N

Nested Partitions Optimization

LEYUAN SHI[1], SIGURDUR OLAFSSON[2]
[1] University of Wisconsin, Madison, USA
[2] Iowa State University, Ames, USA

MSC2000: 90C59, 90C11

Article Outline

Introduction

The *nested partitions* (NP) method is a powerful optimization method that has been found to be very effective for solving large-scale discrete optimization problems. Such problems are common in many practical applications and the NP method is hence useful in diverse application areas. It can be applied to both operational and planning problems and has been demonstrated to effectively solve complex problems in both manufacturing and service industries.

The NP method was first introduced in [4] and application examples include diverse areas such as optimization of beam orientation in radiation therapy [1], feature selection in data mining [3], and product design [5].

Formulation

The NP method is particularly well suited for complex large-scale discrete optimization problems where traditional methods experience difficulty. It is, however, very broadly applicable and can be used to solve any optimization problem that can be stated mathematically in the following generic form:

$$\min_{x \in X} f(x) \,, \tag{1}$$

where the solution space or feasible region X is either a discrete or a bounded set of feasible solutions.

An important special type of problem that can be effectively addressed using the NP method is *mixed integer programs* (MIP) [6]. For such problems there may be one set of discrete variables and one set of continuous variables and the objective function and constraints are both linear. A general MIP can be stated as follows:

$$z_{\text{MIP}} = \min_{x,y \in X} c^1 x + c^2 y \,, \tag{2}$$

where $X = \left\{ x \in Z_+^n, y \in R^n : A^1 x + A^2 y \leq b \right\}$ and we use z_{MIP} to denote any linear objective function, that is, $z_{\text{MIP}} = f(x) = cx$. While some large-scale MIPs can be solved efficiently using exact mathematical programming methods, complex applications often give rise to MIPs where exact solutions can only be found for relatively small problems. When dealing with such complex large-scale problems the NP method provides an attractive alternative. However, even in such cases it may be possible to take advantage of exact mathematical programming methods by incorporating them into the NP framework. The NP method therefore provides a framework for combining the complementary bene-

fits of two optimization approaches that have traditionally been studied separately, namely, mathematical programming and metaheuristics.

Another important class of problems are *combinatorial optimization problems* (COP) where the feasible region is finite but its size typically grows exponentially with the number of input parameters of the problem. A general COP can be stated as follows:

$$\min_{x \in X} f(x) , \tag{3}$$

where $|X| < \infty$, but the objective function $f: X \to \mathbf{R}$ may be a complex nonlinear function. Sometimes it may have no analytic expression and must be evaluated through a model, such as a simulation model, a data mining model, or other application-dependent models. One advantage of the NP method is that it is effective for optimization when f is known analytically (deterministic optimization), when it is noisy (stochastic optimization), or even when it must be evaluated using an external process.

Methods

The NP method is best viewed as a metaheuristic framework, and it has similarities to branching methods in that like branch-and-bound it creates partitions of the feasible region. However, it also has some unique features that make it well suited for very hard large-scale optimization problems.

Metaheuristics have emerged as the most widely used approach for solving difficult large-scale combinatorial optimization problems [2]. A metaheuristic provides a framework to guide application-specific heuristics, such as a greedy local search, by restricting which solution or set of solutions should or can be visited next. For example, the tabu search metaheuristic disallows certain moves that might otherwise be appealing by making the reverse of recent moves tabu or forbidden. At the same time it always forces the search to take the best nontabu move, which enables the search to escape local optima. Similar to tabu search, most metaheuristics guide the search from solution to solution or possibly from a set of solutions to another set of solutions. In contrast, the NP method guides the search by determining where to concentrate the search effort. Any optimization method, such as an application-specific local search, other general purpose heuristic, or a mathematical programming method, can then be integrated within this framework.

The development of metaheuristics and other heuristic search methods has been made largely in isolation from the recent advancements in the use of mathematical programming methods for solving large-scale discrete problems. It is a very important and novel characteristic of the NP method that it provides a natural metaheuristic framework for combining the use of heuristics and mathematical programming and taking advantage of their complementary nature. Indeed, as far as we know, the NP method is the first systematic search method that enables users to simultaneously realize the full benefits of incorporating lower bounds through various mathematical programming methods and using any domain knowledge or heuristic search method for generating good feasible solutions. It is this flexibility that makes the NP method so effective for practical problems.

To concentrate the search effort, the NP method employs a decomposition approach similar to that of branch-and bound. Specifically, in each step the method partitions the space X of feasible solutions into the *most promising region* and the *complementary region*, namely, the set of solutions not contained in the most promising region. The most promising region is then partitioned further into subregions. The partitioning can be done exactly as branching for a branch-and-bound algorithm, but instead of focusing on obtaining lower bounds and comparing those bounds with a single primal feasible solution, the NP methods focuses on generating primal feasible solution from each of the subregions and the complementary region. This results in an upper bound on the performance of each of these regions. The region with the best feasible solution is judged the most promising and the search is focused accordingly. A best upper bound does not guarantee that the corresponding subset contains the optimal solution, but since the NP method also finds primal feasible solutions for the complementary region it is able to recover from incorrect moves. Specifically, if the best solution is found in one of the subregions this becomes the new most promising region, where if it is in the complementary region the NP method backtracks. This focus on generating primal feasible solutions and the global perspective it achieves

through backtracking are distinguishing features of the NP method that set it apart from similar branching methods.

Unlike exact optimization methods such as branch-and-bound the NP method does not guarantee that the correct region is selected in each move of the algorithm. Incorrect moves can be corrected through backtracking, but for the method to be both effective and efficient, the correct move must be made frequently. How this is accomplished depends on how the feasible solutions are generated.

In what we refer to as the *pure NP method* feasible solutions are generated using simple uniform random sampling. To increase the probability of making the correct move the number of samples should be increased. A purely uniform random sampling is rarely efficient, however, and the strength of the NP method is that it can incorporate application-specific methods for generating feasible solutions. In particular, for practical applications domain knowledge can often be utilized to very effectively generate good feasible solutions. We call such implementations *knowledge-based NP methods*. We will also see examples of what we refer to as *hybrid NP methods* where feasible solutions are generated using either general heuristic methods such as greedy local search, genetic algorithm, or tabu search, or mathematical programming methods. If done effectively, incorporating such methods into the NP framework makes it more likely that the correct move is made and hence makes the NP method more efficient. Indeed, such hybrid and knowledge-based implementations are often an order of magnitude more efficient than uniform random sampling.

In addition to the method for generating feasible solutions, the probability of making the correct move depends heavily on the partitioning approach. A generic method for partitioning is usually straightforward to implement but by taking advantage of special structure and incorporating this into intelligent partitioning the efficiency of the NP method may be improved by an order of magnitude. The strength of the NP method is indeed in this flexibility. Special structure, local search, any heuristic search, and mathematical programming can all be incorporated into the NP framework to develop optimization algorithms that are more effective in solving large-scale optimization problems than when these methods are used alone.

Cases

Here we introduce three application examples that illustrate the type of optimization problems for which the NP method is particularly effective. For each application the optimization problem has a complicating aspect that makes it difficult for traditional optimization methods. For the first of these problems, resource-constrained project scheduling, the primary difficulty is in a set of complicating constraints. For the second problem, the feature selection problem, the difficulty lies in a complex objective function. The third problem, radiation treatment planning, has both difficult-to-satisfy constraints and a complex objective function that cannot be evaluated through an analytical expression. Each of the three problems can be solved effectively by the NP method by incorporating our understanding of the application into the framework.

Resource-Constrained Project Scheduling

Planning and scheduling problems arise as critical challenges in many manufacturing and service applications. One such problem is the resource-constrained project scheduling problem that can be described as follows. A project consists of a set of tasks to be performed and given precedence requirements between some of the tasks. The project scheduling problem involves finding the starting time of each task so that the overall completion time of the project is minimized. It is well known that this problem can be solved efficiently using what is called the critical path method that uses forward recursion to find the earliest possible completion time for each task. The completion time of the last task defines the makespan or the completion time of the entire project.

Now assume that one or more resources are required to complete each task. The resources are limited so if a set of tasks requires more than the available resources they cannot be performed concurrently. The problem now becomes NP-hard and cannot be solved efficiently to optimality using any traditional methods. To state the problem we need the following notation: V is the set of all tasks, E is the set of precedence constraints, p_i is the processing time of task $i \in V$, R is the set of resources, R_k are the available resources of type $k \in R$, and r_{ik} are resources of type k required by task i.

The decision variables are the starting times for each task: x_i is the starting time of task $i \in V$.

Finally, for notational convenience we define the set of tasks processed at time t as

$$V(t) = \{i : x_i \le t \le x_i + p_i\} \ . \tag{4}$$

With this notation, we now formulate the resource-constrained project scheduling problem mathematically as follows:

$$\min \max_{i \in V} x_i + p_i \ , \tag{5}$$

$$x_i + p_i \le x_j, \quad \forall (i,j) \in E \ , \tag{6}$$

$$\sum_{i \in V(t)} r_{ik} \le R_k, \quad \forall k \in R, t \in \mathbf{Z}_+^1 \ , \ x_i \in \mathbf{Z}_+^1 \ . \tag{7}$$

Here the precedence constraints (6) are easy, whereas the resource constraints (7) are hard. By this we mean that if the constraints (7) are dropped then the problem becomes easy to solve. Such problems, where complicating constraints transform the problem from easy to very hard, are common in large-scale optimization. Indeed the classic job shop scheduling problem can be viewed as a special case of the resource-constrained project scheduling problem where the machines are the resources. Without the machine availability constraints the job shop scheduling problem reduces to a simple project scheduling problem. Other well-known combinatorial optimization problems have similar properties. For example, without the subset elimination constraints the classic traveling salesman problem reduces to a simple assignment problem that can be solved efficiently.

The flexibility of the NP method allows us to address such problems effectively by taking advantage of special structure when generating feasible solutions. It is important to note that it is very easy to use sampling to generate feasible solutions that satisfy very complicated constraints. Therefore, when faced with a problem with complicating constraints we want to use random sampling to generate partial feasible solutions that resolve the difficult part of the problem and then complete the solution using the appropriate efficient optimization method.

For example, when a feasible solution for the resource-constrained-project scheduling problem is generated, the resource allocation should be generated using random sampling and the solution can then be completed by applying the critical path method to determine the starting times for each task. This requires reformulating the problem so that the resource and precedence constraints can be separated, but such a reformulation is rather easily achieved by noting that the resource constraints can be resolved by determining a sequence between the tasks that require the same resource(s) at the same time. Once this sequence has been determined it can be added as precedence constraints and the remaining solution can be generated using the critical path method. Feasible solutions can therefore be generated in the NP method by first randomly sampling a sequence to resolve resource conflicts and then applying the critical path method. Both procedures are very fast, so complete sample solutions can be generated rapidly.

We also note that constraints that are difficult for optimization methods such as mathematical programming are sometimes very easily addressed in practice by incorporating domain knowledge. For example, a domain expert may easily be able to specify priorities among tasks requiring the same resource(s) in the resource-constrained project scheduling problem. The domain expert can therefore, perhaps with some assistance from an interactive decision support system, specify some priority rules to convert a very complex problem into an easy-to-solve problem. The NP method can effectively incorporate such domain knowledge into the optimization framework by using the priority rules when generating feasible solutions. This is particularly effective because the domain expert would not need to specify priority rules to resolve all resource conflicts. Rather, any available priority rule or other domain knowledge can be incorporated to guide the sampling.

The same structure can be used to partition intelligently. Instead of partitioning directly using the decision variables (x_i), we note that it is sufficient to partition to resolve the resource conflicts. Once those are resolved then the problem is solved. This approach is applicable to any problem that can be decomposed in a similar manner.

Feature Selection

Knowledge discovery and data mining is a relatively new field that has experienced rapid growth owing to

its ability to extract meaningful knowledge from very large databases. One of the problems that must usually be solved as part of practical data mining projects is the feature selection problem, which involves selecting a good subset of variables to be used by subsequent inductive data mining algorithms. The problem of selecting a best subset of variables is well known in the statistical literature as well as in machine learning. The recent explosion of interest in data mining for addressing various business problems has led to a renewed interest in this problem. From an optimization point of view, feature selection can clearly be formulated as a COP where binary decision variables determine if a feature (variable) is included or excluded. The solution space can therefore be stated very simply as all permutations of a binary vector of length n, where n is the number of variables. The size of this feasible region is 2^n, so it experiences exponential growth, but typically there are no additional constraints to complicate its structure.

On the other hand, there is no consensus objective function that measures the quality of a feature or a set of features. Tens of alternatives have been proposed in the literature, including both functions that measure the quality of individual features and functions that measure the quality of a set of features. However, no single measure is satisfactory in all cases and the ultimate measure is therefore: Does it work? In other words, when the features selected are used for learning does it result in a good model being induced? The most effective feature selection approach in terms of solution quality is therefore the wrapper approach, where the quality of a set of features is evaluated by applying a learning algorithm to the set and evaluating its performance. Specifically, an inductive learning algorithm, such as decision tree induction, support vector machines or neural networks, are applied to training data containing only the features selected. The performance of the induced model is evaluated and this performance is used to measure the quality of the feature subset. This objective function is not only nonlinear, but since a new model must be induced for every feature subset it is also very expensive to evaluate.

Mathematically, the feature selection can be stated as the following COP:

$$\min_{x\in\{0,1\}^n} f(x)\,, \tag{8}$$

that is, $X = \{0,1\}^n$. Feature selection is therefore a very hard COP not because of the complexity of the feasible region, although it does grow exponentially, but owing to the complexity of an objective function that is very expensive to evaluate. However, this is also an example where application-specific heuristics can be effectively exploited by the NP method.

