (the remaining components of x kept fixed), and so $x_i^{t+1} := y_i^t$. The iteration described is that of the classic *Gauss–Seidel algorithm* ([35]) (also known as the *relaxation algorithm*, the *coordinate descent method*, and the *method of successive displacements*), originally proposed for the solution of unconstrained problems. The Gauss–Seidel algorithm is hence a special case of the sequential CA algorithm.

In order to compare the three decomposition approaches, we last provide the steplength requirement in the relaxation steplength rule (cf. (14)). The following interval is valid under the assump-

tions that for each $i \in \mathcal{C}$, $\nabla_i f$ is Lipschitz continuous on $\operatorname{dom} u_i \cap X_i$ and each mapping Φ_i^t is strongly monotone:

$$\ell_{i,t} \in \left(0, \frac{2m_{\Phi_i^t}}{M_{\nabla_i f}}\right) \cap [0,1]. \tag{17}$$

Synchronized Parallel Decomposition. The synchronized parallel CA algorithm is identical to the original scheme, where the CA subproblems are constructed to match the separability structure in the constraints.

We presume the existence of a multiprocessor powerful enough to solve the $|\mathcal{C}|$ CA subproblems in parallel. (If fewer than $|\mathcal{C}|$ processors are available, then either some of the subproblems are solved in sequence or, if possible, the number of components is decreased; in either case, the convergence analysis will be the same, with the exception that the value of $|\mathcal{C}|$ may change.)

In the sequential decomposition CA algorithm, the steplengths are chosen individually for the different variable components, whereas the original CA algorithm uses a uniform steplength, ℓ_t. If the relative scaling of the variable components is poor, in the sense that F or u changes disproportionally to unit changes in the different variables x_i, $i \in \mathcal{C}$, then this ill-conditioning may result in a poor performance of the parallel algorithm. Being forced to use the same steplength in all the components can also have an unwanted effect due to the fact that the values of some variable components are close to their optimal ones while others may be far from optimal, in which case one might for example wish to use longer steps for the latter components. These two factors lead us to introduce the possibility to scale the component directions in the synchronized parallel CA algorithm. We stress that such effects cannot in general be accommodated into the original algorithm through a scaling of the mappings Φ_i^t. The scaling factors $s_{i,t}$ introduced are assumed to satisfy

$$0 < \underline{s}_i \leq s_{i,t} \leq 1, \quad i \in \mathcal{C}.$$

Note that the upper bound of one is without any loss of generality.

Assume that F is the gradient of a function $f\colon \operatorname{dom} u \cap X \to \mathbf{R}$. In the parallel algorithm, choose the cost approximating mapping of the form (16), where for each $i \in \mathcal{C}$,

$$y_i \mapsto \Phi_i^t(y_i) := \nabla_i f(x_{\neq i}^t, y_i),$$
$$y_i \in \operatorname{dom} u_i \cap X_i.$$

This mapping is monotone on $\operatorname{dom} u \cap X$ whenever f is convex in each component x_i. Since $\Phi_i^t(x_i^t) = \nabla_i f(x^t)$, $i \in \mathcal{C}$, it follows that the CA subproblem is equivalent (under the above convexity assumption on f) to finding

$$y_i^t \in \arg\min_{y_i \in X_i} \{f(x_{\neq i}^t, y_i) + u_i(y_i)\}.$$

Choosing $\ell_t := 1$ and $s_{i,t} := 1$, $i \in \mathcal{C}$, yields $x^{t+1} := y^t$, and the resulting iteration is that of the *Jacobi algorithm* [35], [8] (also known as the *method of simultaneous displacements*). The Jacobi algorithm, which was originally proposed for the solution of systems of equations, is therefore a parallel CA algorithm where the cost approximating mapping is (18) and unit steps are taken.

The admissible step in component i is $\ell s_{i,t} \in [0,1]$, where

$$\ell \in \left(0, \min_{i \in \mathcal{C}} \left\{ \frac{2m_{\Phi_i^t}}{s_{i,t} M_{\nabla f}} \right\}\right). \tag{18}$$

The maximal step is clearly smaller than in the sequential approach. To this conclusion contributes both the minimum operation and that $M_{\nabla_i f} \leq M_{\nabla f}$; both of these requirements are introduced here because the update is made over all variable components simultaneously. (An intuitive explanation is that the sequential algorithm utilizes more recent information when it constructs the subproblems.) One may therefore expect the sequential algorithm to converge to a solution with a given accuracy in less iterations, although the parallel algorithm may be more efficient in terms of solution time; the scaling introduced by $s_{i,t}$ may also improve the performance of the parallel algorithm to some extent.

Although the parallel version of the algorithm may speed-up the practical convergence rate compared to the sequential one, the need for synchronization in carrying out the updating step will generally deteriorate performance, since faster processors must wait for slower ones. In the next section, we therefore introduce an asynchronous version of the parallel algorithm, in which processors do not wait to receive the latest information available.

Asynchronous Parallel Decomposition. In the algorithms considered in this Section, the synchronization step among the processors is removed. Because the speed of computations and communications can vary among the processors, and communication delays can be substantial, processors will perform the calculations out of phase with each other. Thus, the advantage of reduced synchronization is paid for by an increase in interprocessor communications, the use of outdated information, and a more difficult convergence detection (see [8]). (Certainly, the convergence analysis also becomes more complicated.) Recent numerical experiments indicate, however, that the introduction of such *asynchronous computations* can substantially enhance the efficiency of parallel iterative methods (e.g., [15], [6], [11]).

The model of partial asynchronism that we use is as follows. For each processor (or, variable component) $i \in \mathcal{C}$, we introduce

a) initial conditions, $x_i(t) := x_i^0 \in X_i$, for all $t \leq 0$;

b) a set $\mathcal{T}^i$ of times at which x_i is updated; and

c) a variable $\tau_j^i(t)$ for each $j \in \mathcal{C}$ and $t \in \mathcal{T}^i$, denoting the time at which the value of x_j used by processor i at time t is generated by processor j, satisfying $0 \leq \tau_j^i(t) \leq t$ for all $j \in \mathcal{C}$ and $t \geq 0$.

We note that the sequential CA algorithm and the synchronized parallel CA algorithm can both be expressed as asynchronous algorithms: the cyclic sequential algorithm model is obtained from the choices $\mathcal{T}^i := \cup_{k \geq 0}\{|\mathcal{C}|k+i-1\}$ and $\tau_j^i(t) := t$, while the synchronous parallel model is obtained by choosing $\mathcal{T}^i := \{1, 2, \ldots\}$ and $\tau_j^i(t) := t$, for all i, j and t.

The communication delay from processor j to processor i at time t is $t - \tau_j^i(t)$. The convergence of the *partially asynchronous parallel decomposition CA algorithm* is based on the assumption that this delay is upper bounded: there exists a positive integer P such that

i) for every $i \in \mathcal{C}$ and $t \geq 0$, at least one element of $\{t, \ldots, t+P-1\}$ belongs to $\mathcal{T}^i$;

ii) $0 \leq t - \tau_j^i(t) \leq P-1$ holds for all $i, j \in \mathcal{C}$ and all $t \geq 0$; and

iii) $\tau_i^i(t) = t$ holds for all $i \in \mathcal{C}$ and all $t \geq 0$.

In short, parts i) and ii) of the assumption state that no processor waits for an arbitrarily long time to compute a subproblem solution or to receive a message from another processor. (Note that a synchronized model satisfies $P = 1$.) Part iii) of the assumption states that processor i always uses the most recent value of its own component x_i of x, and is in [58] referred to as a *computational nonredundancy condition*. This condition holds in general when no variable component is updated simultaneously by more than one processor, as, for example, in message passing systems. For further discussions on the assumptions, we refer the reader to [8], [57]; we only remark that they are easily enforced in practical implementations.

The iterate $x(t)$ is defined by the vector of $x_i(t)$, $i \in \mathcal{C}$. At a given time t, processor i has knowledge of a possibly outdated version of $x(t)$; we let

$$x^i(t)^\top := \left[x_1(\tau_1^i(t))^\top, \ldots, x_{|\mathcal{C}|}(\tau_{|\mathcal{C}|}^i(t))^\top\right]$$

denote this vector. (Note that iii) above implies the relation $x_i^i(t) := x_i(\tau_i^i(t)) = x_i(t)$.)

To describe the (partially) asynchronous parallel CA algorithm, processor i updates $x_i(t)$ according to

$$x_i(t+1) := x_i(t) + \ell s_i(y_i(t) - x_i^i(t)), \quad t \in \mathcal{T}^i,$$

where $y_i(t)$ solves the CA subproblem defined at $x^i(t)$, and $s_i \in (0, 1]$ is a scaling parameter. (We define $d_i(t) := y_i(t) - x_i^i(t)$ to be zero at each $t \notin \mathcal{T}^i$.)

The admissible steplength for $i \in \mathcal{C}$ is $\ell s_i \in [0, 1]$, where

$$\ell \in \left(0, \frac{2 \min_{i \in \mathcal{C}}\{m_{\Phi_i}/s_i\}}{M_{\nabla f}[1 + (|\mathcal{C}|+1)P]}\right). \tag{19}$$

If further for some $M \geq 0$ and every $i \in \mathcal{C}$, all vectors x, y in $\operatorname{dom} u \cap X$ with $x_i = y_i$ satisfy

$$\|\nabla_i f(x) - \nabla_i f(y)\| \leq M \|x - y\|, \tag{20}$$

then, in the above result, the steplength restrictions are adjusted to

$$\ell \in \left(0, \frac{2 \min_{i \in \mathcal{C}}\{m_{\Phi_i}/s_i\}}{M_{\nabla f} + (|\mathcal{C}|+1)MP}\right).$$

(We interpret the property (20) as a quantitative measure of the coupling between the variables.)