Significant research has been devoted to methods for measuring the quality of features. This includes information-theoretic methods such as using Shannon's entropy to measure the amount of information contained in each feature: the more information, the more valuable the feature. The entropy is measured for each feature individually and it can hence be used as a very fast local search or a greedy heuristic, where the features with the highest information gain are added one at a time. While such a purely entropy based feature selection will rarely lead to satisfactory results, the NP method can exploit this by using the entropy measure to define an intelligent partitioning.

We let $X(k) \subseteq X$ denote the most promising region in the kth iteration and partition the set into two disjoint subsets (note that $X(0) = X$):

$$X_1(k) = \{x \in X(k)\colon x_i = 1\}\,, \tag{9}$$

$$X_2(k) = \{x \in X(k)\colon x_i = 0\}\,. \tag{10}$$

Hence, a partition is defined by a sequence of features $x_1, x_2, \ldots, x_n$, which determines the order in which the features are either included ($x_i = 1$) or excluded ($x_i = 0$).

We calculate the information gain Gain(i) of feature i, which is the expected reduction in entropy that would occur if we knew the value of feature i, that is,

$$\text{Gain}(i) = I - E(i)\,, \tag{11}$$

where I is the expected information that is needed to classify a given instance and $E(i)$ is the entropy of each feature. The maximum information gain, or equivalently the minimum entropy, determines a ranking of the features. Thus, we select

$$i_1 = \arg\min_{i\in\{1,2,\ldots,n\}} E(i)\,,$$

$$i_2 = \arg \min_{i \in \{1,2,\ldots,n\} \setminus \{i_1\}} E(i) ,$$

$$\vdots$$

$$i_n = \arg \min_{i \in \{1,2,\ldots,n\} \setminus \{i_1,\ldots,i_{n-1}\}} E(i) .$$

The feature order $i_1, i_2, \ldots, i_n$ defines an intelligent partition for the NP method and this has been found to be an order of magnitude more efficient than an average arbitrary partitioning [3]. We can use a similar idea to generate feasible solutions from each region using a sampling strategy that is biased towards including features with high information gain. A very fast greedy heuristic can thus greatly increase the efficiency of the NP method while resulting in much higher quality solutions that the greedy heuristic is able to achieve on its own.

Radiation Treatment Planning

Health care delivery is an area of immense importance where optimization techniques have been used increasingly in recent years. Radiation treatment planning is an important example of this and intensity-modulated radiation therapy (IMRT) is a recently developed complex technology for such treatment. It employs a multileaf collimator to shape the beam and to control, or modulate, the amount of radiation that is delivered from each of the delivery directions (relative to the patient). The planning of the IMRT is very important because it needs to achieve the treatment goal while incurring the minimum possible damage to other organs. Because of its complexity the treatment planning problem is generally divided into several subproblems. The first of these is termed the *beam angle selection* (BAS) problem. In essence, BAS requires the determination of roughly four to nine angles from 360 possible angles subject to various spacing and opposition constraints.

Designing an optimal IMRT plan requires the selection of beam orientations from which radiation is delivered to the patient. These orientations, called beam angles, are currently manually selected by a clinician on the basis of his/her judgment. The planning process proceeds as follows. A dosimetrist selects a collection of angles and waits 10–30 min while a dose pattern is calculated. The resulting treatment is likely to be unacceptable, so the angles and dose constraints are adjusted, and the process is repeated. Finding a suitable collection of angles often takes several hours. The goal of using optimization methods to identify quality angles is to provide a better decision support system to replace the tedious repetitive process just described. An integer programming model of the problem contains a large number of binary variables and the objective value of a feasible point is evaluated by solving a large, continuous optimization problem. For example, in selecting five to ten angles, there are between 4.9^{10} and 8.9×10^{19} subsets of $0, 1, 2, \ldots, 359$.

The BAS problem is complicated by both an objective function with no analytical expression and constraints that are hard to satisfy. In the end an IMRT plan is either acceptable or not and the considerations for determining acceptability are too complex for a simple analytical model. Thus, the acceptability and hence the objective function value for each plan must be evaluated by a qualified physician. This makes evaluating the objective not only expensive in terms of time and effort, but also introduces noise into the objective function because two physicians may not agree on the acceptability of a particular plan. The constraints of the BAS problem are also complicated since each beam angle will result in radiation of organs that are not the target of the treatment. There are therefore two types of constraints: the target should receive the minimum amount of radiation and other organs should receive no more than some maximum amount of radiation. Since these bounds need to be specified tightly the constraints are hard to satisfy.

The BAS problem illustrates how mathematical programming can be effectively incorporated into the NP framework. Since the evaluation of even a single IMRT plan must be done by an expert and is hence both time-consuming and expensive, it is imperative to impose a good structure on the search space that reduces the number of feasible solutions that need to be generated. This can be accomplished through an intelligent partitioning, and specifically by computing the optimal solution of an integer program with a much simplified objective function [1]. The output of the integer program then serves to define an intelligent partitioning. For example, suppose a good angle set (50°, 80°, 110°, 250°, 280°, 310°, 350°) is found by solving the integer program. We can then partition on the first angle in the set, which is 50° in this example. Then one subregion includes angle 50°, and the other excludes 50°.

Conclusions

The NP method is a powerful metaheuristic for solving large-scale discrete optimization problems. The method systematically partitions the feasible region into subregions and moves from one region to another on the basis of information obtained by randomly generating feasible sample solutions from each of the current regions. The method keeps track of which part of the feasible region is the most promising in each iteration and the number of feasible solutions generated, and hence the computational effort is always concentrated in this most promising region.

The efficiency of the NP algorithm depends on making the correct move frequently. This success probability depends in turn on both the partitioning and the method for generating feasible solutions. For any practical application it is therefore important to increase the success probability by developing intelligent partitioning methods, incorporating special structure into weighted sampling, and applying randomized heuristics to generate high-quality feasible solutions.

The NP method has certain connections to standard mathematical programming techniques such as branch-and-bound. However, the NP method is primarily useful for problems that are either too large or too complex for mathematical programming to be effective. But even for such problems mathematical programming methods can often be used to solve either a relaxed problem or a subproblem of the original and these solutions can be effectively incorporated into the NP framework.

The three application examples presented here illustrate the broad usefulness of the NP method in both manufacturing and service industries, and how it can take advantage of special structure and application-specific heuristics to improve the efficiency of the search.

References

1. D'Souza WD, Meyer RR, Shi L (2004) Selection of beam orentations in intensity-modulated radiation therapy using single-beam indices and integer programming. Phys Med Biol 49:3465–3481
2. Gendreau M, Potvin J (2005) Metaheuristics in Combinatorial Optimization. Annu Oper Res 140:189–213
3. Ólafsson S, Yang J (2005) Intelligent Partitioning for Feature Selection. INFORMS J Comput 17(3):339–355
4. Shi L, Ólafsson S (2000) Nested Partitions Method for Global Optimization. Oper Res 48:390–407
5. Shi L, Ólafsson S, Chen Q (2001) An Optimization Framework for Product Design. Manag Sci 47:1681–1692
6. Wolsey L (1998) Integer Programming. Wiley, New York

Network Design Problems

NDP

DIETMAR CIESLIK
University Greifswald, Greifswald, Germany

MSC2000: 05C05, 05C40, 68R10, 90C35

Article Outline

Keywords

Network; Shortest path; Minimum spanning trees; Steiner minimal tree; Network flows

Scientific or engineering applications usually require solving mathematical problems. Such applications in accordance with networks span a wide range, from modeling the evolution of species in biology to modeling soap films for grids of wires; from the design of collections of data to the design of heating or air-conditioning systems in buildings; and from the creation of oil and gas pipelines to the creation of communication networks, road and railway lines. These are all network design problems of significant importance and nontrivial complexity. The network topology and design characteristics of these systems are classical examples of optimization problems.

I. Intuitively speaking, a *network* is a set of points and a set of connections where each connection joins one point to another and has a certain length. The combinatorial structure of such a network is described as a graph G which is defined to be a pair (V, E) where

- V is any finite set of elements, called *vertices*, and
- E is a finite family of elements which are unordered pairs of vertices, called *edges*.

Additionally, assume that a function $l\colon E \to \mathbf{R}$ is given for the edges of the graph G. Usually, assume that l

has only positive values and call it a length-function. A (connected) graph equipped with a length-function is called a *network*.

The *length of a subgraph* $G' = (V', E')$, $V' \subseteq V$, $E' \subseteq E \cap \binom{V'}{2}$, of the network G is defined as

$$L(G') := \sum_{e \in E'} l(e).$$

Each network $G = (V, E)$ with the length-function l: $E \to \mathbf{R}$ is a metric space (V, ρ) by defining the distance function in the way that $\rho(v, v')$ is the length of a shortest path between the vertices v and v' in G. That means that ρ: $V \times V \to \mathbf{R}$ is a real-valued function satisfying:

i) $\rho(v, v') \geq 0$ for all v, v' in V;
ii) $\rho(v, v') = 0$ if and only if $v = v'$;
iii) $\rho(v, v') = \rho(v', v)$ for all v, v' in V; and
iv) $\rho(v, v') \leq \rho(v, w) + \rho(w, v')$ for all v, v', w in V (*triangle inequality*).

The *problem of finding shortest paths* in a graph with a length-function is an important and well-studied problem. Such a path is easy to find by an algorithm created by E.W. Dijkstra [7]:

```
PROCEDURE shortest path
    InputInstance();
    Start with the vertex v;
    Label the vertex v with 0 :  L(v) = 0;
    REPEAT
        Select an edge ww' between a labeled vertex w and an unlabeled vertex w' such that the quantity L(w)+l(ww') is minimal as possible;
        Label w' :  L(w') = L(w) + l(ww')
    UNTIL v' is achieved
END shortest path;
```

A pseudocode for a procedure finding a shortest path between two vertices v and v' in a network

Assume that the procedure runs if all vertices of the network are achieved then the procedure creates a spanning tree rooted at the vertex v containing shortest paths from v to every vertex. Moreover, the label $L(w)$ denotes the distance from v to w, in other terms, $\rho(v, w) = L(w)$.

Conversely, each finite metric space can be represented as a graph with a nonnegative length-function [23].

Graphs lend themselves as natural models of transportation as well as communication networks. Consequently, it is natural to study network design problems such as optimal facility location problems for graphs and as graphs in metric spaces.

The core network design problem is the *minimum spanning tree problem*, where one wish to design a minimum cost network contains a path from each vertex to each other. Such a network must be a tree, which is called a *minimum spanning tree* (MST). Creating a minimum spanning tree is the problem one has the longest history of all ND problems, starting with O. Borůvka [2] in 1926. See [19] for an excellent historical survey.

All the known efficient minimum spanning tree algorithms are special cases of a general greedy method, in which one builds up an MST edge-by-edge, including appropriate short edges and excluding appropriate long edges ones. Perhaps the simplest method to find an MST is due to J.B. Kruskal [30]:

```
PROCEDURE minimum spanning tree
    InputInstance();
    Start without any edge;
    REPEAT
        Choose the shortest edge that does not form
        form a circle with edges already chosen
    UNTIL | V | −1 edges are chosen
END minimum spanning tree;
```

A pseudocode for a procedure finding a minimum spanning tree in a network $G = (V, E)$

About an efficient implementation of Kruskal's algorithm and several more effective procedures compare [3].

II. A general ND problem is for a given configuration of vertices and/or edges to find a network which contains these objects, fulfilling some predetermined requirements and minimizes a given objective function. This is quite general and models a wide variety of problems.

J. McGregor Smith [40] presents a classification of applications for network design problems. Generalizing this is

- *Large region networks*. The metric in large geographic regions is given by the shortest great circle distance between the points on the (Euclidean)

sphere. Large region location problems arise when the difference between the Euclidean and the great circle metrics is considerable. Location of international headquarters or distribution centers and planning oil or natural gas pipelines or long distance telephone lines are examples [27] and [38].

- *Regional networks*. Consider inter-urban networks, like communication networks, railway lines and interstate highway networks. R.F. Love and J.G. Morris [33] study a variety of mathematical forms as samples for intercity, urban and rural road distances. Moreover, they found that the p-norms, and linear combinations of these, are the best possible. Recent surveys are [4] and [34].
- *Macro scale networks*. Chemical processing plants, urban arterial systems and similar intra-urban systems are typical applications. In these situations the rectilinear metric is often used. If the structure of the possible connections is predetermined it is also possible to formulate the problem as a network design problem in graphs.
- *Intermediate scale networks*. Electric, heating and air-conditioning systems in buildings are examples of network optimization problems where Steiner points can reduce the overall minimum cost solution of the network. The rectilinear metric is the most frequent measure of the distance in these applications. For models and methods see [26].
- *Micro scale networks*. The design of very large scale integration (VLSI) networks is an example of Steiner's problem where the overall interconnecting length of the network is crucial for the solution. In this class of applications, the rectilinear metric is again the most frequent metric. It is also conceivable to use a linear combination of rectilinear and maximum norm [26] and [32].
- *Evolutionary networks*. Molecular sequences are used to reconstruct the course of the evolution. Since the evolution is assumed to have proceed from a common ancestral species in a tree-like branching of species, this process is generally modeled by a tree. The key question is the reconstruction of this tree based on the contemporary data. Molecular data comes as either DNA sequences (composed of nucleotides from an alphabet of four letters; namely the four nucleic acids Adenine, Guanine, Thymine and Cytosine) or sequences of proteins (composed of amino acids from an alphabet of 20 letters; namely Alanine, Arginine, Asparagine, …, Valine). As the metric often the Hamming distance is used. Surveys are given in [13] and [39].

III. Considering a group of ND problems which are the problems with connectivity requirements one finds:

- The *bounded degree minimum spanning tree problem* (abbreviated: BDMST-problem), which is defined as follows: Given a finite set N of points in a metric space and an integer $\beta > 1$. Find a spanning tree $T = (N, E)$ such that T has minimal length among all candidates with a maximum degree of the vertices less or equal than β.