Most important to note is that the upper bound on ℓ is (essentially) inversely proportional to the maximal allowed asynchronism P; this is very intuitive, since if processors take longer steps then they should exchange information more often. Conversely, the more outdated the information is, the less reliable it is, hence the shorter step.

The relations among the steplengths in the three approaches (cf. (17), (18), and (19)) quantify the intuitive result that utilizing an increasing degree of parallelism and asynchronism results in a decreasing quality of the step directions, due to the usage of more outdated information; subsequently, smaller steplengths must be used. More detailed discussions about this topic is found in [45, Sect. 8.7.2].

See also: **Dynamic traffic networks**.

References

[1] AUCHMUTY, G.: 'Variational principles for variational inequalities', *Numer. Funct. Anal. Optim.* **10** (1989), 863–874.

[2] AUSLENDER, A.: *Optimisation: Méthodes numériques*, Masson, 1976.

[3] AUSLENDER, A., CROUZEIX, J.P., AND FEDIT, P.: 'Penalty-proximal methods in convex programming', *J. Optim. Th. Appl.* **55** (1987), 1–21.

[4] BAZARAA, M.S., SHERALI, H.D., AND SHETTY, C.M.: *Nonlinear programming: Theory and algorithms*, second ed., Wiley, 1993.

[5] BERTSEKAS, D.P.: *Nonlinear programming*, second ed., Athena Sci., 1999.

[6] BERTSEKAS, D.P., AND CASTANON, D.A.: 'Parallel synchronous and asynchronous implementations of the auction algorithm', *Parallel Comput.* **17** (1991), 707–732.

[7] BERTSEKAS, D.P., AND TSENG, P.: 'Partial proximal minimization algorithms for convex programming', *SIAM J. Optim.* **4** (1994), 551–572.

[8] BERTSEKAS, D.P., AND TSITSIKLIS, J.N.: *Parallel and distributed computation: Numerical methods*, Prentice-Hall, 1989.

[9] BREGMAN, L.M.: 'A relaxation method of finding a common point of convex sets and its application to problems of optimization', *Soviet Math. Dokl.* **7** (1966), 1578–1581.

[10] BROWDER, F.E.: 'Existence and approximation of solutions of nonlinear variational inequalities', *Proc. Nat. Acad. Sci. USA* **56** (1966), 1080–1086.

[11] CHAJAKIS, E.D., AND ZENIOS, S.A.: 'Synchronous and asynchronous implementations of relaxation algorithms for nonlinear network optimization', *Parallel Comput.* **17** (1991), 873–894.

[12] COHEN, G.: 'Optimization by decomposition and coordination: A unified approach', *IEEE Trans. Autom. Control* **AC-23** (1978), 222–232.

[13] COMINETTI, R.: 'Coupling the proximal point algorithm with approximation methods', *J. Optim. Th. Appl.* **95** (1997), 581–600.

[14] ECKSTEIN, J.: 'Nonlinear proximal point algorithms using Bregman functions, with applications to convex programming', *Math. Oper. Res.* **18** (1993), 202–226.

[15] EL BAZ, D.: 'A computational experience with distributed asynchronous iterative methods for convex network flow problems': *Proc. 28th IEEE Conf. Decision and Control*, 1989, pp. 590–591.

[16] FLETCHER, R.: *Practical methods of optimization*, second ed., Wiley, 1987.

[17] FRANK, M., AND WOLFE, P.: 'An algorithm for quadratic programming', *Naval Res. Logist. Quart.* **3** (1956), 95–110.

[18] FUKUSHIMA, M.: 'Equivalent differentiable optimization problems and descent methods for asymmetric variational inequality problems', *Math. Program.* **53** (1992), 99–110.

[19] GOLDSTEIN, A.A.: 'Convex programming in Hilbert space', *Bull. Amer. Math. Soc.* **70** (1964), 709–710.

[20] HA, C.D.: 'A generalization of the proximal point algorithm', *SIAM J. Control Optim.* **28** (1990), 503–512.

[21] HARKER, P.T., AND PANG, J.-S.: 'Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications', *Math. Program.* **48** (1990), 161–220.

[22] KIWIEL, K.C.: 'Generalized Bregman projections in convex feasibility problems', *J. Optim. Th. Appl.* **96** (1998), 139–157.

[23] KIWIEL, K.C.: 'Subgradient method with entropic projections for convex nondifferentiable minimization', *J. Optim. Th. Appl.* **96** (1998), 159–173.

[24] KORPELEVICH, G.M.: 'The extragradient method for finding saddle points and other problems', *Matekon* **13** (1977), 35–49.

[25] LARSSON, T., AND MIGDALAS, A.: 'An algorithm for nonlinear programs over Cartesian product sets', *Optim.* **21** (1990), 535–542.

[26] LARSSON, T., AND PATRIKSSON, M.: 'A class of gap functions for variational inequalities', *Math. Program.* **64** (1994), 53–79.

[27] LEVITIN, E.S., AND POLYAK, B.T.: 'Constrained minimization methods', *USSR Comput. Math. Math. Phys.* **6** (1966), 1–50.

[28] LUO, Z.-Q., AND TSENG, P.: 'On the convergence of a matrix splitting algorithm for the symmetric monotone linear complementarity problem', *SIAM J. Control Optim.* **29** (1991), 1037–1060.

[29] MANGASARIAN, O.L.: 'Convergence of iterates of an inexact matrix splitting algorithm for the symmetric monotone linear complementarity problem', *SIAM J. Optim.* **1** (1991), 114–122.

[30] MARCOTTE, P., AND DUSSAULT, J.-P.: 'A sequential linear programming algorithm for solving monotone variational inequalities', *SIAM J. Control Optim.* **27** (1989), 1260–1278.

[31] MARCOTTE, P., AND ZHU, D.: 'Global convergence of descent processes for solving non strictly monotone variational inequalities', *Comput. Optim. Appl.* **4** (1995), 127–138.

[32] MARTINET, B.: 'Détermination approchée d'un point fixe d'une application pseudo-contractante', *C.R. Hebdom. Séances de l'Acad. Sci. (Paris), Sér. A* **274** (1972), 163–165.

[33] MINTY, G.J.: 'Monotone (nonlinear) operators in Hilbert space', *Duke Math. J.* **29** (1962), 341–346.

[34] MOREAU, J.-J.: 'Proximité et dualité dans un espace Hilbertien', *Bull. Soc. Math. France* **93** (1965), 273–299.

[35] ORTEGA, J.M., AND RHEINBOLDT, W.C.: *Iterative solution of nonlinear equations in several variables*, Acad. Press, 1970.

[36] PANG, J.-S.: 'On the convergence of a basic iterative method for the implicit complementarity problem', *J. Optim. Th. Appl.* **37** (1982), 149–162.

[37] PATRIKSSON, M.: 'Partial linearization methods in nonlinear programming', *J. Optim. Th. Appl.* **78** (1993), 227–246.

[38] PATRIKSSON, M.: 'A unified description of iterative algorithms for traffic equilibria', *Europ. J. Oper. Res.* **71** (1993), 154–176.

[39] PATRIKSSON, M.: 'A unified framework of descent algorithms for nonlinear programs and variational inequalities', *PhD Thesis Dept. Math. Linköping Inst. Techn.* (1993).

[40] PATRIKSSON, M.: 'On the convergence of descent methods for monotone variational inequalities', *Oper. Res. Lett.* **16** (1994), 265–269.

[41] PATRIKSSON, M.: *The traffic assignment problem – Models and methods*, Topics in Transportation. VSP, 1994.

[42] PATRIKSSON, M.: 'Merit functions and descent algorithms for a class of variational inequality problems', *Optim.* **41** (1997), 37–55.

[43] PATRIKSSON, M.: 'Cost approximation: A unified framework of descent algorithms for nonlinear programs', *SIAM J. Optim.* **8** (1998), 561–582.

[44] PATRIKSSON, M.: 'Decomposition methods for differentiable optimization problems over Cartesian product sets', *Comput. Optim. Appl.* **9** (1998), 5–42.

[45] PATRIKSSON, M.: *Nonlinear programming and variational inequality problems: A unified approach*, Vol. 23 of *Applied Optim.*, Kluwer Acad. Publ., 1998.

[46] PATRIKSSON, M.: 'Cost approximation algorithms with nonmonotone line searches for a general class of nonlinear programs', *Optim.* **44** (1999), 199–217.

[47] POLYAK, B.T.: 'Gradient methods for the minimisation of functionals', *USSR Comput. Math. Math. Phys.* **3** (1963), 864–878.

[48] POLYAK, B.T.: 'A general method of solving extremum problems', *Soviet Math. Dokl.* **8** (1967), 593–597.

[49] POLYAK, B.T.: *Introduction to optimization*, Optim. Software, 1987.

[50] PSHENICHNY, B.N., AND DANILIN, YU.M.: *Numerical methods in extremal problems*, MIR, 1978.

[51] ROCKAFELLAR, R.T.: 'Augmented Lagrangians and applications of the proximal point algorithm in convex programming', *Math. Oper. Res.* **1** (1976), 97–116.

[52] ROCKAFELLAR, R.T.: 'Monotone operators and the proximal point algorithm', *SIAM J. Control Optim.* **14** (1976), 877–898.

[53] SHOR, N.Z.: *Minimization methods for non-differentiable functions*, Springer, 1985.

[54] TEBOULLE, M.: 'Convergence of proximal-like algorithms', *SIAM J. Optim.* **7** (1997), 1069–1083.

[55] TIKHONOV, A.N.: 'Solution of incorrectly formulated problems and the regularization method', *Soviet Math. Dokl.* **4** (1963), 1035–1038.

[56] TIKHONOV, A.N., AND ARSENIN, V.YA.: *Solutions of ill-posed problems*, Wiley, 1977. (Translated from the Russian.)

[57] TSENG, P., BERTSEKAS, D.P., AND TSITSIKLIS, J.N.: 'Partially asynchronous, parallel algorithms for network flow and other problems', *SIAM J. Control Optim.* **28** (1990), 678–710.

[58] ÜRESIN, A., AND DUBOIS, M.: 'Asynchronous iterative algorithms: Models and convergence', in L. KRONSJÖ AND D. SHUMSHERUDDIN (eds.): *Advances in Parallel Algorithms*, Blackwell, 1992, pp. 302–342.

[59] WU, J.H., FLORIAN, M., AND MARCOTTE, P.: 'A general descent framework for the monotone variational inequality problem', *Math. Program.* **61** (1993), 281–300.

[60] ZHU, D.L., AND MARCOTTE, P.: 'Modified descent methods for solving the monotone variational inequality problem', *Oper. Res. Lett.* **14** (1993), 111–120.

[61] ZHU, D.L., AND MARCOTTE, P.: 'An extended descent framework for variational inequalities', *J. Optim. Th. Appl.* **80** (1994), 349–366.

[62] ZUHOVICKIĬ, S.I., POLYAK, R.A., AND PRIMAK, M.E.: 'Two methods of search for equilibrium points of n-person concave games', *Soviet Math. Dokl.* **10** (1969), 279–282.

Michael Patriksson
Dept. Math. Chalmers Univ. Technol.
SE-412 96 Göteborg, Sweden
E-mail address: mipat@math.chalmers.se

MSC2000: 90C30
Key words and phrases: linearization methods, gradient projection, quasi-Newton, Frank–Wolfe, sequential quadratic programming, operator splitting, proximal point, Levenberg–Marquardt, auxiliary problem principle, subgradient optimization, variational inequality problem, merit

function, Cartesian product, Gauss–Seidel, Jacobi, asynchronous computation.

CRISS-CROSS PIVOTING RULES, *criss-cross*

From the early days of linear optimization (LO) (or linear programming), many people have been looking for a *pivot algorithm* that avoids the *two-phase procedure* needed in the simplex method when solving the general LO problem in standard primal form

$$\min\left\{c^\top x\colon Ax = b,\ x \geq 0\right\},$$

and its dual

$$\max\left\{b^\top y\colon A^\top y \leq c\right\}.$$

Such a method was assumed to rely on the intrinsic symmetry behind the *primal and dual problems* (i.e. it hoped to be *selfdual*), and it should be able to start with any *basic solution*.

There were several attempts made to relax the feasibility requirement in the simplex method. It is important to mention Dantzig's [7] parametric selfdual simplex algorithm. This algorithm can be interpreted as *Lemke's algorithm* [22] for the corresponding linear complementarity problem (cf. **Linear complementarity problem**) [23]. In the 1960s people realized that pivot sequences through possibly infeasible basic solutions might result in significantly shorter paths to the optimum. Moreover a selfdual one phase procedure was expected to make linear programming more easily accessible for broader public. Probably these advantages stimulated the introduction of the *criss-cross method* by S. Ziont [39], [40].

Ziont's Criss-Cross Method. Assuming that the reader is familiar with both the primal and dual simplex methods, Ziont's criss-cross method can easily be explained.

- It can be initialized by any, possibly both primal and dual infeasible *basis*.

  If the basis is optimal, we are done.

  If the basis is not *optimal*, then there are some primal or dual infeasible variables. One might choose any of these.

  It is advised to choose once a primal and then a dual infeasible variable, if possible.

- If the selected variable is dual infeasible, then it enters the basis and the leaving variable is chosen among the primal feasible variables in such a way that primal feasibility of the currently primal feasible variables is preserved.

  If no such basis exchange is possible another infeasible variable is selected.

- If the selected variable is primal infeasible, then it leaves the basis and the entering variable is chosen among the dual feasible variables in such a way that dual feasibility of the currently dual feasible variables is preserved.

  If no such basis exchange is possible another infeasible variable is selected.

If the current basis is infeasible, but none of the infeasible variables allows a pivot fulfilling the above requirements then it is proved that the LO problem has no optimal solution.

Once a primal or dual feasible solution is reached then Ziont's method reduces to the primal or dual simplex method, respectively.

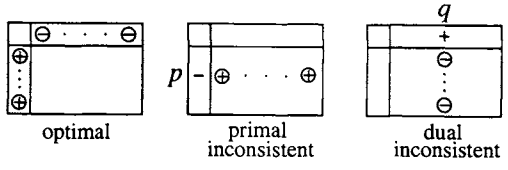
One attractive character of Ziont's criss-cross method is primal-dual symmetry (selfduality), and this alone differentiates itself from the simplex method. However it is not clear how one can design a finite version (i.e. a finite pivot rule) of this method. Both lexicographic perturbation and minimal index resolution seem not to be sufficient to prove finiteness in the general case when the initial basis is both primal and dual infeasible. Nevertheless, although not finite, this is the first published criss-cross method in the literature.

The other thread, that lead to finite criss-cross methods, was the intellectual effort to find finite, other than the lexicographic rule [4], [8], variants of the simplex method. These efforts were also stimulated by studying the combinatorial structures behind linear programming. From the early 1970s in several branches of the optimization theory, finitely convergent algorithms were published. In particular A.W. Tucker [32] introduced the *consistent labeling* technique in the Ford–Fulkerson maximal flow algorithm; pivot selection rules based on least-index ordering, such as the Bard-type scheme for the P-matrix linear complementarity problem (K.G. Murty, [24]) and the celebrated least-index rule in linear and oriented matroid programming

(R.G. Bland, [2]). A thorough survey of pivot algorithms can be found in [29].

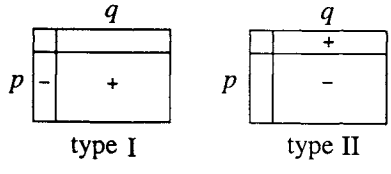
It is remarkable that almost at the same time, in different parts of the world (China, Hungary, USA) essentially the same result was obtained independently by approaching the problem from quite different directions.

Below we will refer to the standard simplex (basis) tableau. A tableau is called *terminal* if it gives a primal and dual optimal solution or evidence of primal or dual infeasibility/inconsistency of the problem. Terminal tableaus have the following sign structure.



Terminal tableaus.

The pivot operations at all known pivot methods, including all variants of the primal and dual simplex method and Ziont's criss-cross method have the following properties. When a primal infeasible variable is selected to leave the basis, the entering variable is selected so that after the pivot both variables involved in the pivot will be primal feasible. Analogously, when a dual infeasible variable is selected to enter the basis, then the leaving variable is selected in such a way that after the pivot both variables involved in the pivot will be dual feasible. Such pivots are called *admissible*. The sign structure of tableaus at an admissible pivot of 'type I' and 'type II' are demonstrated by the following figure.



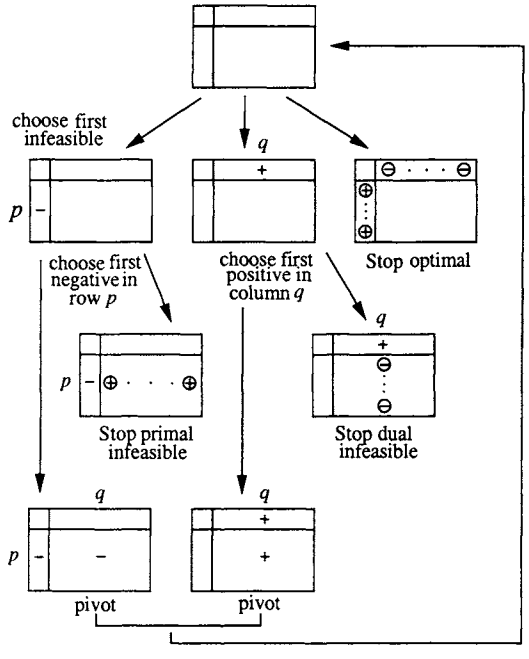
Admissible pivot situations.

Observe that, while dual(primal) simplex pivots preserve dual(primal) feasibility of the basic solution, admissible pivots do not in general. Admissible pivots extract the greedy nature of pivot selection, i.e. 'repair primal/dual infeasibility' of the pivot variables.

The Least-Index Criss-Cross Method. The first finite criss-cross algorithm, which we call the *least-index criss-cross method*, was discovered independently by Y.Y. Chang [4], T. Terlaky [26], [28], [27] and Zh. Wang [34]; further, a strongly related general recursion by D. Jensen [18]. Chang presented the algorithm for positive semidefinite linear complementarity problems, Terlaky for linear optimization, for oriented matroids and with coauthors for QP, LCP and for oriented matroid LCP [19], [16], [9], while Wang primarily for the case of oriented matroids.

The least-index criss-cross method is perhaps the simplest finite pivoting method to LO problems. This criss-cross method is a purely combinatorial pivoting method, it uses admissible pivots and traverses through different (possibly both primal and dual infeasible) bases until the associated basic solution is optimal, or an evidence of primal or dual infeasibility is found.

To ease the understanding a figure is included that shows the scheme of the least-index criss-cross method.



Scheme of the least-index criss-cross method.

Observe the simplicity of the algorithm:

- It can be initiated with any basis.
- No two phases are needed.
- No ratio test is used to preserve feasibility, only the signs of components in a basis

tableau and a prefixed ordering of variables determine the pivot selection.

Several finiteness proofs for the least-index criss-cross method can be found in the literature. The proofs are quite elementary, they are based on the orthogonality of the primal and dual spaces [26], [28], [34], [29], [14]; on recursive argumentation [33], [18], [11] or on lexicographically increasing lists [11], [14].