  Finding a BDMST of maximum degree $\beta = 2$ is equivalent to solving the *traveling salesman problem*, which is known to be NP-hard. The borderline which divides the classes NP-hard and $\mathcal{P}$ for the BDMST problem depending on the space and the quantity β are described in [5] and [37].
- Many modified (minimum) spanning tree problems are presented in the literature, for instance
  - find a spanning tree interconnecting all vertices with minimal maximum degree;
  - find a spanning tree interconnecting all vertices with at least k leaves in the tree;
  - find a spanning tree $T = (V, E)$ such that the quantity

    $$\sum_{v,v' \in V} L(\text{the path from } v \text{ to } v' \text{ in } T)$$

    is minimal;
  - find a spanning tree isomorphic to a given tree;

  These problems are NP-complete in general, but it is shown that they can be solved more easily in several specific cases, [15,17], and [28].
- The *Steiner minimal tree problem*, where one seek a minimum network that connects a set N of designated terminal points. Any network solving this problem must be a tree, which is called a *Steiner minimal tree* (SMT). It may contain vertices different from the points which are to be connected. Such points are called *Steiner points*. In other terms; an SMT for N is a minimum spanning tree on $N \cup Q$, where Q is a set of additional vertices inserted into the metric space in order to achieve a minimal solution. In general, however, it is impossible to compute the number of Steiner points in an easy way

independently from the determination of an SMT. Additionally, Steiner point locations in the space are not prespecified from a candidate list of point locations.

Steiner's problem for graphs was originally formulated by S.B. Hakimi [22] in 1971. Since then, the problem has received considerable attention in the literature. Several exact algorithms and heuristics have been suggested and discussed.

R.M. Karp [29] showed that Steiner's problem is NP-complete. An algorithm which finds a solution in exponential time is given in [8]. The algorithm is based on the dynamic programming methodology using a decomposition property. On the other hand, Hakimi [22] remarked that an SMT for N in a network $G = (V, E)$ can be found by the enumeration of minimum spanning trees of subgraphs of G induced by supersets of N. E.L. Lawler [31] suggested a modification of this algorithm, using the fact that the number of Steiner points is bounded by $|N| - 2$ which shows that not all subsets V' must be considered.

A recent survey about Steiner minimal trees in networks is given in [26].

Algorithmic problems on arbitrary graphs often remain NP-complete even when restricted to special classes of graphs, yet may becomes solvable in polynomially bounded time on others. For instance, Steiner's problem remains NP-complete even for planar graphs [14]; yet, it can be solved in linear time in several specific graphs [41,43,44].

The geometric version of the Steiner minimal tree problem was originated by P. Fermat [10] early in the 17th century and by C.F. Gauss [16] in 1836. Perhaps starting with the book [6], in 1941, the *Gauss problem* became popularized under the name of *Steiner's problem*. That is: Given finite set of points in a Euclidean space, find a network which connects all points of the set with minimal length.

A classical survey of Steiner's problem in the Euclidean plane is [18] and is termed 'Steiner minimal tree' for the shortest interconnecting network and 'Steiner points' for the additional vertices.

Without loss of generality, the following is true for any SMT for a finite set N of points:

1) the degree of each vertex is at most three;
2) the degree of each Steiner point equals three; and two edges is incident to a Steiner point meet at as angle of 120°;
3) there are at most $|N| - 2$ Steiner points.

In the Euclidean plane one has a geometric construction by ruler and compass originated in [35] and [42]. Recent (1999) surveys about Steiner minimal trees, also in other geometries than the Euclidean ones, are given in [4] and [26].

Clearly, an MST is an approximation of an SMT. More exactly: One can find a tree interconnecting a finite set of points in a metric space in fast time (namely, $O(n^2)$-time, where n is the number of given points or the number of vertices in N, respectively) with a length at most twice the length of a shortest possible tree, namely an SMT. Hence, it is of interest to consider the quantity

$$\inf\left\{\frac{L(\text{SMT for } N)}{L(\text{SMT for } N)}:\ \begin{array}{c}N \text{ a finite set}\\ \text{of points}\end{array}\right\},$$

which is called the *Steiner ratio* of the space and says how much the total length of an MST can be decreased by allowing Steiner points.

space	Steiner ration	source
Plane with rectilinear norm	$2/3 = 0.66666\cdots$	[25]
Euclidean norm	$\sqrt{3}/2 = 0.86602\cdots$	[9]
Plane with p-norm	$2/3 \le m \le \sqrt{3}/2$	[4]

IV. Most of the ND problems can be modeled by an integer program.

Let $G = (V, E$ be a graph. A cut S in G is a partition of the vertex-set V into two nonempty parts, S and $V \setminus S$. An edge e crosses the cut S, written by $e \in \gamma(S)$, if it has exactly one endpoint in each part. Now, let l: $E \to \mathbf{R}$ be a length-function. Define the following integer linear program:

$$\begin{cases}\min & \displaystyle\sum_{e\in E} l(e)\cdot x_e\\ \text{s.t.} & \displaystyle\sum_{e\in\gamma(S)} x_e \ge p(S), \quad \emptyset \ne S \subset V,\\ & x_e \in \{0,1\}, \quad e \in E,\end{cases}$$

whereby p: $2^V \to \mathbf{N}$ is a function parametrizing the ND problem which is to solve. If one has two vertices v and v', and sets $p(S) = 1$ when S contains v but not v', then the program models the shortest path problem (between v and v'). If $p(S) = 1$ for all cuts S, then the program models the minimum spanning tree problem. Other ND problems with connectivity constraints discussed by integer linear programs are mentioned in [20] and [21].

V. The second group of ND problems are the problems with capacity constraints.

Let $G = (V, E)$ be a directed graph, usually called a *digraph*, that is

- V is a finite set of elements, called *vertices*, and
- $E \subseteq V \times V$ is a finite family, called the *set of edges*.

Assume, that there are two distinguished vertices, a source v_0 and a sink v_1 in G, and that there is a (directed) path from v_0 to v_1. Additionally, assume that there is a nonnegative capacity-function c: $E \to \mathbf{R}$.

A flow f on the digraph G is a nonnegative function on the edges such that

1) f does not exceed the capacities: $0 \leq f(e) \leq c(e)$ for every edge e; and
2) f satisfies the so-called *Kirchhoff-condition*

$$\sum_{(u,v)\in E} f(u,v) = \sum_{(v,w)\in E} f(v,w)$$

for every vertex $v \in V \setminus \{v_0, v_1\}$.

The quantity

$$\sum_{(v_0,v)\in E} f(v_0,v) = \sum_{(v,v_1)\in E} f(v,v_1)$$

is called the *value of the flow* f. The problem is to find a flow of maximum value, called a *maximum flow*.

The fundamental theory of network flows was developed by L.R. Ford Jr. and D.R. Fulkerson [11,12]: Similarly as above, a cut is defined to be a vertex partition S and $V \setminus S$ such that $v_0 \in S$ and $v_1 \in V \setminus S$. The capacity of the cut is

$$c(S) = \sum_{e\in\gamma(S)} c(e).$$

Ford and Fulkerson's main result, the *max-flow min-cut theorem*, states that the maximum flow value equals the minimum cut capacity.

Ford and Fulkerson proved this theorem by devising an algorithm that, given a flow f, either finds a cut whose capacity equals the flow value or finds a way to increase the the flow value along an augmenting path from v_0 to v_1.

```
PROCEDURE maximum flow
    InputInstace();
    Start with the zero flow;
    REPEAT
        find an augmenting path from v0 to v1;
        increase the flow value by altering the flows
        along the edges of the path
    UNTIL it no longer applies
END maximum flow;
```

A pseudocode for a procedure finding a maximum flow from a source v_0 to a sink v_1 in a graph $G = (V, E)$ equipped with a capacity-function

If all capacities are integers this procedure produces a maximum flow. Note, if the capacities are arbitrary real numbers the algorithm need never terminate, and successive flow values, though they will converge, need not converge to the maximum flow value.

Surveys for network flow algorithms, including classical work and a discussion of complexity, are [1] and [36].

VI. Combining ND problems with connectivity and capacity constraints there is the minimum cost flow problem, which is to determine a least cost shipment of a commodity through a network in order to satisfy the network possibilities.

Let $G = (V, E)$ be a digraph. Associated with each vertex v there is a number $b(v) \in \mathbf{R}$, satisfying $\sum_{v\in V} b(v) = 0$. A vertex v is called a *source*, a *sink* or a *transshipment vertex* if $b(v)$ is positive, negative or zero, respectively. Additionally, assume that there is a nonnegative capacity-function c: $E \to \mathbf{R}$ and a positive cost-function (i. e., length-function) l: $E \to \mathbf{R}$. Then the *minimum cost flow problem* is formulated as

$$\begin{cases} \min & \displaystyle\sum_{e\in E} l(e)\cdot x_e \\ \text{s.t.} & \displaystyle\sum_{(w,v)\in E} x_{(w,v)} - \sum_{(v,w)\in E} x_{(v,w)} = b(v), \\ & \quad v \in V, \\ & 0 \leq x_e \leq c(e), \qquad e \in E, \\ & x_e \quad \in \{0,1\}, \quad e \in E. \end{cases}$$

This problem is discussed in the literature many times; one of several available good sources is [1] which includes several polynomial time algorithms. A general background is [24].

See also

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

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms and applications. Prentice-Hall, Englewood Cliffs
2. Borůvka O (1926) O jistém problému minimálnim. Práca Moravské Prirodovědecké Spolnečnosti 3:37–58
3. Cheriton D, Tarjan RE (1976) Finding minimum spanning trees. SIAM J Comput 5:724–742
4. Cieslik D (1998) Steiner minimal trees. Kluwer, Dordrecht
5. Cieslik D (1998) Using Hadwiger numbers in network design. In: DIMACS, 40. Am Math Soc, Providence, pp 59–78
6. Courant R, Robbins H (1941) What is mathematics? Oxford Univ. Press, Oxford
7. Dijkstra EW (1959) A note on two problems in connection with graphs. Numer Math 1:269–271
8. Dreyfus SE, Wagner RA (1972) The Steiner problem in graphs. Networks 1:195–207
9. Du D-Z, Hwang FK (1992) A proof of the Gilbert–Pollak conjecture on the Steiner ratio. Algorithmica 7:121–136
10. Fermat P (1934) Abhandlungen über Maxima und Minima. Oswalds Klassiker der exakten Wissenschaft, vol 238. H. Miller, reprint from original.
11. Ford LR Jr, Fulkerson DR (1956) Maximal flow through a network. Canad J Math 8:399–404
12. Ford LR Jr, Fulkerson DR (1962) Network flow theory. Princeton Univ. Press, Princeton
13. Foulds LR (1994) Graph theory applications. Springer, Berlin
14. Garey MR, Johnson DS (1977) The rectilinear Steiner tree problem is NP-complete. SIAM J Appl Math 32:826–834
15. Garey MR, Johnson DS (1979) Computers and intractibility. Freeman, New York
16. Gauss CF (1917) Briefwechsel Gauss–Schuhmacher. In: Werke, vol. X. pp 459–468
17. Gavish B (1982) Topological design of centralized computer networks - Formulations and algorithms. Networks 12:355–377
18. Gilbert EN, Pollak HO (1968) Steiner minimal trees. SIAM J Appl Math 16:1–29
19. Graham RL, Hell P (1985) On the history of the minimum spanning tree problem. Ann Hist Comput 7:43–57
20. Grötschel M, Monma CL (1990) Integer polyhedra arising from certain network design problems with connectivity constraints. SIAM J Discret Math 3:502–523
21. Grötschel M, Monma CL, Stoer M (1994) Design of survivable networks. In: Handbook Oper Res and Management Sci. North-Holland, Amsterdam
22. Hakimi SB (1971) Steiner's problem in graphs and its implications. Networks 1:113–133
23. Hakimi SL, Yau SS (1964) Distance matrix of a graph and its realizability. Quart Appl Math 22:305–317
24. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht
25. Hwang FK (1976) On Steiner minimal trees with rectilinear distance. SIAM J Appl Math 30:104–114
26. Hwang FK, Richards DS, Winter P (1992) The Steiner tree problem. North-Holland, Amsterdam
27. Ivanov AO, Tuzhilin AA (1994) Minimal networks - The Steiner problem and its generalizations. CRC Press, Boca Raton
28. Jungnickel D (1994) Graphen, Netzwerke und Algorithmen. BI Wissenschaftsverlag, Mannheim
29. Karp RM (1962) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of Computer Computations. Springer, New York, pp 85–103
30. Kruskal JB (1956) On the shortest spanning subtree of a graph and the travelling salesman problem. Proc 7:48–50
31. Lawler EL (1976) Combinatorial optimization - Networks and matroids. Holt, Rinehart and Winston, New York
32. Lengauer T (1990) Combinatorial algorithms for integrated circuit layout. Teubner and Wiley, Stuttgart
33. Love RF, Morris JG (1972) Modelling inter-city road distances by mathematical function. J Oper Res Soc 23:61–71
34. Love RF, Morris JG, Wesolowsky G (1989) Facilities location - Models and methods. North-Holland, Amsterdam
35. Melzak ZA (1961) On the problem of Steiner. Canad Math Bull 4:143–148

36. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization. Prentice-Hall, Englewood Cliffs
37. Robins G, Salowe JS (1995) Low-degree minimum spanning trees. Discrete Comput Geom 14:151–165
38. Rubinstein JH, Weng JF (1997) Compression theorems and Steiner ratios on spheres. J Combin Optim 1:67–78
39. Setubal J, Meidanis J (1997) Introduction to computational molecular biology. PWS, Boston, MA
40. Smith JM (1985) Generalized Steiner network problems in engineering design. In: Design Optimization. pp 119–161
41. Wald JA, Colbourn CJ (1983) Steiner trees, partial 2-trees, and minimum IFI networks. Networks 13:159–167
42. Winter P (1985) An algorithm for the Steiner problem in the Euclidean plane. Networks 15:323–345
43. Winter P (1986) Generalized Steiner problem in series-parallel networks. J Algorithms 7:549–566
44. Winter P (1987) Steiner problems in networks: A survey. Networks 17:129–167

Network Location: Covering Problems

TIMOTHY J. LOWE
University Iowa, Iowa City, USA

MSC2000: 90C35, 90B10, 90B80

Article Outline

Keywords

Maximum coverage location problem; Uncapacitated facility location problem

The *covering problem on a network* involves the decision problem of determining the location of one or more 'facilities' or 'centers' to provide service to several clients located at known points on the network. To provide service to a client, a facility must be located 'close enough' to the client. In a more general version of the problem, there may be fixed costs associated with locating facilities, and also there may be penalty costs associated with not serving clients (not locating a facility close enough to the client). In this situation, to minimize total cost there is a trade-off between establishing facilities and not serving clients.

Let $N(V, A)$ be a connected undirected network with node set $V = \{v_1, \ldots, v_m\}$ and arc set A. Each arc $a \in A$ has a given length $l_a \geq 0$. If we consider $a = [v_i, v_j]$ as a line segment with length l_a, then any point on a can be defined by its distance from v_i (or from v_j). We denote by $d(x, y)$ the shortest path distance on the arcs of N between the points x and y on N. If X is a set of points on N, then $D(X, y)$ denotes the distance between y and a point in X which is closest to y.

Without loss of generality, we assume that the clients are located at the nodes of N. Furthermore, we assume that each node $v_i \in V$ is the site of exactly one client. To specify the notion of 'coverage', for $i = 1, \ldots, m$, let the 'covering radius' $r_i \geq 0$ be associated with v_i. In order for client i to be covered, we require that at least one facility x be located so that $d(x, v_i) \leq r_i$. Equivalently, if X is a set of located facilities, $D(X, v_i) \leq r_i$.

In our version of the problem, we assume that facilities can only be located at members of a subset V' of V, where $V' = \{v_1, \ldots, v_n\}$, $n \leq m$. Since both clients and facilities are located at nodes of N, we can define an $m \times n$ (0–1) covering matrix A, where $a_{ij} = 1$ if and only if $d(v_j, v_i) \leq r_i$. Thus if $a_{ij} = 1$ and if there is a facility located at v_j, then client i is covered. Let $c_j > 0$ be the cost of locating a facility at node v_j, $j = 1, \ldots, n$, and let $b_i > 0$ be the penalty cost associated with not covering client i. Finally, let z_i, $i = 1, \ldots, m$ be a (0–1) variable. Also, let x_j, $j = 1, \ldots, n$ be a (0–1) variable. Then with the above, we can formulate the covering problem as the following (0–1) integer programming problem:

$$\text{(P)} \quad \min \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} b_i z_i$$

subject to:

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j + z_i \geq 1, & i = 1, \ldots, m, \\ x_j \in \{0, 1\}, & j = 1, \ldots, n, \\ z_i \in \{0, 1\}, & i = 1, \ldots, m. \end{cases}$$

In an optimal solution to problem (P), note that $z_i = 1$ only when client i is not covered. That is, only when $x_j = 0$ for every j where $d(v_j, v_i) \leq r_i$.

For some versions of the problem there is a 'budget constraint' imposed on the facilities. This constraint usually takes the form:

$$\sum_{j=1}^{n} c_j x_j \leq M. \tag{1}$$

In the special case where each $c_j = 1$, the budget constraint simply limits the number of facilities that can be located (to a total of M). This problem, when each $b_i = 1$ and the cost of locating facilities is eliminated from the objective function, is often called the *maximum coverage location problem*, since when the objective function is minimized, a minimum number of clients are not covered and so a maximum number of clients are covered. Henceforth, we will only consider versions of problem (P) that do not include (1).

Several applications of problem (P) and its variants have been reported in the literature (see [11] for a partial list of applications). Besides direct applications of the problem, in [9] it is shown that the *p-center problem on a network* (the problem of locating no more than p facilities to minimize the maximum distance between any client and its closest facility) can be solved by solving a sequence of covering problems. As further evidence of its applicability, in [7] it is shown that the *uncapacitated facility location problem* can be formulated as a covering problem. Thus, the covering problem is one of the very fundamental network location problems.

On a general network, the covering problem is known to be *NP*-hard. Heuristics for general (0–1) covering problems have been well-studied ([3,4,5,8,10]).

A popular heuristic is the greedy heuristic which works as follows. (For notational convenience, in the following discussion of the heuristic we assume that each client must be covered. Thus the z_i variables can be removed from (P). If this is not the case, modifications to the heuristic are obvious.) At each iteration t, let A^t be the submatrix of A that consists of rows and columns of A that have not been removed thus far. For each column j of A^t compute $r_j^t = c_j/I_j^t$, where I_j^t is the number of nonzero entries in column j of A^t. Set x_{j^*} to 1 where j^* is an index j for which r_j^t is minimum.

Remove j^* from A^t as well as every row i where $a_{ij^*} = 1$. The resulting matrix is A^{t+1} and the heuristic continues until there are no nonzero entries remaining in the matrix.

Thus the heuristic chooses the column of A^t for which the corresponding facility covers uncovered clients at 'least cost per coverage'. The procedure continues until all clients are covered.

There is a performance guarantee of the greedy heuristic that is independent of the cost coefficients, but does depend upon the entries in the matrix A. It can be shown that the ratio of the objective function value derived via the greedy approach to the optimal objective function value will not exceed $H(d) = \sum_{k=1}^{d} 1/k$, where d is the maximum number of nonzero entries in any column of A.

When the underlying network, $N(V, A)$, is a tree, T (a connected undirected network without cycles), problem (P) can be solved in polynomial time. A. Kolen and A. Tamir [7] have shown that in the case of a tree network, the rows and columns of A can be permuted (in polynomial time) so that the matrix A is in *standard greedy form* (does not contain the submatrix $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$). They then show that any covering problem (P) where A is in standard greedy form can be solved in $O(mn)$. See [1] for the existence of covering problems where A is standard greedy, but where the problem cannot be derived from a tree network location problem. Thus, the Kolen–Tamir approach may be applicable to a broader class of covering problems on networks.

In the special case of (P) on a tree, where each $c_j = 1$ and every client must be covered (b_i is sufficiently large to prohibit noncoverage), even more efficient solution methods are possible (see [2,6], and [12]). With each $c_j = 1$, problem (P) reduces to the problem of minimizing the number of facilities located subject to the condition that every client is covered.

The procedure of B.C. Tansel, R.L. Francis, T.J. Lowe, and M.L. Chen [12] involves a physical 'string model'. Their approach does not involve direct use of the formulation (P) and facilities can be located anywhere (on nodes or arcs) on the tree T. The procedure begins by inscribing straight-line segments on a planar surface such that each segment represents an arc of T. Next, a string of length r_i is fastened to each node v_i of the inscribed tree.

The procedure involves pulling the strings toward the interior of the tree by first considering strings fas-

tened to nodes at the tips of the tree. Facilities are located at points on the tree where the ends of tight strings reach. Once a facility is located each remaining string that reaches the facility is removed from the model, since the corresponding client is covered by that facility.

See also

- ▶ Auction Algorithms
- ▶ Combinatorial Optimization Algorithms in Resource Allocation Problems
- ▶ Communication Network Assignment Problem
- ▶ Competitive Facility Location
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Facility Location with Externalities
- ▶ Facility Location Problems with Spatial Interaction
- ▶ Facility Location with Staircase Costs
- ▶ Generalized Networks
- ▶ Global Optimization in Weber's Problem with Attraction and Repulsion
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ MINLP: Application in Facility Location-Allocation
- ▶ Multifacility and Restricted Location Problems
- ▶ Network Design Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Optimizing Facility Location with Rectilinear Distances
- ▶ Piecewise Linear Network Flow Problems
- ▶ Production-Distribution System Design Problem
- ▶ Resource Allocation for Epidemic Control
- ▶ Shortest Path Tree Algorithms
- ▶ Single Facility Location: Circle Covering Problem
- ▶ Single Facility Location: Multi-objective Euclidean Distance Location
- ▶ Single Facility Location: Multi-objective Rectilinear Distance Location
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Transportation and Location Problems
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium
- ▶ Voronoi Diagrams in Facility Location
- ▶ Warehouse Location Problem

References

1. Broin M, Lowe TJ (1986) A dynamic programming algorithm for covering problems with (greedy) totally balanced constraint matrices. SIAM J Alg Discrete Meth 7:348–357
2. Chandrasekaran R, Tamir A (1982) Polynomially bounded algorithms for locating p-centers on a tree. Math Program 22:304–315
3. Chvátal V (1979) A greedy heuristic for the set-covering problem. Math Oper Res 4:233–235
4. Hochbaum DS (1997) Approximating covering and packing problems: Set cover, vertex cover, independent set and related problems. In: Hochbaum DS (ed) Approximation Algorithms for NP-Hard Problems. PWS, Boston
5. Johnson DS (1974) Approximation algorithms for combinatorial problems. J Comput Syst Sci 9:256–278
6. Kariv O, Hakimi SL (1979) An algorithmic approach to network location problems. Part 1. The p-centers. SIAM J Appl Math 37:513–538
7. Kolen A, Tamir A (1990) Covering problems. In: Discrete Location Theory. Wiley, New York
8. Lovász L (1975) On the ratio of optimal integral and fractional covers. Discret Math 13:383–390
9. Minieka E (1970) The m-center problem. SIAM Rev 12:138–139
10. Nemhauser G, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York
11. Schilling DA, Jayaraman V, Barkhi R (1993) A review of covering problems in facility location. Location Sci 1:25–55
12. Tansel BC, Francis RL, Lowe TJ, Chen ML (1982) Duality and distance constraints for the nonlinear *p*-center problem and covering problem on a tree network. Oper Res 30:725–744

Neural Networks for Combinatorial Optimization

THEODORE B. TRAFALIS, SUAT KASAP
School of Industrial Engineering,
University Oklahoma, Norman, USA

MSC2000: 90C27, 90C30

Article Outline

Keywords

Combinatorial optimization; Neural networks; Nonlinear programming; Global optimization

Most of the engineering design problems and applications can be formulated as a nonlinear programming problem in which the objective function to be optimized is nonlinear and has many local optima in its feasible region. It is desirable to find a local optimum that corresponds to a near global optimum. The problem of finding a global minimum or maximum is known as the *global optimization problem*. The main difficulties in finding a global optimum are that there are no operationally useful optimality conditions for identifying whether a point is indeed a global optimum, except in cases of special structured problems [16].

In this article, one class of global optimization problems, specifically combinatorial optimization problems will be mainly discussed. The combinatorial optimization problems deal with problems of maximizing or minimizing an objective function subject to inequality and/or equality constraints over a set of combinatorial alternatives. Finding optimal solutions to such problems where the decision variables are combinatorial or discrete is known as *combinatorial optimization problems*. A naive way to solve combinatorial optimization problems is to list all of the feasible solutions of a given problem, then evaluate its objective function, and choose the best one as an optimum solution. However, even though it is possible in principle to solve the problem in this way, in practice it is not, because of the large number of possible feasible solutions to any problem of a reasonable size. Most of the combinatorial optimization problems are *NP*-complete [34]. Because of the combinatorial structure of these problems the time needed to solve them grows exponentially with the size of the problem. For *NP*-complete problems, there is no algorithm that provides an exact solution to the problem in polynomial time.

Over the last three decades, the combinatorial optimization problem is one of the most challenging problems that has received considerable attention. The traditional solution methodologies for combinatorial optimization problems can be categorized into three groups as exact, heuristic, and approximation methods [50]. Exact methods guarantee to obtain an optimum solution, but the computational time for obtaining an optimum solution is usually an exponential function of the number of variables. The simplex method and branch and bound methods are some examples of exact methods. The heuristic methods are problem-specific methods based on success without formal analysis of performance. Simulated annealing (SA), tabu search, and genetic algorithms are some examples of heuristic methods [45]. On the other hand, the approximation methods generate feasible solutions that are near optimal solutions. Neural networks based methods are perceived as approximation methods.

This article reviews the use of NNs for combinatorial optimization problems and provides a survey of most of the NN approaches that have been applied to combinatorial optimization problems. In Section 2, basic definitions, classifications and applications of NNs are introduced. In Section 3, the mathematical basis of problem formulation for NNs based on an energy function is explained. In Section 4, various combinatorial optimization problems studied on NNs are presented. Finally, Section 5 is the conclusion.

Neural Networks

Neural Network (NN) models are algorithms for intellectual tasks such as learning and optimization that are based on the concept of how the human brain works. A NN model is composed of a large number of processing elements called *neurons*. Each neuron is connected to other neurons by links, each with an associated weight. Neurons without links toward them are called *input neurons* and with no link leaving away from them are called *output neurons*. The neurons are represented by state variables. State variables are functions of the weighted-sum of input variables and other state variables. Each neuron performs a simple transformation at the same time in a parallel-distributed manner. The input-output relation of the transformation in a neuron is characterized by an *activation function*.

Some examples of activation functions are the threshold function, linear, and sigmoidal functions. The combination of input neurons, output neurons, and links between neurons with associated weights constitute the architecture of the NN.

Mathematically, a NN model is a directed graph with the following properties [12]:

1) Each neuron (node) is associated with a state variable S_i and an activation threshold (bias) v_i.
2) Each link (edge) between two neurons i and j associates with it a weight w_{ij}.
3) Each neuron (node) defines a transfer function $f_i(S_i, w_{ij}, v_i)$, which determines the state of the neuron.

The classification of NNs can be done in different ways. According to the nature of activation, they can be categorized as deterministic and stochastic NNs. *Deterministic NNs* activate their states by using deterministic activation functions such as the threshold, linear, or sigmoidal functions. Most of the existing NN models are deterministic. *Stochastic NNs* activate their states according to a probability distribution. A typical example of stochastic NNs is the Boltzmann Machine (BM). According to the nature of the connectivity, NNs can be categorized as feed-forward and recurrent NNs. If a directed graph has no closed paths, then it is called a *feed-forward NN*. A typical example of the feed-forward NN is the popular multilayer perceptron. Conversely, if a directed graph has closed paths, then it is called a *recurrent NN*. A typical example of the recurrent NN is the Hopfield NN ([9,12]).