0	Let an ordering of the variables be fixed. Let $T(B)$ be an arbitrary basis tableau (it can be neither primal nor dual feasible);
1	Let r be the minimal i such that either x_i is primal infeasible or x_i has a negative reduced cost. IF there is no r, THEN stop; the first terminal tableau is obtained, thus $T(B)$ is optimal.
2	IF x_r is primal infeasible THEN let $p := r$; $q := \min\{\ell\colon t_{p\ell} < 0\}$. IF there is no q, THEN stop; the second terminal tableau is obtained, thus the primal problem is infeasible. Go to Step 3. IF x_r is dual infeasible, THEN let $q := r$; $p := \min\{\ell\colon t_{\ell q} > 0\}$. IF there is no q, THEN stop; the third terminal tableau is obtained, thus the dual problem is infeasible. Go to Step 3.
3	Pivot on (p, q). Go to Step 1.

The least-index criss-cross rule.

One of the most important consequences of the finiteness of the least-index criss-cross method is the *strong duality theorem* of linear optimization. This gives probably the simplest algorithmic proof of this fundamental result:

THEOREM 1 (*Strong duality theorem*) Exactly one of the following two cases occurs.

- At least one of the primal problem and the dual problem is infeasible.
- Both problems have an optimal solution and the optimal objective values are equal.

□

Other Interpretations. The least-index criss-cross method can be interpreted as a *recursive algorithm*. This recursive interpretation and the finiteness proof based on it can be derived from the results in [3], [2], [18] and can be found in [33].

Recursive Interpretation.

As performing the least-index criss-cross method at each pivot one can make a note of the larger of the two indices $r = \max\{p, q\}$ that entered or left the basis. In this list, an index must be followed by another larger one before the same index occurs anew.

The recursive interpretation is becoming apparent when one notes that it is based on the observation that the size of the solved subproblem (the subproblem for which a terminal tableau is obtained) is monotonically increasing.

The third interpretation is based on the proof technique developed by J. Edmonds and K. Fukuda [9] and adapted by Fukuda and T. Matsui [11] to the case of the least-index criss-cross method.

Lexicographically Increasing List.

Let u be an binary vector with appropriate dimension, set initially to be the zero vector. In applying the algorithm let $r = \max\{p, q\}$ be the larger of the two indices p that entered or q that left the basis.
At each pivot update u as follows: let $u_r = 1$ and $u_i = 0$, $\forall i < r$. The remaining components of u stay unchanged. Then at each step of the least-index criss-cross method the vector u strictly increases lexicographically, thus the method terminates in a finite number of steps.

Other Finite Criss-Cross Methods. Both the recursive interpretation and the hidden flexibility of pivot selection in the least-index criss-cross method make it possible to develop other finite variants. Such finite criss-cross methods, which do not rely on a fixed minimal index ordering, were developed on the basis of the finite simplex rules presented by S. Zhang [37]. These finite criss-cross rules [38] are as follows.

First-in last-out rule (FILO).

First, choose a primal or dual infeasible variable that has changed its basis-nonbasis status most recently.

Then choose a variable in the selected row or column so that the pivot entry fulfills the sign requirement of the admissible pivot selection and which has changed its basis/nonbasis status most recently.

When more than one candidates occur with the same pivot age then one break tie as you like (e.g. randomly).

This rule can easily be realized by assigning an 'age' vector u to the vector of the variables and using a pivot counter k. Initially we set $k = 0$ and $u = 0$. Increase k by one at each pivot and we set the pivot coordinates of u equal to k. Then the pivot selections are made by choosing the variable with the highest possible u_i value satisfying the sign requirements.

Most Often Selected Variable Rule.

First, choose a primal or dual infeasible variable that has changed its basis-nonbasis status most frequently.

Then choose a variable in the selected row or column so that the pivot entry fulfills the sign requirement of the admissible pivot selection and which has changed its basis/nonbasis status most frequently.

When more than one candidates occur with the same pivot age then one break tie as you like (e.g. randomly).

The most often selected rule can also be realized by assigning another 'age' vector u to the vector of the variables. Initially we set $u = 0$. At each pivot we increase the pivot-variable components of u by one. Then the pivot selections are made by choosing the variable with the highest possible u_i value satisfying the sign requirement.

Exponential and Average Behavior. The worst-case exponential behavior of the least-index criss-cross method was studied by C. Roos [25]. Roos' exponential example is a variant of the cube of V. Klee and G.J. Minty [21]. In this example the starting solution is the origin defined by a feasible basis, the variables are ordered so that the least-index criss-cross method follows a simplex path, i.e. without making any ratio test feasibility of the starting basis is preserved. Another exponential example was presented by Fukuda and Namiki [12] for linear complementarity problems.

Contrary to the clear result on the worst-case behavior, to date not much is known about the expected or average number of pivot steps required by finite criss-cross methods.

Best-Case Analysis of Admissible Pivot Methods. As it was discussed above, and it is the case for many simplex algorithms, the least-index criss-cross method is not a polynomial time algorithm. A question naturally arises: whether there exists a polynomial criss-cross method? Unfortunately no answer to this question is available at this moment. However some weaker variants of this question can be answered positively. The problem is stated as follows: An arbitrary basis is given. What is the shortest admissible pivot path from this given basis to an optimal basis?

For *nondegenerate problems*, [10] shows the existence of such an admissible pivot sequence of length at most m. The nondegeneracy assumption is removed in [15]. This result solves a relaxation of the $d-$step conjecture.

Observe, that we do not know of any such result for feasibility preserving, i.e. simplex algorithms. In fact, the maximum length of feasibility-preserving pivot sequences between two feasible bases is not known to be bounded by a polynomial in the size of the given LO problem.

Generalizations. Finite criss-cross methods were generalized to solve fractional linear optimization problems, to large classes of linear complementarity problems (LCPs; cf. **Linear complementarity problem**) and to oriented matroid programming problems (OMPs).

Fractional Linear Optimization. Fractional linear or, as it is frequently referred to, *hyperbolic pro-*

gramming, can be reformulated as a linear optimization problem. Thus it goes without surprise that the least-index criss-cross method is generalized to this class of optimization problems as well [17].

Linear Complementarity Problems. The largest solvable class of LCPs is the class of LCPs with a *sufficient matrix* [6], [5]. The LCP least-index criss-cross method is a proper generalization of the LO criss-cross method. When the LCP arises from a LO problem, the LO criss-cross method is obtained.

Convex Quadratic Optimization. Convex quadratic optimization problems give an LCP with a bisymmetric coefficient matrix. Because a bisymmetric matrix is semidefinite and semidefinite matrices form a subclass of sufficient matrices, one obtain a finite criss-cross algorithm for convex quadratic optimization problems as well. Such criss-cross algorithms were published e.g. in [20]. The least-index criss-cross method is extremely simple for the *P-matrix* LCP. Starting from an arbitrary complementary basis, here the least-indexed infeasible variable leaves the basis and it is replaced by its complementary pair. This algorithm was originally proposed in [24], and studied in [12]. The general case of sufficient LCPs was treated in [4], [13], [16].

Oriented Matroids. The intense research in the 1970s on *oriented matroids* and oriented matroid programming [2], [9] gave a new insight in pivot algorithms. It became clear that although the simplex method has rich combinatorial structures, some essential results like the finiteness of Bland's least-index simplex rule [2] does not hold in the oriented matroid context. Edmonds and Fukuda [9] showed that it might cycle in the oriented matroid case due to the possibility of nondegenerate cycling which is impossible in the linear case.

The predecessors of finite criss-cross rules are: Bland's recursive algorithm [3], [2], the Edmonds–Fukuda algorithm [9], its variants and generalizations [1], [35], [36], [37]). All these are variants of the simplex method in the linear case, i.e. they preserve the feasibility of the basis, but not in the oriented matroid case. In the case of oriented matroid programming only Todd's finite lexicographic method [30], [31] preserves feasibility of the basis and therefore yields a finite simplex algorithm for oriented matroids.

The least-index criss-cross method is a finite criss-cross method for oriented matroids [28], [34]. A general recursive scheme of finite criss-cross methods is given in [18]. Finite criss-cross rules are also presented for oriented matroid quadratic programming and for oriented matroid linear complementarity problems [19], [13].

See also: **Linear programming; Pivoting algorithms for linear programming generating two paths; Simplicial pivoting algorithms for integer programming; Principal pivoting methods for linear complementarity problems; Lexicographic pivoting rules; Least-index anticycling rules; Probabilistic analysis of simplex algorithms.**

References

[1] BJORNER, A., VERGNAS, M. LAS, STURMFELS, B., WHITE, N., AND ZIEGLER, G.: *Oriented matroids*, Cambridge Univ. Press, 1993.

[2] BLAND, R.G.: 'A combinatorial abstraction of linear programming', *J. Combin. Th. B* **23** (1977), 33–57.

[3] BLAND, R.G.: 'New finite pivoting rules for the simplex method', *Math. Oper. Res.* **2** (1977), 103–107.

[4] CHANG, Y.Y.: 'Least index resolution of degeneracy in linear complementarity problems', *Techn. Report Dept. Oper. Res. Stanford Univ.* **14** (1979).

[5] COTTLE, R., PANG, J.S., AND STONE, R.E.: *The linear complementarity problem*, Acad. Press, 1992.

[6] COTTLE, R.W., PANG, J.-S., AND VENKATESWARAN, V.: 'Sufficient matrices and the linear complementarity problem', *Linear Alg. & Its Appl.* **114/115** (1987), 235–249.

[7] DANTZIG, G.B.: *Linear programming and extensions*, Princeton Univ. Press, 1963.

[8] DANTZIG, G.B., ORDEN, A., AND WOLFE, P.: 'Notes on linear programming: Part I – The generalized simplex method for minimizing a linear form under linear inequality restrictions', *Pacific J. Math.* **5**, no. 2 (1955), 183–195.

[9] FUKUDA, K.: 'Oriented matroid programming', *PhD Thesis Waterloo Univ.* (1982).

[10] FUKUDA, K., LUETHI, H.-J., AND NAMIKI, M.: 'The existence of a short sequence of admissible pivots to an optimal basis in LP and LCP', *ITOR* **4** (1997), 273–284.

[11] FUKUDA, K., AND MATSUI, T.: 'On the finiteness of the criss-cross method', *Europ. J. Oper. Res.* **52** (1991), 119–124.

[12] FUKUDA, K., AND NAMIKI, M.: 'On extremal behaviors of Murty's least index method', *Math. Program.* **64** (1994), 365–370.

[13] FUKUDA, K., AND TERLAKY, T.: 'Linear complementarity and oriented matroids', *J. Oper. Res. Soc. Japan* **35** (1992), 45–61.

[14] FUKUDA, K., AND TERLAKY, T.: 'Criss-cross methods: A fresh view on pivot algorithms': *Math. Program., (B) Lectures on Math. Program., ISMP97*, Vol. 79, 1997, pp. 369–396.

[15] FUKUDA, K., AND TERLAKY, T.: 'On the existence of short admissible pivot sequences for feasibility and linear optimization problems', *Techn. Report Swiss Federal Inst. Technol.* (1999).

[16] HERTOG, D. DEN, ROOS, C., AND TERLAKY, T.: 'The linear complementarity problem, sufficient matrices and the criss-cross method', *Linear Alg. & Its Appl.* **187** (1993), 1–14.

[17] ILLÉS, T., SZIRMAI, Á., AND TERLAKY, T.: 'A finite criss-cross method for hyperbolic programming', *Europ. J. Oper. Res.* **114** (1999), 198–214.

[18] JENSEN, D.: 'Coloring and duality: Combinatorial augmentation methods', *PhD Thesis School OR and IE, Cornell Univ.* (1985).

[19] KLAFSZKY, E., AND TERLAKY, T.: 'Some generalizations of the criss-cross method for the linear complementarity problem of oriented matroids', *Combinatorica* **9** (1989), 189–198.

[20] KLAFSZKY, E., AND TERLAKY, T.: 'Some generalizations of the criss-cross method for quadratic programming', *Math. Oper. Statist. Ser. Optim.* **24** (1992), 127–139.

[21] KLEE, V., AND MINTY, G.J.: 'How good is the simplex algorithm?', in O. SHISHA (ed.): *Inequalities-III*, Acad. Press, 1972, pp. 1159–175.

[22] LEMKE, C.E.: 'On complementary pivot theory', in G.B. DANTZIG AND A.F. VEINOTT (eds.): *Mathematics of the Decision Sci. Part I.*, Lect. Appl. Math. 11, Amer. Math. Soc., 1968, pp. 95–114.

[23] LUSTIG, I.: 'The equivalence of Dantzig's self-dual parametric algorithm for linear programs to Lemke's algorithm for linear complementarity problems applied to linear programming', *SOL Techn. Report Dept. Oper. Res. Stanford Univ.* **87**, no. 4 (1987).

[24] MURTY, K.G.: 'A note on a Bard type scheme for solving the complementarity problem', *Opsearch* **11**, no. 2–3 (1974), 123–130.

[25] ROOS, C.: 'An exponential example for Terlaky's pivoting rule for the criss-cross simplex method', *Math. Program.* **46** (1990), 78–94.

[26] TERLAKY, T.: 'Egy új, véges criss-cross módszer lineáris programozási feladatok megoldására', *Alkalmazott Mat. Lapok* **10** (1984), 289–296, English title: A new, finite criss-cross method for solving linear programming problems. (In Hungarian.)

[27] TERLAKY, T.: 'A convergent criss-cross method', *Math. Oper. Statist. Ser. Optim.* **16**, no. 5 (1985), 683–690.

[28] TERLAKY, T.: 'A finite criss-cross method for oriented matroids', *J. Combin. Th. B* **42**, no. 3 (1987), 319–327.

[29] TERLAKY, T., AND ZHANG, S.: 'Pivot rules for linear programming: A survey on recent theoretical developments', *Ann. Oper. Res.* **46** (1993), 203–233.

[30] TODD, M.J.: 'Complementarity in oriented matroids', *SIAM J. Alg. Discrete Meth.* **5** (1984), 467–485.

[31] TODD, M.J.: 'Linear and quadratic programming in oriented matroids', *J. Combin. Th. B* **39** (1985), 105–133.

[32] TUCKER, A.: 'A note on convergence of the Ford–Fulkerson flow algorithm', *Math. Oper. Res.* **2**, no. 2 (1977), 143–144.

[33] VALIAHO, H.: 'A new proof of the finiteness of the criss-cross method', *Math. Oper. Statist. Ser. Optim.* **25** (1992), 391–400.

[34] WANG, ZH.: 'A conformal elimination free algorithm for oriented matroid programming', *Chinese Ann. Math.* **8**, no. B1 (1985).

[35] WANG, ZH.: 'A modified version of the Edmonds–Fukuda algorithm for LP in the general form', *Asia-Pacific J. Oper. Res.* **8**, no. 1 (1991).

[36] WANG, ZH.: 'A general deterministic pivot method for oriented matroid programming', *Chinese Ann. Math. B* **13**, no. 2 (1992).

[37] ZHANG, S.: 'On anti-cycling pivoting rules for the simplex method', *Oper. Res. Lett.* **10** (1991), 189–192.

[38] ZHANG, S.: 'New variants of finite criss-cross pivot algorithms for linear programming', *Europ. J. Oper. Res.* **116** (1999), 607–614.

[39] ZIONTS, S.: 'The criss-cross method for solving linear programming problems', *Managem. Sci.* **15**, no. 7 (1969), 426–445.

[40] ZIONTS, S.: 'Some empirical test of the criss-cross method', *Managem. Sci.* **19** (1972), 406–410.

Tamás Terlaky
Dept. Comput. & Software McMaster Univ.
1280 Main Street West
Hamilton, Ontario, Canada L8S 4L7
E-mail address: Terlaky@McMaster.ca

MSC2000: 90C05, 90C33, 90C20, 05B35, 65K05
Key words and phrases: pivot rules, criss-cross method, cycling, recursion, linear optimization, oriented matroids.

CUTTING PLANE METHODS FOR GLOBAL OPTIMIZATION

In solving global and combinatorial optimization problems cuts are used as a device to discard portions of the feasible set where it is known that no optimal solution can be found. Specifically, given the optimization problem

$$\min\{f(x)\colon x \in D \subset \mathbf{R}^n\}, \qquad (1)$$

if x^0 is an unfit solution and there exists a function $l(x)$ satisfying $l(x^0) > 0$, while $l(x) \leq 0$ for every optimal solution x, then by adding the inequality $l(x) \leq 0$ to the constraint set we exclude x^0 without excluding any optimal solution. The inequality $l(x) \leq 0$ is called a *valid cut*, or briefly, a *cut*. Most often the function $l(x)$ is affine: the cut is then said to be linear, and the hyperplane $l(x) = 0$ is called a cutting plane. However, nonlinear cuts have proved to be useful, too, for a wide class of problems.

Cuts may be employed in different contexts: outer and inner approximation (conjunctive cuts), branch and bound (disjunctive cuts), or in combined form.

Outer Approximation. Let $\Omega \subset \mathbf{R}^n$ be the set of optimal solutions of problem (2). Suppose there exists a family $\mathcal{P}$ of polytopes $P \supset \Omega$ such that for each $P \in \mathcal{P}$ a *distinguished point* $z(P) \in P$ (conceived of as some approximate solution) can be defined satisfying the following conditions:

A1) $z(P)$ always exists (unless $\Omega = \emptyset$) and can be computed by an efficient procedure;

A2) given any $P \in \mathcal{P}$ and the associated distinguished point $z = z(P)$, we can recognize when $z \in \Omega$ and if $z \notin \Omega$, we can construct an affine function $l(x)$ such that $P' = P \cap \{x\colon l(x) \leq 0\} \in \mathcal{P}$, and $l(z) > 0$, while $l(x) \leq 0$, $\forall x \in \Omega$, i.e. $\Omega \subset P' \subset P \setminus \{z\}$.

Under these conditions, one can attempt to solve problem (2) by the following *outer approximation method* (OA method) [8]:

0 Start with an initial polytope $P_1 \in \mathcal{P}$. Set $k = 1$.

1 Compute the distinguished point $z^k = z(P_k)$ (by A1)). If $z(P_k)$ does not exist, terminate: the problem is infeasible.
If $z(P_k) \in \Omega$, terminate. Otherwise, continue.

2 Using A2), construct an affine function $l_k(x)$ such that $P_{k+1} = P_k \cap \{x\colon l_k(x) \leq 0\} \in \mathcal{P}$ and $l_k(x)$ strictly separates z^k from Ω, i.e. satisfies

$$l_k(z^k) > 0, \quad l_k(x) \leq 0 \; \forall x \in \Omega. \qquad (2)$$

Set $k \leftarrow k + 1$ and return to Step 1.

Prototype OA (outer approximation) procedure.

The algorithm is said to be *convergent* if it is either finite or generates an infinite sequence $\{z^k\}$ every cluster point of which is an optimal solution of problem (2).

Usually the distinguished point z^k is defined as a vertex of the polytope P_k satisfying some criterion (e.g., minimizing a given concave function). In these cases, the implementation of the above algorithm requires a procedure for computing, at iteration k, the vertex set V_k of the current polytope P_k. At the beginning, V_1 is supposed to be known, while P_{k+1} is obtained from P_k simply by adding one more linear constraint $l_k(x) \leq 0$. Using this information V_{k+1} can be derived from V_k by an on-line vertex enumeration procedure [1].