Modern era of NNs is said to have begun with the introductory work of W.S. McCullough and W. Pitts [28]. They have proposed a general theory of information processing based on networks of neurons. Each one of these neurons can only take the output values 1 or 0 by representing the active and resting states of neurons respectively. NNs have come a long way from the early days of McCullough and Pitts. NNs have established themselves as an interdisciplinary subject with rich connections in the neuroscience, psychology, physical sciences, mathematics and engineering.

NNs have very close ties with optimization and the ties are manifested mainly into two aspects. On one aspect, a lot of learning algorithms have been developed based on optimization techniques to train NNs to perform modeling tasks ([2,9,12,37,39]). On the other aspect, NNs have been developed for solving optimization problems ([4,6,12,26,36,40,44,50,51]). Because of the inherent nature of parallel and distributed information processing in NNs, they are promising computational models for solving large scale optimization problems in real-time ([3,5,21,22,23,49]).

Neural Networks and Combinatorial Optimization Problems

NNs have been proposed as a model of computation to solve combinatorial optimization problems. To solve combinatorial optimization problems using NNs requires a mapping of the problem onto the NNs in such a way that one can identify a solution from outputs of the neurons. In other words, to solve the combinatorial optimization problems using NNs, the key is to map the problem into the architecture of the NN for which the stable state represents the solution of the combinatorial optimization problem.

Combinatorial optimization problems can be solved using NNs by following two approaches.

1) By formulating a combinatorial optimization problem in terms of minimizing an energy function of either discrete or continuous variables ([1,3,4,5,8,13,15,35,36,40]).
2) By designing competition based NNs in which neurons are allowed to compete to become active under certain conditions ([7,9]).

These two approaches suggest that NNs are an alternative for solving combinatorial optimization problems as compared to other optimization techniques. Motivations for using NNs include the improvement in the speed of the operation through massively parallel computation and possible hardware design advantages. One of the main advantages of NN approaches to classical optimization approaches is the inherently parallel and distributed nature of the dynamic solution procedure. Therefore, NNs are capable to solve large scale optimization problems in real time ([3,5,21,22,23,49]).

NNs have been used to solve many combinatorial optimization problems since the pioneering work of J.J. Hopfield and D.W. Tank [15]. They formulated the traveling salesman problem (TSP) on a highly interconnected NN and made exploratory numerical studies on a 10-city and 30-city TSP. They showed that an energy function can be defined for an analog Hopfield NN and the NN always converges to a local min-

imum of this function. To find the (near) global minimum of the energy function, the Boltzmann Machine (BM) was proposed [2]. The BM was the first attempt to combine SA and NN architectures to solve combinatorial optimization problems. However, BM is designed for discrete variables with the disadvantage of being so slow. Another approach by combining SA and NNs was proposed in [24]. In this study, a noise term has been added to the neural dynamics. The intensity of that noise term has been shown to depend on the state of the neuron as well as on a temperature parameter T. By selecting an appropriate temperature schedule $T(t)$, the resulting NN converged to a near global minimum of the NN energy function that is in the neighborhood of the global minimum of the combinatorial optimization problem. The Hopfield NN was extended to handle inequality constraints and simplified its convergence characteristics by the eigenvalue analysis [1]. The mathematical basis of the behavior of the Hopfield NN by means of an idealization of the Hopfield network was given in [47]. This study helped to give a better understanding of the relationship between NNs and the combinatorial optimization. The analog neural solution of the combinatorial optimization problem was considered [48]. The solution method was analyzed based on the Lagrange multiplier for the continuous relaxation problem of 0-1 integer programming. The theory and methodology of the deterministic NNs for the combinatorial optimization were presented in [50]. This study was extended to the convex programming [51]. The use of NN methods for the combinatorial optimization problems was reviewed ([13,26,44]). A systematic approach to design competition based NNs for the combinatorial optimization was presented in [7]. The competition based NNs were studied in detail [9].

The procedure of NN approaches to the combinatorial optimization mostly begins with the formulation of an energy function. Ideally, the minimum of this energy function corresponds to the optimal solution of the combinatorial optimization problem. Most of the existing NN approaches to combinatorial optimization problems formulate an energy function by incorporating an objective function and constraints through functional transformation and numerical weighting. A functional transformation is usually used to convert constraints to a penalty function to penalize the violations of constraints. Numerical weighting is often used to balance constraint satisfaction and objective minimization. In the NN formulation for combinatorial optimization problem, constraint violations enter to the energy function in an explicit way. In general, such an energy function will have the following form:

$$E = \sum_i A_i(\text{constraint violation})_i + \text{cost} \tag{1}$$

where $A_i > 0$ and cost is an objective function that is independent from the constraint violations. By minimizing the energy function E, we attempt to minimize the cost while at the same time minimize the constraint violations.

The second step in designing NNs for combinatorial optimization problems is to derive a dynamic equation (also known as state equation or motion equation). The dynamic equation of the NN prescribes the motion of the activation states of the NN. A properly derived dynamic equation can ensure that the state of the NN reaches equilibrium and the equilibrium-state of the NN satisfies the constraints and optimizes the objective function of the problem. Currently, the dynamic equations of most NNs for optimization problems are derived by letting the time derivative of a state vector to be directly proportional to the negative gradient of an energy function. The dynamic equations and energy functions for a wide variety of combinatorial optimization problems were studied [28]. The last step is to determine the architecture of the NN in terms of neurons and connections based on the derived dynamic equation in such a way that one can identify a solution from the outputs of the neurons.

The success of optimization using NNs lies in the formulation of an energy function, based on the objective function and constraints of the given optimization problem, and the derivation of a dynamic equation of NNs, based on the formulated energy function.

The NN approaches to optimization problems have been started with [14]. He has pioneered an approach for solving minimization problems by utilizing the collective computational capabilities of NNs. He has mapped the objective function and the problem constraints onto a quadratic energy function of the neural states that presents the energy of the system of neurons.

The energy function had the following form:

$$E(S) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij}S_iS_j - \sum_{i=1}^{N} S_iI_i \ , \tag{2}$$

where S_i represents the output state of neurons. C_{ij} represents the strength of the link between neurons i and j. I_i represents the input bias; the input of neurons is denoted as x_i. The motion equations of neurons is defined by

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = -x_i + \sum_{j\neq i}^{N} C_{ij}S_i + I_i \ , \quad \text{where} \quad S_i = f_i(x_i) \ . \tag{3}$$

Hopfield showed that for a NN with symmetric connections and nonnegative elements on the diagonal of the C matrix, the NN will always converge by performing the gradient descent to a stable state, which is a local minimum of the energy function. Many difficult optimization problems can be formulated as the minimization of a quadratic form and thus can be mapped onto the NN model. Hopfield and Tank [15] extended the discrete model to an analog model where $0 \leq S_i \leq 1$. The modified energy function is:

$$E(S) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij}S_iS_j - \sum_{i=1}^{N} S_iI_i + \sum_{i=1}^{N}\frac{1}{R_i}\int_0^{v_i} g_i^{-1}(S)\ \mathrm{d}S \ , \tag{4}$$

where R_i is the input resistance to unit i, and g_i is the activation function. The dynamics of the neuron update is defined by

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = -x_i + \sum_{j\neq i}^{N} C_{ij}S_i + I_i \ , \quad \text{where} \quad S_i = \frac{1}{2}(1+\tanh(x_i)) \ . \tag{5}$$

This model is synchronous, continuous and deterministic.

NNs to minimize the energy function are composed of either discrete or continuous neurons. The possibilities of using different types of NNs such as discrete-state and discrete-time, continuous-state and discrete-time, and continuous-state and continuous-time networks to solve combinatorial optimization problems were investigated [29]. One usually prefers to use continuous neurons rather than discrete ones for two reasons. First, a continuous network tends to avoid oscillations between stable states. Second, solutions are much better than those provided by discrete NNs, because the valley of energy landscapes are wider and neuron outputs are not restricted to corners of the hypercube during convergence. NNs do not guarantee globally optimal solutions but they compute locally optimal solutions.

In the last two decades, spin glass theories of statistical physics have been extensively studied because of their applications to other areas like optimization and neural networks ([10,30,46]). A similarity between an NN of the type proposed in [28] and a system of elementary magnetic spins was pointed in [25]. These ideas were developed further [14] by studying how such a NN or a spin system can store and retrieve information. The idea of an energy function was used to formulate a new way of understanding the computation performed by NNs.

To solve combinatorial optimization problems using NNs and the mean field theory (MFT) of the statistical physics ([3,4,5,17,18,20,31,40]) was introduced by C. Peterson and J.R. Anderson [38]. This was the first detailed attempt to use the MFT and NNs for solving combinatorial optimization problems after a brief description by Hopfield and Tank [15]. The problem was mapped onto the NN such that a neuron being 'on' corresponded to a certain decision and then relaxed the system using the MFT in order to escape from local minima. In this study, the problem was mapped onto an *Ising glass model* ([10,11,31,40]). Another method for finding approximate solutions to combinatorial optimization problems within NNs and the MFT concept was presented [40]. The problem was mapped onto a *Potts glass model* ([19,31,40,46,52]) in this study. A mean field annealing (MFA) was described based on the MFT and SA. Similar studies based on the MFT of statistical physics to solve combinatorial optimization problems have been studied ([4,5,49]). A general framework for the MFA algorithm was derived and its relationship with to Hopfield NNs was shown [4]. A NN mapping and the MFT solution method for finding good solutions to combinatorial optimization prob-

lems containing inequality constraints like the knapsack problem were developed ([32,33]).

A method to solve combinatorial optimization problems was proposed ([17,18]) based on 'two-layer random field model' using the technique of the MFA. This method determined the appropriate values of the weights of two kinds of terms in objective function; a cost term that should be minimized and a constraint term that expresses constraints imposed on solutions. A parallel MFT method and a new temperature scheduling method named as maximum entropy cooling schedule were proposed [43]. The relationship between the MFT NN model and the continuous-time Hopfield NN was analyzed [20] by using the theory of dynamical systems. This study showed that the asynchronous MFT model was equivalent to the continuous-time Hopfield NN on the nature of the fixed points and hence guaranteed theoretically usage of the MFT model for solving combinatorial optimization problems in place of the continuous-time Hopfield NN.

Combinatorial Optimization Problems

In this section, we list some of the combinatorial optimization problems that have been studied to be solved using NNs. Most of these problems are *NP*-complete [34]. The mapping of combinatorial optimization problems onto NNs by focusing graph problems such as graph K-partitioning, maximum graph matching, and maximum clique problems was studied [13]. The use of NNs for combinatorial optimization problems was reviewed [26]. A number of interesting combinatorial optimization problems such as graph partitioning, traveling salesman problem, vertex cover, maximum clique, maximum independent set, number partitioning, maximum matching, set cover, and graph coloring were studied to show how to map these problems into NNs [44]. A NN mapping and the MFT solution method for finding good solutions to combinatorial optimization problems containing inequality constraints like the knapsack problem were developed [32]. The MFT approach to the knapsack problem was extended to multiple knapsacks and generalized assignment problems [33]. The following three problems have been studied mostly by several researchers using different approaches.

Quadratic Assignment Problem

The quadratic assignment problem (QAP) represents a large class of combinatorial optimization problems arising in a variety of planning and designing contexts. The QAP seeks to minimize a quadratic cost function for assignment of a number of objects to positions that can be mathematically described as follows [50]:

$$\begin{cases} \min & f(x) = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} c_{ijkl}x_{ij}x_{kl} \\ \text{s.t.} & \sum_{i=1}^{N} x_{ij} = 1, \quad j = 1,\ldots,N, \\ & \sum_{j=1}^{N} x_{ij} = 1, \quad i = 1,\ldots,N, \\ & x_{ij} \in \{0,1\}, \quad i,j = 1,\ldots,N, \end{cases} \tag{6}$$

where c_{ijkl} denotes the cost associated with assigning object i to position j and object k to position l for $i, j, k, l = 1, \ldots, N$.

An energy function for this problem can be written as follows [8]:

$$E(S) = \frac{A}{2}\sum_{i=1}^{n}\sum_{j\neq i}\sum_{k=1}^{m}\sum_{l\neq k} c_{ijkl}S_{ik}S_{jl} + \frac{B}{2}\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{l\neq k} S_{ik}S_{il} + \frac{C}{2}\sum_{k=1}^{m}\left(1-\sum_{i=1}^{n} S_{ik}\right)^2. \tag{7}$$

The first term is the just cost function. The second term specifies the constraint that at most one position can be assigned to each object and the third term specifies the constraint that every object must be assigned by exactly one assignment. This term prevents the solution of no assignments. The QAP was studied in ([6,7,8,50]).

Graph Partitioning Problem

The graph partitioning (GP) problem can be defined as follows: Given a set of N nodes with a given connectivity, partition them into K sets each with N/K nodes such that the net connectivity (cut-size) is minimal between each set. If $K = 2$, then this problem is known as the *graph bipartitioning problem*.

An energy function for this problem can be written as follows [42]: For each vertex i, a binary unit $S_i = +1$ or -1 is assigned, and for each pair of vertices S_i, S_j, $i \neq$

j, a value $T_{ij} = 1$ if they are connected, and $T_{ij} = 0$ if they are not connected is assigned:

$$E(S) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}S_iS_j + \frac{\alpha}{2}\left(\left(\sum_{i=1}^{N} S_i\right)^2 - \sum_{i=1}^{N} S_i^2\right), \tag{8}$$

where α is an imbalance parameter. The first term corresponds to the cost function and the second term corresponds to the constraints of the problem that guarantee equal partition. $T_{ij}S_iS_j$ is 0 whenever vertices i and j are not connected at all, positive whenever connected vertices i and j are in the same partition and negative when they are in separate partition. The GP was studied in ([5,6,38,40,41,42,44,49]). The graph bipartitioning was studied in ([4,6,36,40,41,42]). The graph K-partitioning was studied in ([13,44]).