EXAMPLE 1 (Concave minimization.)

Consider the problem (1) where $f(x)$ is concave and D is a convex compact set with $\operatorname{int} D \neq \emptyset$.

Assume that D is defined by a convex inequality $g(x) \leq 0$ and let $w \in \operatorname{int} D$. Take $\mathcal{P}$ to be the collection of all polytopes containing D. For every $P \in \mathcal{P}$ define $z := z(P)$ to be a minimizer of $f(x)$ over the vertex set V of P (hence, by concavity of $f(x)$, a minimizer of $f(x)$ over P). Clearly, if $z \in D$, it solves the problem. Otherwise, the line segment joining z to w meets the boundary of D at a unique point y and the affine function $l(x) = \langle p, x - y\rangle + g(y)$ with $p \in \partial g(y)$ strictly separates D from z (indeed, $l(z) = g(z) > 0$ while $l(x) \leq g(x) - g(z) + g(z) \leq 0$ for all $x \in D$) . Obviously $P' = P \cap \{x\colon l(x) \leq 0\} \in \mathcal{P}$, so Assumptions A1) and A2) are fulfilled and the OA algorithm can be applied. The convergence of the algorithm is easy to establish. □

EXAMPLE 2 (Reverse convex programming.)

Consider the problem (1) where $f(x) = \langle c, x\rangle$, while $D = \{x \in \mathbf{R}^n\colon h(x) \leq 0 \leq g(x)\}$ with $g(x)$, $h(x)$ continuous convex functions. Assume that the problem is *stable*, i.e. that $D = \operatorname{cl}(\operatorname{int} D)$, so a feasible solution $\bar{x} \in D$ is optimal if and only if

$$\{x \in D\colon \langle c, x - \bar{x}\rangle \leq 0\} \subset \{x\colon g(x) \leq 0\}. \qquad (3)$$

Also for simplicity assume a point w is available satisfying $\max\{h(w), g(w)\} < 0$ and $\langle c, w\rangle < \min\{\langle c, x\rangle\colon h(x) \leq 0 \leq g(x)\}$ (the latter assumption amounts to assuming that the constraint $g(x) \geq 0$ is essential).

Let Ω be the set of optimal solutions, $\mathcal{P}$ the collection of all polytopes containing Ω. For every

$P \in \mathcal{P}$ let $z = z(P)$ be a maximizer of $g(x)$ over the vertex set V of the polyhedron $P \cap \{x\colon \langle c, x\rangle \leq \gamma\}$, where γ is the value of the objective function at the best feasible solution currently available (set $\gamma = +\infty$ if no feasible solution is known yet). By (3), if $g(z) \leq 0$, then γ is the optimal value (for $\gamma < +\infty$), or the problem is infeasible (for $\gamma = +\infty$). Otherwise, $g(z) > 0$, and we can construct an affine function $l(x)$ strictly separating z from Ω as follows. Since $\max\{h(w), g(w)\} < 0$ while $\max\{h(z), g(z)\} > 0$ the line segment joining z, w meets the surface $\max\{h(x), g(x)\} = 0$ at a unique point y.

1) If $g(y) = 0$ (while $h(y) \leq 0$), then y is a feasible solution and since $y = \lambda w + (1-\lambda)z$ for some $\lambda \in (0,1)$ we must have $\langle c, y\rangle = \lambda\langle c, w\rangle + (1-\lambda)\langle c, z\rangle < \gamma$, so the cut $l(x) = \langle c, x - y\rangle \leq 0$ strictly separates z from Ω.

2) If $h(y) = 0$, then the cut $l(x) = \langle p, x - y\rangle + h(y) \leq 0$, where $p \in \partial h(y)$, strictly separates z from Ω (indeed, $l(x) \leq h(x) - h(y) + h(y) = h(x) \leq 0$ for all $x \in \Omega$ while $l(z) > 0$ because $l(w) < 0$, $l(y) = 0$).

Thus assumptions A1), A2) are satisfied, and again the OA algorithm can be applied. The convergence of the OA algorithm for this problem is established by a more elaborate argument than for the concave minimization problem (see [3], [8]). □

Various variants of OA method have been developed for a wide class of optimization problems, since any optimization problem described by means of differences of convex functions can be reduced to a reverse convex program of the above form [3]. However, a difficulty with this method when solving large scale problems is that the size of the vertex set V_k of P_k may grow exponentially with k, creating serious storage problems and making the computation of V_k almost impracticable.

Inner Approximation. Consider the concave minimization problem under linear constraints, i.e. the problem (2) when $f(x)$ is a concave function and D is a polytope in $\mathbf{R}^n$.

Without loss of generality we may assume that 0 is a vertex of D. For any real number $\gamma \leq f(0)$, the set $C_\gamma = \{x \in \mathbf{R}^n\colon f(x) \geq \gamma\}$ is convex and $0 \in D \cap C_\gamma$. Of course, $D \subset C_\gamma$ if and only if $f(x) \geq \gamma$ for all $x \in D$.

The idea of the *inner approximation* method (IA method), also called the *polyhedral annexation* method (or PA method) [3], is to construct a sequence of expanding polytopes $P_1 \subset P_2 \subset \cdots$ together with a nonincreasing sequence of real numbers $\gamma_1 \geq \gamma_2 \geq \cdots$, such that $\gamma_k \in f(D)$, $P_k \subset C_{\gamma_k}$, $k = 1, 2, \ldots$, and eventually $D \subset P_h$ for some h: then $\gamma_h \leq f(x)$ for all $x \in D$, i.e. γ_h will be the optimal value.

For every set $P \subset \mathbf{R}^n$ let P° be the *polar* of P, i.e. $P^\circ = \{y \in \mathbf{R}^n\colon \langle y, x\rangle \leq 1, \forall x \in P\}$. As is well known P° is a closed convex set containing 0 (in fact a polyhedron if P is a polyhedron), and $P \subset Q$ only if $P^\circ \supset Q^\circ$; moreover, if C is a closed convex set containing 0, then $(C^\circ)^\circ = C$. Therefore, setting $S_k = (P^k)^\circ$, the IA method amounts to constructing a sequence of nested polyhedra $S_1 \supset \cdots \supset S_h$ satisfying $S_k^\circ \subset C_{\gamma_k}$, $k = 1, \ldots, h$ and $S_h \subset D^\circ$. The key point in this scheme is: Given $\gamma_k \in f(D)$ and a polyhedron S_k such that $S_k^\circ \subset C_{\gamma_k}$, check whether $S_k \subset D^\circ$ and if there is $y^k \in S_k \setminus D^\circ$, then construct a cut $l_k(y) \leq 1$ to exclude y^k and to form a smaller polyhedron S_{k+1} such that $S_{k+1}^\circ \subset C_{\gamma_{k+1}}$ for some $\gamma_{k+1} \in f(D)$ satisfying $\gamma_{k+1} \leq \gamma_k$.

To deal with this point, define $s(y) = \max\{\langle y, x\rangle\colon x \in D\}$. Since $y \in D^\circ$ whenever $s(y) \leq 1$ we will have $S_k \subset D^\circ$ whenever

$$\max\{s(y)\colon y \in S_k\} \leq 1. \tag{4}$$

But clearly the function $s(y)$ is convex as the pointwise maximum of a family of linear functions. Therefore, denoting the vertex set and the extreme direction set of S_k by V_k, U_k, respectively, we will have (4) (i.e. $S_k \subset D^\circ$) whenever

$$\begin{cases} \max\{s(y)\colon y \in V_k\} \leq 1, \\ \max\{s(y)\colon y \in U_k\} \leq 0. \end{cases} \tag{5}$$

Thus, checking the inclusion $S_k \subset D^\circ$ amounts to checking (5), a condition that fails to hold in either of the following cases:

$$s(y^k) > 1 \quad \text{for some } y^k \in V_k \tag{6}$$

$$s(y^k) > 0 \quad \text{for some } y^k \in U_k. \tag{7}$$

In each case, it can be verified that if x^k maximizes $\langle y^k, x\rangle$ over D, and $\gamma_{k+1} = \min\{\gamma_k, f(x^k)\}$ while

$$\theta_k = \sup\left\{\theta\colon f(\theta x^k) \geq \gamma_{k+1}\right\},$$

then $S_{k+1} = S_k \cap \{y\colon \langle x^k, y\rangle \leq 1/\theta_k\}$ satisfies

$$P_{k+1} := S^\circ_{k+1} = \operatorname{conv}(P_k \cup \{\theta_k x^k\}) \subset C_{\gamma_{k+1}}.$$

In the case (6), S_{k+1} no longer contains y^k while in the case (7), y^k is no longer an extreme direction of S_{k+1}. In this sense, the cut $\langle x^k, y\rangle \leq 1/\theta_k$ excludes y^k. We can thus state the following algorithm.

0	By translating if necessary, make sure that 0 is a vertex of D. Let $\overline{x}^1$ be the best basic feasible solution available, $\gamma_1 = f(\overline{x}^1)$. Take a simplex $P_1 \subset C_{\gamma_1}$ and let $S_1 = P_1^\circ$, V_1 = vertex set of S_1, U_1 = extreme direction set of S_1. Set $k = 1$.
1	Compute $s(y)$ for every new $y \in (V_k \cup U_k) \setminus \{0\}$. If (5) holds, then terminate: $S_k \subset D^\circ$ so $\overline{x}^k$ is a global optimal solution.
2	If (6) or (7) holds, then let $$x^k \in \arg\max\left\{\left\langle y^k, x\right\rangle : x \in D\right\}.$$ Update the current best feasible solution by comparing x^k and $\overline{x}^k$. Set $\gamma_{k+1} = f(\overline{x}^{k+1})$.
3	Compute $\theta_k = \max\{\theta \geq 1\colon f(\theta x^k) \geq \gamma_{k+1}\}$ and let $$S_{k+1} = S_k \cap \left\{y\colon \left\langle x^k, y\right\rangle \leq \frac{1}{\theta_k}\right\}.$$ From V_k and U_k derive the vertex set V_{k+1} and the extreme direction set U_{k+1} of S_{k+1}. Set $k \leftarrow k+1$ and go to Step 1.