Traveling Salesman Problem

The traveling salesman problem (TSP) can be defined as follows: Given a list of N cities and the distances, d_{ij}, among them, find the shortest possible tour through a set of N cities, visiting each one exactly once. Note that the TSP is a special case of the GP problem where $K = N$. For the n-city TSP, the number of possible tours are $n!$. However, a tour describes an order in which cities are visited. For the n-city TSP, there are $2n$ tours of equal path-length. Therefore, there are $2!/2n$ distinct paths for closed TSP tours. The total of $N = n^2$ neurons required to map the n-city TSP to NNs. To enable the N neurons in the TSP network, to compute a solution to the problem, the network must be described by an energy function in which the lowest energy state correspond to the best path. Hopfield and Tank [15] used the following energy function to solve the TSP problem. The output S_{ij} indicates whether city i is assigned to position j in the tour or not.

$$E(S) = \frac{A}{2}\sum_{X}^{N}\sum_{i}^{N}\sum_{j\neq i}^{N} S_{Xi}S_{Xj} + \frac{B}{2}\sum_{i}^{N}\sum_{X}^{N}\sum_{X\neq Y}^{N} S_{Xi}S_{Yi} + \frac{C}{2}\left(\sum_{X}^{N}\sum_{i}^{N} S_{Xi} - N\right)^2 + \frac{D}{2}\sum_{X}^{N}\sum_{X\neq Y}^{N}\sum_{i}^{N} d_{XY}S_{Xi}(S_{Y,i+1} + S_{Y,i-1}). \tag{9}$$

The first three terms characterize the general problem constraints. The first triple sum is zero if and only if each city row X contains no more than '1', (the rest of the entries being zero). The second triple sum is zero if and only if each 'position in tour' column contains no more than '1', (the rest of the entries being zero). The third triple sum is zero if and only if there are n entries of '1' in the entire matrix. The last triple sum is the cost term, the length of the path corresponding to a given tour. The TSP was studied in ([6,15,17,18,26,27,35,40,41,42,43,50]).

In addition to these problems, the following problems have been also studied:

- The Airline Crew Scheduling [21].
- The Constraint Satisfaction Problem [26].
- The Generalized Assignment Problem ([7,33]).
- The Graph Coloring Problem [44].
- The Job Scheduling [7].
- The Knapsack Problem ([1,7,32,33,36,41,42]).
- The Maximum Clique Problem ([13,22,23,44]).
- The Maximum Independent Set Problem [44].
- The Maximum Matching Problem ([13,44]).
- The Morphism Problems [13].
- The Number Partitioning [44].
- The Satellite Broadcasting Scheduling [3].
- The School Scheduling ([41,42]).
- The Set Cover Problem [44].
- The Vertex Cover Problem [44].

Conclusions

The use of NNs for the combinatorial optimization problems and the NN approaches that have been applied to combinatorial optimization problems were reviewed. These approaches suggest that NNs are an alternative for solving combinatorial optimization problems as compared to other optimization techniques. Motivations for using NNs include the improvement in the speed of the operation through massively parallel computation, and possible hardware design advan-

tages. One of the main advantages of NN approaches to classical optimization approaches is the inherently parallel and distributed nature of the dynamic solution procedure. Therefore, NNs are capable to solve large scale optimization problems in real time. This is essential in many engineering design, control, and optimization problems. Because of the inherent nature of parallel and distributed information processing in NNs, they are promising computational models for solving large scale optimization problems in real-time.

See also

- ► Neuro-dynamic Programming
- ► Replicator Dynamics in Combinatorial Optimization
- ► Unconstrained Optimization in Neural Network Training

References

1. Abe S, Kawakami J, Hirasawa K (1992) Solving inequality constrained combinatorial optimization problems by the Hopfield neural networks. Neural Networks 5:663–670
2. Ackley DH, Hinton GE, Sejnowski TJ (1985) A learning algorithm for Boltzman machines. Cognitive Sci 9:147–169
3. Ansari N, Hou ESH, Yu Y (1995) A new method to optimize the satellite broadcasting schedules using the mean field annealing of a Hopfield neural network. IEEE Trans Neural Networks 6(2):470–482
4. Bilbro G, Mann R, Miller TK, Snyder WE, VanDenBout DE, White M (1988) Optimization by mean field annealing. In: Touretzky DS (eds) Proc. Annual Conf. Advances in Neural Information Processing Systems. Morgan Kaufmann, San Mateo, pp 91–98
5. Bilbro GL, Snyder WE, Garnier SJ, Gault JW (1992) Mean field annealing formalism for constructing GNC-like algorithms. IEEE Trans Neural Networks 3(1):131–138
6. Cichocki A, Unbehauen R (1993) Neural networks for optimization and signal processing. Wiley, New York
7. Fang L, Li T (1990) Design of competition-based neural networks for combinatorial optimization. Internat J Neural Systems 1(3):221–235
8. Fang L, Wilson WH, Li T (1990) Mean field annealing neural net for quadratic assignment. In: Internat. Neural Network Conf., Paris, July 9–13. Kluwer, Dordrecht, pp 282–286
9. Fausett L (1994) Fundamentals of neural networks: Architectures, algorithms, and applications. Prentice-Hall, Englewood Cliffs
10. Fischer KH, Hertz JA (1991) Spin glasses. Cambridge Univ Press, Cambridge
11. Glauber RJ (1963) Time-dependent statistics of the Ising model. J Math Phys 4(2):294–307
12. Haykin S (1994) Neural networks: A comprehensive foundation. MacMillan College Publ., New York
13. Herault L, Niez JJ (1991) Neural networks and combinatorial optimization: A study of NP-complete graph problems. In: Gelenbe E (eds) Neural Networks: Advances and Applications. Elsevier, Amsterdam, pp 165–213
14. Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. In: Proc Natl Acad Sci USA, Biophysics, pp 2554–2558
15. Hopfield JJ, Tank DW (1985) Neural computation of decisions in optimization problems. Biol Cybern 52:141–152
16. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht
17. Igarashi H (1993) A solution to combinatorial optimization problems using a two-layer random field model: Mean-field approximation. In: World Congress on Neural Networks, Portland, July 11-15. Lawrence Erlbaum Ass., Philadelphia, pp 283–286
18. Igarashi H (1994) A solution for combinatorial optimization problems using a two-layer random field model: Mean-field approximation. Syst Comput Jpn 25(8):61–71
19. Kanter I, Sompolinsky H (1987) Graph optimization problems and the Potts glass. J Phys A: Math Gen 20:L673–L679
20. Kurita N, Funahashi K-I (1996) On the Hopfield neural networks and mean field theory. Neural Networks 9(9):1531–1540
21. Lagerholm M, Peterson C, Söderberg B (1997) Airline crew scheduling with Potts neurons. Neural Comput 9:1589–1599
22. LeeK-C, Takefuji Y (1996) Maximum clique problems: Part 1. In: Takefuji Y, Wang J (eds) Neural Computing for Optimization and Combinatorics. World Sci., Singapore, pp 31–61
23. LeeK-C, Takefuji Y (1996) Maximum clique problems: Part 2. In: Takefuji Y, Wang J (eds) Neural Computing for Optimization and Combinatorics. World Sci., Singapore, pp 63–77
24. Levy BC, Adams MB (1987) Global optimization with stochastic neural networks. In: Neural Networks for Optimization and Signal Processing, Proc. First Internat. Conf. on Neural Networks, San Diego, pp 681–690
25. Little WA (1974) The existence of persistent states in the brain. Math Biosci 19:101–120
26. LooiC-K (1992) Neural network methods in combinatorial optimization. Comput Oper Res 19(3–4):191–208
27. Matsuda S (1996) Set-theoretic comparison of mapping of combinatorial optimization problems to Hopfield neural networks. Syst Comput Jpn 27(6):45–59
28. McCullough WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophysics 5:115–133
29. Melamed II (1994) Neural networks and combinatorial optimization. Automation and Remote Control 55(11):1553–1584

30. Mezard M, Parisi G, Virasoro MA (1987) Spin glass theory and beyond. World Sci., Singapore
31. Muller B, Reinhardt J (1990) Neural networks: An introduction. Springer, Berlin
32. Ohlsson M, Peterson C, Söderberg B (1993) Neural networks for optimization problems with inequality constraints: The knapsack problem. Neural Computation 5:331–339
33. Ohlsson M, Pi H (1997) A study of the mean field approach to knapsack problems. Neural Networks 10(2):263–271
34. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization: Algorithms and complexity. Prentice-Hall, Englewood Cliffs
35. Peterson C (1990) Parallel distributed approaches to combinatorial optimization: Benchmark studies on traveling salesman problem. Neural Computation 2:261–269
36. Peterson C (1993) Solving optimization problems with mean field methods. Phys A 200:570–580
37. Peterson C, Anderson JR (1987) A mean field theory learning algorithm for neural networks. Complex Systems 1:995–1019
38. Peterson C, Anderson JR (1988) Neural networks and NP-complete optimization problems; a performance study on the graph bisection problem. Complex Systems 2:59–89
39. Peterson C, Hartman E (1989) Explorations of the mean field theory learning algorithm. Neural Networks 2:475–494
40. Peterson C, Söderberg B (1989) A new method for mapping optimization problems onto neural networks. Internat J Neural Systems 1(1):3–22
41. Peterson C, Söderberg B (1993) Artificial neural networks. In: Reeves CR (eds) Modern heuristic techniques for combinatorial problems. Wiley, New York, pp 197–242
42. Peterson C, Söderberg B (1997) Artificial neural networks. In: Aarts E, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, New York, pp 177–213
43. Qian F, Hirata H (1994) A parallel computation based on mean field theory for combinatorial optimization and Boltzman machines. Syst Comput Jpn 25(12):86–97
44. Ramanujam J, Sadayappan P (1995) Mapping combinatorial optimization problems onto neural networks. Inform Sci 82:239–255
45. Reeves CR (1993) Modern heuristic techniques for combinatorial problems. Wiley, New York
46. Reichl LE (1980) A modern course in statistical physics. Univ Texas Press, Austin
47. Uesaka Y (1993) Mathematical basis of neural networks for combinatorial optimization problems. Optoelectronics - Devices and Technologies 8(1):1–9
48. Urahama K (1995) Mathematical programming formulation for neural combinatorial optimization algorithms. Electron Comm Jpn 78(9):67–75
49. VanDenBout DE, Miller TK (1990) Graph partitioning using annealed networks. IEEE Trans Neural Networks 1(2):192–203
50. Wang J (1994) Deterministic neural networks for combinatorial optimization. In: Omidvar OM (eds) Progress in Neural Networks. Ablex, Stamford, CT, pp 319–340
51. Wang J (1996) Recurrent neural networks for optimization. In: Chen CH (eds) Fuzzy Logic and Neural Network Handbook. McGraw-Hill, New York
52. Wu FY (1982) The Potts model. Rev Modern Phys 54(1):235–268

Neuro-Dynamic Programming

NDP

DIMITRI P. BERTSEKAS
Labor. Information and Decision Systems,
Massachusetts Institute Technol., Cambridge, USA

MSC2000: 90C39

Article Outline

Keywords

Dynamic programming; Optimization; Reinforcement learning; Simulation; Neural networks

Neuro-dynamic programming (NDP for short) is a relatively new class of dynamic programming methods for control and sequential decision making under uncertainty. These methods have the potential of dealing with problems that for a long time were thought to be intractable due to either a large state space or the lack of an accurate model. The methods discussed combine ideas from the fields of neural networks, artificial intelligence, cognitive science, simulation, and approximation theory. In this article, we delineate the major conceptual issues, we survey a number of recent developments, we describe some computational experience, and we address a number of open questions.

We consider systems where decisions are made in stages. The outcome of each decision is not fully predictable but can be anticipated to some extent before the next decision is made. Each decision results in some immediate cost but also affects the context in which future decisions are to be made and therefore affects the cost incurred in future stages. *Dynamic programming* (DP for short) provides a mathematical formalization of the trade-off between immediate and future costs.

Generally, in DP formulations there is a discrete-time dynamic system whose state evolves according to given transition probabilities that depend on a decision/control u. In particular, if we are in state i and we choose decision u, we move to state j with given probability $p_{ij}(u)$. Simultaneously with this transition, we incur a cost $g(i, u, j)$. In comparing, however, the available decisions u, it is not enough to look at the magnitude of the cost $g(i, u, j)$; we must also take into account how desirable the next state j is. We thus need a way to rank or rate states j. This is done by using the optimal cost (over all remaining stages) starting from state j, which is denoted by $J^*(j)$. These costs can be shown to satisfy some form of *Bellman's equation*

$$J^*(i) = \min_u \mathsf{E}\left\{g(i, u, j) + J^*(j) | i, u\right\} ,$$
$$\text{for all } i ,$$

where j is the state subsequent to i, and $\mathsf{E}\{\cdot|i, u\}$ denotes expected value with respect to j, given i and u. Generally, at each state i, it is optimal to use a control u that attains the minimum above. Thus, decisions are ranked based on the sum of the expected cost of the present period, and the optimal expected cost of all subsequent periods.

The objective of DP is to calculate numerically the optimal cost function J^*. This computation can be done off-line, i. e., before the real system starts operating. An optimal policy, that is, an optimal choice of u for each i, is computed either simultaneously with J^*, or in real time by minimizing in the right-hand side of Bellman's equation. It is well known, however, that for many important problems the computational requirements of DP are overwhelming, mainly because of a very large number of states and controls (Bellman's 'curse of dimensionality'). In such situations a suboptimal solution is required.

Cost Approximations in Dynamic Programming

NDP methods are suboptimal methods that center around the approximate evaluation of the optimal cost function J^*, possibly through the use of neural networks and/or simulation. In particular, we replace the optimal cost $J^*(j)$ with a suitable approximation $\widetilde{J}(j, r)$, where r is a vector of parameters, and we use at state i the (suboptimal) control $\widetilde{\mu}(i)$ that attains the minimum in the (approximate) right-hand side of Bellman's equation

$$\widetilde{\mu}(i) = \arg\min_u \mathsf{E}\left\{g(i, u, j) + \widetilde{J}(j, r) | i, u\right\} .$$

The function $\widetilde{J}$ will be called the *scoring function*, and the value $\widetilde{J}(j, r)$ will be called the *score* of state j. The general form of $\widetilde{J}$ is known and is such that once the parameter vector r is determined, the evaluation of $\widetilde{J}(j, r)$ of any state j is fairly simple.

We note that in some problems the minimization over u of the expression

$$\mathsf{E}\left\{g(i, u, j) + \widetilde{J}(j, r) | i, u\right\}$$

may be too complicated or too time-consuming for making decisions in real-time, even if the scores $\widetilde{J}(j, r)$ are simply calculated. In such problems we may use a related technique, whereby we approximate the expression minimized in Bellman's equation,

$$Q(i, u) = \mathsf{E}\left\{g(i, u, j) + J^*(j) | i, u\right\} ,$$

which is known as the Q-factor corresponding to (i, u). In particular, we replace $Q(i, u)$ with a suitable approximation $\widetilde{Q}(i, u, r)$, where r is a vector of parameters. We then use at state i the (suboptimal) control that minimizes the approximate Q-factor corresponding to i:

$$\widetilde{\mu}(i) = \arg\min_u \widetilde{Q}(i, u, r) .$$

Much of what will be said about approximation of the optimal cost function also applies to approximation of Q-factors. We thus focus primarily on approximation of the optimal cost function J^*.

We are interested in problems with a large number of states and in scoring functions $\widetilde{J}$ that can be described with relatively few numbers (a vector r of small dimension). Scoring functions involving few parameters are called *compact representations*, while the tabular description of J^* are called the *lookup table represen-*

tation. Thus, in a lookup table representation, the values $J^*(j)$ are stored in a table for all states j. In a typical compact representation, only the vector r and the general structure of the scoring function $\widetilde{J}(\cdot, r)$ are stored; the scores $\widetilde{J}(j, r)$ are generated only when needed. For example, $\widetilde{J}(j, r)$ may be the output of some neural network in response to the input j, and r is the associated vector of weights or parameters of the neural network; or $\widetilde{J}(j, r)$ may involve a lower-dimensional description of the state j in terms of its 'significant features', and r is the associated vector of relative weights of the features. Thus determining the scoring function $\widetilde{J}(j, r)$ involves two complementary issues:

1) deciding on the general structure of the function $\widetilde{J}(j, r)$; and
2) calculating the parameter vector r so as to minimize in some sense the error between the functions $J^*(\cdot)$ and $\widetilde{J}(\cdot, r)$.

Approximations of the optimal cost function have been used in the past in a variety of DP contexts. Chess playing programs represent a successful example. A key idea in these programs is to use a position evaluator to rank different chess positions and to select at each turn a move that results in the position with the best rank. The *position evaluator* assigns a numerical value to each position, according to a heuristic formula that includes weights for the various features of the position (material balance, piece mobility, king safety, and other factors). Thus, the position evaluator corresponds to the scoring function $\widetilde{J}(j, r)$ above, while the weights of the features correspond to the parameter vector r. Usually, some general structure of position evaluator is selected (this is largely an art that has evolved over many years, based on experimentation and human knowledge about chess), and the numerical weights are chosen by trial and error or (as in the case of the champion program Deep Thought) by 'training' using a large number of sample grandmaster games.

As the chess program paradigm suggests, intuition about the problem, heuristics, and trial and error are all important ingredients for constructing cost approximations in DP. However, it is important to supplement heuristics and intuition with more systematic techniques that are broadly applicable and retain as much as possible the nonheuristic aspects of DP.

NDP aims to develop a methodological foundation for combining dynamic programming, compact representations, and simulation to provide the basis for a rational approach to complex stochastic decision problems.

Approximation Architectures

An important issue in function approximation is the *selection of architecture*, that is, the choice of a parametric class of functions $\widetilde{J}(\cdot, r)$ or $\widetilde{Q}(\cdot, \cdot, r)$ that suits the problem at hand. One possibility is to use a neural network architecture of some type. We should emphasize here that in this article we use the term 'neural network' in a very broad sense, essentially as a synonym to 'approximating architecture'. In particular, we do not restrict ourselves to the classical multilayer perceptron structure with sigmoidal nonlinearities. Any type of universal approximator of nonlinear mappings could be used in our context. The nature of the approximating structure is left open in our discussion, and it could involve, for example, radial basis functions, wavelets, polynomials, splines, etc.

Cost approximation can often be significantly enhanced through the use of *feature extraction*, a process that maps the state i into some vector $f(i)$, called the *feature vector* associated with the state i. Feature vectors summarize, in a heuristic sense, what are considered to be important characteristics of the state, and they are very useful in incorporating the designer's prior knowledge or intuition about the problem and about the structure of the optimal controller. For example in a queueing system involving several queues, a feature vector may involve for each queue a three-value indicator, that specifies whether the queue is 'nearly empty', 'moderately busy', or 'nearly full'. In many cases, analysis can complement intuition to suggest the right features for the problem at hand.

Feature vectors are particularly useful when they can capture the 'dominant nonlinearities' in the optimal cost function J^*. By this we mean that $J^*(i)$ can be approximated well by a 'relatively smooth' function $\widehat{J}(f(i))$; this happens for example, if through a change of variables from states to features, the function J^* becomes a (nearly) linear or low-order polynomial function of the features. When a feature vector can be chosen to have this property, one may consider approximation architectures where both features and (relatively simple) neural networks are used together. In partic-

ular, the state is mapped to a feature vector, which is then used as input to a neural network that produces the score of the state. More generally, it is possible that both the state and the feature vector are provided as inputs to the neural network.

A simple method to obtain more sophisticated approximations, is to partition the statee space into several subsets and construct a separate cost function approximation in each subset. For example, by using a linear or quadratic polynomial approximation in each subset of the partition, one can construct piecewise linear or piecewise quadratic approximations over the entire state space. An important issue here is the choice of the method for partitioning the state space. Regular partitions (e. g., grid partitions) may be used, but they often lead to a large number of subsets and very time-consuming computations. Generally speaking, each subset of the partition should contain 'similar' states so that the variation of the optimal cost over the states of the subset is relatively smooth and can be approximated with smooth functions. An interesting possibility is to use features as the basis for partition. In particular, one may use a more or less regular discretization of the space of features, which induces a possibly irregular partition of the original state space. In this way, each subset of the irregular partition contains states with 'similar features'.

Simulation and Training

Some of the most successful applications of neural networks are in the areas of pattern recognition, nonlinear regression, and nonlinear system identification. In these applications the neural network is used as a universal approximator: the input-output mapping of the neural network is matched to an unknown nonlinear mapping F of interest using a least squares optimization. This optimization is known as *training the network*. To perform training, one must have some training data, that is, a set of pairs $(i, F(i))$, which is representative of the mapping F that is approximated.

It is important to note that in contrast with these neural network applications, in the DP context there is no readily available training set of input-output pairs $(i, J^*(i))$, which can be used to approximate J^* with a least squares fit. The only possibility is to evaluate (exactly or approximately) by simulation the cost functions of given (suboptimal) policies, and to try to iteratively improve these policies based on the simulation outcomes. This creates analytical and computational difficulties that do not arise in classical neural network training contexts. Indeed the use of simulation to evaluate approximately the optimal cost function is a key new idea, that distinguishes the methodology of this presentation from earlier approximation methods in DP.

Using simulation offers another major advantage: it allows the methods of this article to be used for systems that are hard to model but easy to simulate; that is, in problems where an explicit model is not available, and the system can only be observed, either as it operates in real time or through a software simulator. For such problems, the traditional DP techniques are inapplicable, and estimation of the transition probabilities to construct a detailed mathematical model is often cumbersome or impossible.

There is a third potential advantage of simulation: it can implicitly identify the 'most important' or 'most representative' states of the system. It appears plausible that if these states are the ones most often visited during the simulation, the scoring function will tend to approximate better the optimal cost for these states, and the suboptimal policy obtained will perform better.

Neuro-Dynamic Programming

The name 'neuro-dynamic programming' expresses the reliance of the methods of this article on both DP and neural network concepts. In the artificial intelligence community, where the methods originated, the name *reinforcement learning* is also used. In common artificial intelligence terms, the methods allow systems to 'learn how to make good decisions by observing their own behavior, and use built-in mechanisms for improving their actions through a reinforcement mechanism'. In less anthropomorphic DP terms, 'observing their own behavior' relates to simulation, and 'improving their actions through a reinforcement mechanism' relates to iterative schemes for improving the quality of approximation of the optimal cost function, or the Q-factors, or the optimal policy. There has been a gradual realization that reinforcement learning techniques can be fruitfully motivated and interpreted in terms of classical DP concepts such as value and policy iteration; see the survey [1], and the book [6], which

point out the connections between the artificial intelligence/reinforcement learning viewpoint and the control theory/DP viewpoint, and give many references.

The currently most popular methodology in NDP iteratively adjusts the parameter vector r of the scoring function $\widetilde{J}(j, r)$ as it produces sample state trajectories $(i_0, \ldots, i_{k+1}, \ldots,)$ by using simulation. These trajectories correspond to either a fixed stationary policy, or to a 'greedy' policy that applies, at state i, the control u that minimizes the expression

$$\mathsf{E}\left\{g(i,u,j)+\widetilde{J}(j,r)|i,u\right\},$$

where r is the current parameter vector. A central notion here is the notion of a *temporal difference*, defined by

$$d_k = g(i_k,u_k,i_{k+1})+\widetilde{J}(i_{k+1},r)-\widetilde{J}(i_k,r),$$

and expressing the difference between our expected cost estimate $\widetilde{J}(i_k, r)$ at state i_k and the predicted cost estimate $g(i_k, u_k, i_{k+1}) + \widetilde{J}(i_{k+1}, r)$ based on the outcome of the simulation. If the cost approximations were exact, the average temporal difference would be zero by Bellman's equation. Thus, roughly speaking, the values of the temporal differences can be used to make incremental adjustments to r so as to bring about an approximate equality (on the average) between expected and predicted cost estimates along the simulated trajectories. This viewpoint, formalized by R.S. Sutton in [5], can be implemented through the use of gradient descent/stochastic approximation methodology. Sutton proposed a family of methods of this type, called TD(λ), and parameterized by a scalar $\lambda \in [0, 1]$. One extreme, TD(1), is closely related to Monte-Carlo simulation and least squares parameter estimation, while the other extreme, TD(0), is closely related to stochastic approximation. A related method is Q-learning, introduced by C.J.C.H. Watkins [9], which is a stochastic approximation-like method that iterates on the Q-factors. While there is convergence analysis of TD(λ) and Q-learning for the case of lookup table representations (see [8,4]), the situation is much less clear in the case of compact representations. A number of results have been derived for approximate policy and value iteration methods, which are obtained from the traditional DP methods after compact representations of the various cost functions involved are introduced.

While the theoretical support for the NDP methodology is only now emerging, there have been quite a few reports of successes with problems too large and complex to be treated in any other way. A particularly impressive success is the development of a backgammon playing program as reported by G. Tesauro [7]. Here a neural network provided a compact representation of the optimal cost function of the game of backgammon by using simulation and TD(λ). The training was performed by letting the program play against itself. After training for several months, the program nearly defeated the human world champion. Variations of the method used by Tesauro have been used with success in a variety of applications.

The recent experience of several researchers, involving several engineering applications, has confirmed that NDP methods can be impressively effective in problems where traditional DP methods would be hardly applicable and other heuristic methods would have a limited chance of success. We note, however, that the practical application of NDP is computationally very intensive, and often requires a considerable amount of trial and error. Fortunately, all the computation and experimentation with different approaches can be done off-line. Once the approximation is obtained off-line, it can be used to generate decisions fast enough for use in real time. In this context, we mention that in the machine learning literature, reinforcement learning is often viewed as an 'on-line' method, whereby the cost approximation is improved as the system operates in real time. This is reminiscent of the methods of traditional adaptive control.

Extensive references for the material of this article are the research monographs [3,6]. A more limited textbook discussion is given in [2]. The survey [1], and the overviews [10,11], and other papers in the edited volume [12] point out the connections between the artificial intelligence/reinforcement learning viewpoint and the control theory/DP viewpoint, and give many references.