IA Algorithm (for concave minimization).

It can be shown that the IA algorithm is finite [3]. Though this algorithm can be interpreted as dual to the OA algorithm, its advantage over the OA method is that it can be started at any vertex of D, so that each time the set V_k has reached a certain critical size, it can be stopped and 'restarted' at a new vertex of D, using the last obtained best value of $f(x)$ as the initial γ_1. In that way the set V_k can be kept within manageable size.

Note that if D is contained in a cone M and $P_1 = \{x \in M\colon \langle v^1, x\rangle \leq 1\} \subset C_{\gamma_1}$, then it can be shown that (7) automatically holds, and only (6) must be checked [6].

Concavity Cut. The cuts mentioned above are used to separate an unfit solution from some convex set containing at least one optimal solution. They were first introduced in convex programming [2], [4]. Another type of cuts originally devised for concave minimization [7] is the following.

Suppose that a feasible solution $\overline{x}$ has already been known with $f(\overline{x}) = \gamma$ and we would like to check whether there exists a better feasible solution. One way to do that is to take a vertex x^0 of D with $f(x^0) > \gamma$ and to construct a cone M, as small as possible, vertexed at x^0, containing D and having exactly n edges. Since x^0 is interior to the convex set $C_\gamma = \{x\colon f(x) \geq \gamma\}$, each ith edge of M, for $i = 1, \ldots, n$, meets the boundary of C_γ at a uniquely defined point y^i (assuming that C_γ is bounded). Through these n points $y^1, \ldots, y^n$ (which are affinely independent) one can draw a unique hyperplane, of equation $\pi(x - x^0) = 1$ such that $\pi(y^i - x^0) = 1$ $(i = 1, \ldots, n)$, hence $\pi = e^\top U^{-1}$, where U is the matrix of columns $y^1 - x^0, \ldots, y^n - x^0$ and e denotes a vector of n ones. Since the linear inequality

$$e^\top U^{-1}(x - x^0) \geq 1 \tag{8}$$

excludes x^0 without excluding any feasible solution x better than $\overline{x}$, this inequality defines a valid cut. In particular, if it so happens that the whole polytope D is cut off, i.e. if

$$D \subset \left\{x\colon e^\top U^{-1}(x - x^0) \leq 1\right\}, \tag{9}$$

then $\overline{x}$ is a global optimal solution.

This cut is often referred to as a γ-valid *concavity cut* for (f, D) at x^0 [3]. Its construction requires the availability of a cone $M \supset S$ vertexed at x^0 and having exactly n edges. In particular, if the vertex x^0 of D has exactly n neighboring vertices then M can be taken to be the cone generated by the n halflines from x^0 through each of these neighbors of x^0. Note, however, that the definition of the concavity cut can be extended so that its construction is possible even when the cone M has more than n edges (as e.g., when x^0 is a degenerated vertex of D).

Condition (9), sufficient for optimality, suggests a cutting method for solving the linearly constrained concave minimization problem by using concavity cuts to iteratively reduce the feasible polyhedron. Unfortunately, experience has shown that concavity cuts, when applied repeatedly, tend to become shallower and shallower. Though these cuts can be significantly strengthened by exploiting additional structure of the problem (e.g., in concave quadratic minimization, bilinear program-

ming [5] and also in low rank nonconvex problems [6]), pure cutting methods are often outperformed by *branch and cut* methods where cutting is combined with successive partition of the space [8].

Concavity cuts have also been used in combinatorial optimization ('intersection cuts', or in a slightly extended form, 'convexity cuts').

Nonlinear Cuts. In many problems, *nonlinear cuts* arise in a quite natural way.

For example, consider the following problem of *monotonic optimization* [10]:

$$\max\{f(x)\colon g(x) \le 1,\ h(x) \ge 1,\ x \in \mathbf{R}^n_+\}, \quad (10)$$

where f, g, h are continuous increasing functions on $\mathbf{R}^n_+$ (a function $f(x)$ is said to be *increasing* on $\mathbf{R}^n_+$ if $0 \le x \le x' \Rightarrow f(x) \le f(x')$; the notation $x \le x'$ means $x_i \le x'_i$ for all i while $x < x'$ means $x_i < x'_i$ for all i). As argued in [10], a very broad class of optimization problems can be cast in the form (10). Define $G = \{x \in \mathbf{R}^n_+\colon g(x) \le 1\}$, $H = \{x \in \mathbf{R}^n_+\colon h(x) \ge 1\}$, so that the problem is to maximize $f(x)$ over the feasible set $G \cap H$. Clearly

$$0 \le x \le x' \in G \quad \Rightarrow \quad x \in G, \quad (11)$$

$$0 \le x \le x' \notin H \quad \Rightarrow \quad x \notin H. \quad (12)$$

Assume that $g(0) < 1$ and $0 < a \le x \le b$ for all $x \in G \cap H$ (so $0 \in \operatorname{int} G$, $b \in H$). From (11) it follows that if $z \in \mathbf{R}^n_+ \setminus G$ and $\pi(z)$ is the last point of G on the halfline from 0 through z, then the cone $K_{\pi(z)} = \{x \in \mathbf{R}^n_+\colon x > \pi(z)\}$ separates z from G, i.e. $G \cap K_{\pi(z)} = \emptyset$, while $z \in K_{\pi(z)}$.

A set of the form $P = \cup_{y \in V}\{x\colon 0 \le y\}$, where V is a finite subset of $\mathbf{R}^n_+$, is called a *polyblock* of vertex set V [9]. A vertex v is said to be *improper* if $v \le v'$ for some $v' \in V \setminus \{v\}$. Of course, improper vertices can be dropped without changing P. Also if $P \supset G \cap H$ then the polyblock of vertex set $V' = V \cap H$ still contains $G \cap H$ because $v \notin H$ implies that $[0, v] \cap H = \emptyset$. With these properties in mind we can now describe the *polyblock approximation* procedure for solving (10).

Start with the polyblock $P_1 = [0, b] \supset G \cap H$ and its vertex set $V_1 = \{b\} \subset H$. At iteration k we have a polyblock $P_k \supset G \cap H$ with vertex set $V_k \subset H$. Let $y^k \in \arg\max\{f(x)\colon x \in V_k\}$. Clearly y^k maximizes $f(x)$ over P_k, and $y^k \in H$, so if $y^k \in G$ then y^k is an optimal solution. If $y^k \notin G$ then the point $x^k = \pi(y^k)$ determines a cone K_{x^k} such that the set $P_{k+1} = P_k \setminus K_{x^k}$ excludes y^k but still contains $G \cap H$. It turns out that P_{k+1} is a polyblock whose vertex set V_{k+1} is obtained from V_k by adding n points $v^{k,1}, \ldots, v^{k,n}$ (which are the n vertices of the hyperrectangle $[x^k, y^k]$ adjacent to y^k) and then dropping all those which do not belong to H. With this polyblock P_{k+1}, we pass to iteration $k+1$.

In that way we generate a nested sequence of polyblocks $P_1 \supset P_2 \supset \cdots \supset G \cap H$. It can be proved that either y^k is an optimal solution at some iteration k or $f(y^k) \searrow \gamma := \max\{f(x)\colon x \in G \cap H\}$.

A similar method can be developed for solving the problem

$$\min\{f(x)\colon g(x) \le 1, h(x) \ge 1, x \in \mathbf{R}^n_+\}$$

by interchanging the roles of g, h and a, b. In contrast with what happens in OA methods, the vertex set V_k of the polyblock P_k in the polyblock approximation algorithm is extremely easy to determine. Furthermore this method admits restarts, which provide a way to prevent stall and overcome storage difficulties when solving large scale problems [10].

References

[1] Chen, P., Hansen, P., and Jaumard, B.: 'On-line and off-line vertex enumeration by adjacent lists', *Oper. Res. Lett.* **10** (1991), 403–409.

[2] Cheney, E.W., and Goldstein, A.A.: 'Newton's method for convex programming and Tchebycheff approximation', *Numerische Math.* **1** (1959), 253–268.

[3] Horst, R., and Tuy, H.: *Global optimization: deterministic approaches*, third ed., Springer, 1996.

[4] Kelley, J.E.: 'The cutting plane method for solving convex programs', *J. SIAM* **8** (1960), 703–712.

[5] Konno, H.: 'A cutting plane algorithm for solving bilinear programs', *Math. Program.* **11** (1976), 14–27.

[6] Konno, H., Thach, P.T., and Tuy, H.: *Optimization on low rank nonconvex structures*, Kluwer Acad. Publ., 1997.

[7] Tuy, H.: 'Concave programming under linear constraints', *Soviet Math.* **5** (1964), 1437–1440.

[8] Tuy, H.: *Convex analysis and global optimization*, Kluwer Acad. Publ., 1998.

[9] Tuy, H.: 'Normal sets, polyblocks and monotonic optimization', *Vietnam J. Math.* **27**, no. 4 (1999), 277–300.

[10] Tuy, H.: 'Monotonic optimization: Problems and solution approaches', *SIAM J. Optim.* (2000/to appear).

Hoang Tuy
Inst. Math.
P.O. Box 631,
Bo Ho, 10000 Hanoi
E-mail address: htuy@hn.vnn.vn

MSC2000: 90C26
Key words and phrases: cutting plane method, outer approximation, inner approximation, polyhedral annexation, concavity cut, intersection cut,convexity cut, nonlinear cut, polyblock approximation, monotonic optimization.

CUTTING-STOCK PROBLEM

A company that produces large rolls of paper, textile, steel, etc., usually faces the problem of how to cut the large rolls into smaller rolls, called finished rolls, in such a way that the demands for all finished rolls be satisfied. Any large roll is cut according to some cutting pattern and the problem is to find the cutting patterns to be used and to how many large rolls they should be applied. We assume, for the sake of simplicity, that each large roll has width W, an integer multiple of some unit and the finished roll widths are also specified by some integers $w_1, \ldots, w_m$. Let a_{ij} designate the number of rolls of width w_i produced by the use of the jth pattern, $i = 1, \ldots, m$, $j = 1, \ldots, n$. Let further b_i designate the demand for roll i, $i = 1, \ldots, m$, and $c_j = 1, j = 1, \ldots, n$. If $A = (a_{ij})$, $\mathbf{b} = (b_1, \ldots, b_m)^\top$, $\mathbf{c} = (c_1, \ldots, c_n)^\top$, then the problem is:

$$\begin{cases} \min & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{cases}$$

Here x_j means the number of jth cutting patterns to be used and as such, an all integer solution would be required to the problem. However, one is usually satisfied with an optimal solution of the above problem without the integrality restriction and, having that, a simple round-off procedure provides us with the solution to the problem.

In the above problem, however, the matrix A is huge, therefore we do not, and in most cases cannot, create it, by enumerating the cutting patterns. P.C. Gilmore and R.E. Gomory [3], [4] resolved this difficulty by an ingenious column generation technique. It works in such a way that we generate column j, in the course of the simplex algorithm, whenever needed. Assume that B is the current basis and designate by $\boldsymbol{\pi}$ the corresponding dual vector, i.e., the solution of the linear equation: $\boldsymbol{\pi}^\top B = \mathbf{c}_B^\top$. Now, if $\mathbf{a} = (a_1, \ldots, a_m)^\top \in \mathbf{Z}_+^m$ satisfies the inequality $\mathbf{w}^\top \mathbf{a} \leq W$, then, by definition, $\mathbf{a}$ represents a cutting pattern, a column of the matrix A. Since the cutting-stock problem is a minimization problem, the basis B is optimal if $\boldsymbol{\pi}^\top \mathbf{a} \leq 1$ for any $\mathbf{a}$ that satisfies $\mathbf{w}^\top \mathbf{a} \leq W$. We can check it by solving the linear program:

$$\begin{cases} \min & \boldsymbol{\pi}^\top \mathbf{a} \\ \text{s.t.} & \mathbf{w}^\top \mathbf{a} \leq W \\ & \mathbf{a} \in \mathbf{Z}_+^m. \end{cases}$$

If the optimum value is greater than 1, then the optimal $\mathbf{a}$ vector may enter the basis, otherwise B is an optimal basis and $\mathbf{x}_B$ is an optimal solution to the problem. The problem to find the vector $\mathbf{a}$ is a knapsack problem for which efficient solution methods exist.

In practice, however, frequently more complicated cutting-stock problems come up, due to special customer requirements depending on quality and other characteristics. In addition, we frequently need to include set up costs, capacity constraints and costs due to delay in manufacturing. These lead to the development of special algorithms as described in [1], [4], [5], [6], [7]. Recently Cs.I. Fábián [2] formulated stochastic variants of the cutting-stock problem, for use in fiber manufacturing.

See also: **Integer programming**.

References

[1] DYCKHOFF, H., KRUSE, H.J., ABEL, D., AND GAL, T.: 'Trim loss and related problems', *OMEGA Internat. J. Management Sci.* **13** (1985), 59–72.

[2] FÁBIÁN, CS.I.: 'Stochastic programming model for optical fiber manufacturing', *RUTCOR Res. Report* **34-98** (1998).

[3] GILMORE, P.C., AND GOMORY, R.E.: 'A linear programming approach to the cutting stock problem', *Oper. Res.* **9** (1961), 849–859.

[4] GILMORE, P.C., AND GOMORY, R.E.: 'A linear programming approach to the cutting stock problem, Part II', *Oper. Res.* **11** (1963), 863–888.

[5] GILMORE, P.C., AND GOMORY, R.E.: 'Multistage cutting stock problems of two and more dimensions', *Oper. Res.* **13** (1965), 94–120.

[6] JOHNSON, M.P., RENNICK, C., AND ZAK, E.: 'Skiving addition to the cutting stock problem in the paper industry', *SIAM Rev.* **39**, no. 3 (1998), 472–483.

[7] NICKELS, W.: 'A knowledge-based system for integrated solving cutting stock problems and production control in the paper industry', in G. MITRA (ed.): *Mathematical Models for Decision Support*, Springer, 1988.

András Prékopa
RUTCOR, Rutgers Center for Operations Research
New Jersey, USA
E-mail address: prekopa@rutcor.rutgers.edu
Csaba I. Fábián
Eötvös Loránd Univ.
Hungary
E-mail address: fabian@cs.elte.hu

MSC2000: 90B90, 90C59
Key words and phrases: cutting-stock problem, cutting pattern, column generation, knapsack problem.

CYCLIC COORDINATE METHOD, *CCM*

Often the solution of multivariable optimization problems it is desired to be done with a *gradient-free algorithm*. This may be the case when gradient evaluations are difficult, or in fact gradients of the underlying optimization method do not exist. Such a method that offers this feature is the method of the cyclic coordinate search and its variants.

The minimization problem considered is:

$$\min_{\mathbf{x}} f(\mathbf{x}).$$

The method in its basic form uses the coordinate axes as the search directions. In particular, the search directions $\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(n)}$, where the $\mathbf{d}^{(i)}$ are vectors of zeros, except for a 1 in the ith position. Therefore along each search direction $\mathbf{d}^{(i)}$ the corresponding variable x_i is changed only, with all remaining variables being kept constant to their previous values.

It is assumed here that the minimization is carried out in order over all variables with indices $1, \ldots, n$ at each iteration of the algorithm. However there are variants. The first of these is the *Aitken double sweep method*, which processes first the variables in the order mentioned above, and then in the second sweep returns in reverse order, that is $n-1, \ldots, 1$. The second variant is termed the *Gauss–Southwell method* [2], according to which the component (variable) with largest partial derivative magnitude in the gradient vector is selected for line searching. The latter requires the availability of first derivatives of the objective function.

The algorithm of the cyclic coordinate method can be summarized as follows:

1. Initialization Select a tolerance $\epsilon > 0$, to be used in the termination criterion of the algorithm. Select an initial point $\mathbf{x}^{(0)}$ and initialize by setting $\mathbf{z}^{(1)} = \mathbf{x}^{(0)}$. Set $k = 0$ and $i = 1$.
2. Main iteration Let α_i^* (scalar variable) be the optimal solution to the line search problem of minimizing $f(\mathbf{z}^{(i)} + \alpha \mathbf{d}_i)$. Set $\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} + \alpha_i^* \mathbf{d}^{(i)}$. If $j < n$, then increase i to $i+1$ and repeat step 2. Otherwise, if $j = n$, then go to step 3.
3. Termination check Set $\mathbf{x}^{(k+1)} = \mathbf{z}^{(n)}$. If the termination criterion is satisfied, for example $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \leq \epsilon$, then stop. Else, set $\mathbf{z}^{(1)} = \mathbf{x}^{(k+1)}$. Increase k to $k+1$, set $i = 1$ and repeat step 2.

The steps above outline the basic cyclic coordinate method, the Aitken and Gauss–Southwell variants can be easily included by modifying the main algorithm.

In terms of convergence rate comparisons, D.G. Luenberger [3] remarks that such comparisons are not easy. However, an interesting analysis presented there indicates that roughly $n-1$ coordinate searches can be as effective as a single gradient search. Unless the variables are practically uncoupled from one another then coordinate search seems to require approximately n line searches to bring about the same effect as one step of steepest descent.

It can generally be proved that the cyclic coordinate method, when applied to a differentiable function, will converge to a stationary point ([3], [1]). However, when differentiability is not present then the method can stall at a suboptimal point. Interestingly there are ways to overcome such difficulties, such as by applying at every pth iteration (a heuristic number, user specified) the search direction $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$. This is even applied in practice for differentiable functions, as it is found to be helpful in accelerating convergence. These modifications are referred to as *acceleration steps* or *pattern searches*.

See also: **Hooke–Jeeves method**; **Rosenbrock method**; **Powell method**; **Sequential simplex method**.

References

[1] BAZARAA, M.S., SHERALI, H.D., AND SHETTY, C.M.: *Nonlinear programming, theory and algorithms*, Wiley, 1993.

[2] FORSYTHE, G.E.: *Finite difference methods for partial differential equations*, Wiley, 1960.

[3] LUENBERGER, D.G.: *Linear and nonlinear programming*, second ed., Addison-Wesley, 1984.

Vassilios S. Vassiliadis
Chemical Engin. Dept. Univ. Cambridge
Pembroke Street, Cambridge CB2 3RA, UK
E-mail address: vsv20@cheng.cam.ac.uk
Raúl Conejeros
Chemical Engin. Dept. Univ. Cambridge
Pembroke Street, Cambridge CB2 3RA, UK
E-mail address: rjcrc2@cheng.cam.ac.uk

MSC2000: 90C30
Key words and phrases: cyclic coordinate search, line search methods, pattern search, Aitken double sweep method, Gauss–Southwell method, nondifferentiable optimization.