See also

- ► Dynamic Programming: Average Cost Per Stage Problems
- ► Dynamic Programming in Clustering

- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Multiple Objective Dynamic Programming
- ▶ Neural Networks for Combinatorial Optimization
- ▶ Replicator Dynamics in Combinatorial Optimization
- ▶ Unconstrained Optimization in Neural Network Training

References

1. Barto AG, Bradtke SJ, Singh SP (1995) Real-time learning and control using asynchronous dynamic programming. Artif Intell 72:81–138
2. Bertsekas DP (1995) Dynamic programming and optimal control, vol II, Athena Sci., Belmont
3. Bertsekas DP, Tsitsiklis JN (1996) Neuro-dynamic programming. Athena Sci., Belmont
4. Jaakkola T, Jordan MI, Singh SP (1994) On the convergence of stochastic iterative dynamic programming algorithms. Neural Computation 6:1185–1201
5. Sutton RS (1988) Learning to predict by the methods of temporal differences. Machine Learning 3:9–44
6. Sutton RS, Barto AG (1998) Reinforcement learning. MIT, Cambridge
7. Tesauro G (1992) Practical issues in temporal difference learning. Machine Learning 8:257–277
8. Tsitsiklis JN (1994) Asynchronous stochastic approximation and Q-learning. Machine Learning 16:185–202
9. Watkins CJCH (1989) Learning from delayed rewards. PhD Thesis Cambridge Univ, Cambridge
10. Werbös PJ (1992) Approximate dynamic programming for real-time control and neural modeling. In: White DA, Sofge DA (eds) Handbook of Intelligent Control. v. Nostrand, Princeton
11. Werbös PJ (1992) Neurocontrol and supervised learning: An overview and valuation. In: White DA, Sofge DA (eds) Handbook of Intelligent Control. v. Nostrand, Princeton
12. White DA, Sofge DA (eds) (1992) Handbook of Intelligent Control. v. Nostrand, Princeton

New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization

NECULAI ANDREI[1,2]

[1] Center for Advanced Modeling and Optimization, Research Institute for Informatics, Bucharest, Romania

[2] Academy of Romanian Scientists, Bucharest, Romania

MSC2000: 49M07, 49M10, 90C06, 65K

Article Outline

Keywords and Phrases

Unconstrained optimization; Hybrid conjugate gradient method; Conjugacy condition; Numerical comparisons; Dolan–Moré profile

New hybrid conjugate gradient algorithms are proposed and analyzed. In these hybrid algorithms the famous parameter β_k is computed as a convex combination of the Polak–Ribière–Polyak and Dai–Yuan conjugate gradient algorithms. In one hybrid algorithm the parameter in convex combination is computed in such a way that the conjugacy condition is satisfied, independent of the line search. In the other, the parameter in convex combination is computed in such a way that the conjugate gradient direction is the Newton direction. The algorithm uses the standard Wolfe line search conditions. Numerical comparisons with conjugate gradient algorithms using a set of 750 unconstrained optimization problems, some of them from the CUTE library, show that the hybrid computational scheme based on the conjugacy condition outperforms the known hybrid conjugate gradient algorithms.

Introduction

Let us consider the nonlinear unconstrained optimization problem

$$\min \{f(x) : x \in R^n\} , \tag{1}$$

where $f : R^n \to R$ is a continuously differentiable function, bounded from below. For solving this problem, starting from an initial guess $x_0 \in R^n$, a nonlinear conjugate gradient method generates a sequence $\{x_k\}$ as

$$x_{k+1} = x_k + \alpha_k d_k , \tag{2}$$

where $\alpha_k > 0$ is obtained by line search, and the directions d_k are generated as

$$d_{k+1} = -g_{k+1} + \beta_k s_k , \quad d_0 = -g_0 . \tag{3}$$

In (3) β_k is known as the conjugate gradient parameter, $s_k = x_{k+1} - x_k$ and $g_k = \nabla f(x_k)$. Consider $\|.\|$ the Euclidean norm and define $y_k = g_{k+1} - g_k$. The line search in the conjugate gradient algorithms is often based on the standard Wolfe conditions:

$$f(x_k + \alpha_k d_k) - f(x_k) \le \rho \alpha_k g_k^{\mathrm{T}} d_k , \tag{4}$$

$$g_{k+1}^{\mathrm{T}} d_k \ge \sigma g_k^{\mathrm{T}} d_k , \tag{5}$$

where d_k is a descent direction and $0 < \rho \le \sigma < 1$. Plenty of conjugate gradient methods are known, and an excellent survey of these methods, with special attention to their global convergence, was given by Hager and Zhang [19]. Different conjugate gradient algorithms correspond to different choices for the scalar parameter β_k. Some of these methods, such those of Fletcher and Reeves (FR) [16], Dai and Yuan (DY) [12] and conjugate descent (CD) proposed by Fletcher [15], have strong convergence properties, but they may have modest practical performance owing to jamming:

$$\beta_k^{\mathrm{FR}} = \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{g_k^{\mathrm{T}} g_k} , \quad \beta_k^{\mathrm{DY}} = \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k} ,$$
$$\beta_k^{\mathrm{CD}} = \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{-g_k^{\mathrm{T}} s_k} .$$

On the other hand, the methods of Polak and Ribière [23] and Polyak (PRP) [24], Hestenes and Stiefel (HS) [20] or Liu and Storey (LS) [22] in general may not be convergent, but they often have better computational performances:

$$\beta_k^{\mathrm{PRP}} = \frac{g_{k+1}^{\mathrm{T}} y_k}{g_k^{\mathrm{T}} g_k} , \quad \beta_k^{\mathrm{HS}} = \frac{g_{k+1}^{\mathrm{T}} y_k}{y_k^{\mathrm{T}} s_k} , \quad \beta_k^{\mathrm{LS}} = \frac{g_{k+1}^{\mathrm{T}} y_k}{-g_k^{\mathrm{T}} s_k} .$$

In this contribution we focus on hybrid conjugate gradient methods. These methods are combinations of different conjugate gradient algorithms; mainly they are proposed to avoid the jamming phenomenon. One of the first hybrid conjugate gradient algorithms was introduced by Touati-Ahmed and Storey [28], where the parameter β_k is computed as

$$\beta_k^{TS} = \begin{cases} \beta_k^{\mathrm{PRP}} = \frac{g_{k+1}^{\mathrm{T}} y_k}{\|g_k\|^2}, & \text{if } 0 \le \beta_k^{\mathrm{PRP}} \le \beta_k^{\mathrm{FR}}, \\ \beta_k^{\mathrm{FR}} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}, & \text{otherwise.} \end{cases}$$

The PRP method has a built-in restart feature that directly addresses jamming. Indeed, when the step s_k is small, then the factor y_k in the numerator of β_k^{PRP} tends to zero. Therefore, β_k^{PRP} becomes small and the search direction d_{k+1} is very close to the steepest descent direction $-g_{k+1}$. Hence, when the iterations jam, the method of Touati-Ahmed and Storey uses the PRP computational scheme.

Another hybrid conjugate gradient method was given by Hu and Storey [21], where β_k in (3) is

$$\beta_k^{\mathrm{HuS}} = \max \{0, \min \{\beta_k^{\mathrm{PRP}}, \beta_k^{\mathrm{FR}}\}\} .$$

As above, when the method of Hu and Storey is jamming, then the PRP method is used instead.

The combination of LS and CD conjugate gradient methods leads to the following hybrid method:

$$\beta_k^{\mathrm{LS-CD}} = \max \{0, \min \{\beta_k^{\mathrm{LS}}, \beta_k^{\mathrm{CD}}\}\} .$$

The CD method of Fletcher [15] is very similar to the FR method. With an exact line search, the CD method is identical to the FR method. Similarly, for an exact line search, the LS method is identical to the PRP method. Therefore, the hybrid LS–CD method with an exact line search has similar performances as the hybrid method of Hu and Storey.

Gilbert and Nocedal [17] suggested a combination between PRP and FR methods as

$$\beta_k^{\mathrm{GN}} = \max \{-\beta_k^{\mathrm{FR}}, \min \{\beta_k^{\mathrm{PRP}}, \beta_k^{\mathrm{FR}}\}\} .$$

Since β_k^{FR} is always nonnegative, it follows that β_k^{GN} can be negative. The method of Gilbert and Nocedal has the same advantage of avoiding jamming.

Using the standard Wolfe line search, the DY method always generates descent directions and if the gradient is Lipschitz-continuous the method is globally convergent. In an effort to improve their algorithm, Dai and Yuan [13] combined their algorithm with other conjugate gradient algorithms, and proposed the following two hybrid methods:

$$\beta_k^{\mathrm{hDY}} = \max\left\{-c\beta_k^{\mathrm{DY}}, \min\left\{\beta_k^{\mathrm{HS}}, \beta_k^{\mathrm{DY}}\right\}\right\} ,$$

$$\beta_k^{\mathrm{hDYz}} = \max\left\{0, \min\left\{\beta_k^{\mathrm{HS}}, \beta_k^{\mathrm{DY}}\right\}\right\} ,$$

where $c = (1-\sigma)/(1+\sigma)$. For the standard Wolfe conditions (4) and (5), under the Lipschitz continuity of the gradient, Dai and Yuan [13] established the global convergence of these hybrid computational schemes.

In the following we propose another hybrid conjugate gradient as a convex combination of PRP and DY conjugate gradient algorithms. We selected these two methods for combination in a hybrid conjugate gradient algorithm because the PRP algorithm has good computational properties, on one hand, and the DY algorithm has strong convergence properties, on the other hand. Often the PRP method performs better in practice than the DY method and we speculate this in order to have a good practical conjugate algorithm. The structure of this chapter is as follows. In Sect. "New Hybrid Conjugate Gradient Algorithms" we introduce our hybrid conjugate gradient algorithm and prove that it generates descent directions satisfying in some conditions the sufficient descent condition. Section "The New Hybrid Algorithms (CCOMB, NDOMB)" presents the algorithms and in Sect. "Convergence Analysis" we show the convergence analysis. In Sect. "Numerical Experiments" some numerical experiments and performance profiles of Dolan–Moré [14] corresponding to this new hybrid conjugate gradient algorithm and some other conjugate gradient algorithms are presented. The performance profiles corresponding to a set of 750 unconstrained optimization problems in the CUTE test problem library [6] as well as some other unconstrained optimization problems presented in [1] show that this hybrid conjugate gradient algorithm outperforms the known hybrid conjugate gradient algorithms.

New Hybrid Conjugate Gradient Algorithms

The iterates $x_0, x_1, x_2, \ldots$ of our algorithm are computed by means of recurrence (2) where the step size $\alpha_k > 0$ is determined according to the Wolfe conditions (4) and (5), and the directions d_k are generated by the rule:

$$d_{k+1} = -g_{k+1} + \beta_k^{\mathrm{N}} s_k , \qquad d_0 = -g_0 , \tag{6}$$

where

$$\begin{aligned}\beta_k^{\mathrm{N}} &= (1-\theta_k)\beta_k^{\mathrm{PRP}} + \theta_k \beta_k^{\mathrm{DY}} \\ &= (1-\theta_k)\frac{g_{k+1}^{\mathrm{T}} y_k}{g_k^{\mathrm{T}} g_k} + \theta_k \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k}\end{aligned} \tag{7}$$

and θ_k is a scalar parameter satisfying $0 \le \theta_k \le 1$, which needs to be determined. Observe that if $\theta_k = 0$, then $\beta_k^{\mathrm{N}} = \beta_k^{\mathrm{PRP}}$, and if $\theta_k = 1$, then $\beta_k^{\mathrm{N}} = \beta_k^{\mathrm{DY}}$. On the other hand, if $0 < \theta_k < 1$, then β_k^{N} is a convex combination of β_k^{PRP} and β_k^{DY}.

Referring to the PRP method, Polak and Ribière [23] proved that when function f is strongly convex and the line search is exact, then the PRP method is globally convergent. In an effort to understand the behavior of the PRP method, Powell [25] showed that if the step length $s_k = x_{k+1} - x_k$ approaches zero, the line search is exact and the gradient $\nabla f(x)$ is Lipschitz-continuous, then the PRP method is globally convergent. Additionally, assuming that the search direction is a descent direction, Yuan [29] established the global convergence of the PRP method for strongly convex functions and a Wolfe line search. For general nonlinear functions the convergence of the PRP method is uncertain. Powell [26] gave a three-dimensional example, in which the function to be minimized is not strongly convex, showing that even with an exact line search the PRP method may not converge to a stationary point. Later on Dai [7] presented another example, this time with a strongly convex function for which the PRP method fails to generate a descent direction. Therefore, theoretically the convergence of the PRP method is limited to strongly convex functions. For general nonlinear functions the convergence of the PRP method is established under restrictive conditions (Lipschitz continuity, exact line search and the step size tends to zero). However, the numerical experiments presented, for example, by Gilbert and Nocedal [17] proved that the PRP method is one of the best conjugate

gradient methods, and this was the main motivation to consider it in (7).

On the other hand, the DY method always generates descent directions, and in [8] Dai established a remarkable property for the DY conjugate gradient algorithm, relating the descent directions to the sufficient descent condition. It was shown that if there exist constants γ_1 and γ_2 such that $\gamma_1 \le \|g_k\| \le \gamma_2$ for all k, then for any $p \in (0,1)$ there exists a constant $c > 0$ such that the sufficient descent condition $g_i^{\mathrm{T}} d_i \le -c \|g_i\|^2$ holds for at least $\lfloor pk \rfloor$ indices $i \in [0,k]$, where $\lfloor j \rfloor$ denotes the largest integer $\le j$. Therefore, this property is the main reason we consider the DY method in (7).

It easy to see that

$$d_{k+1} = -g_{k+1} + (1-\theta_k)\frac{y_k^{\mathrm{T}} g_{k+1}}{g_k^{\mathrm{T}} g_k} s_k + \theta_k \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k} s_k . \tag{8}$$

Supposing that d_k is a descent direction ($d_0 = -g_0$), then for the algorithm given by (2) and (8) we can prove the following result.

Theorem 1 *Assume that α_k in algorithm (2) and (8) is determined by Wolfe line search (4) and (5). If $0 < \theta_k < 1$, and*

$$\left| \frac{g_k^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} \right| \|g_{k+1}\|^2 \ge \frac{(g_{k+1}^{\mathrm{T}} y_k)(g_{k+1}^{\mathrm{T}} s_k)}{\|g_k\|^2} , \tag{9}$$

then direction d_{k+1} given by (8) is a descent direction.

Proof Since $0 < \theta_k < 1$, from (8) we get

$$\begin{aligned} g_{k+1}^{\mathrm{T}} d_{k+1} &= -\|g_{k+1}\|^2 + (1-\theta_k)\frac{y_k^{\mathrm{T}} g_{k+1}}{g_k^{\mathrm{T}} g_k} g_{k+1}^{\mathrm{T}} s_k \\ &\quad + \theta_k \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k} g_{k+1}^{\mathrm{T}} s_k \\ &\le -\|g_{k+1}\|^2 + \frac{y_k^{\mathrm{T}} g_{k+1}}{g_k^{\mathrm{T}} g_k} g_{k+1}^{\mathrm{T}} s_k \\ &\quad + \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k} g_{k+1}^{\mathrm{T}} s_k \\ &= \left(-1 + \frac{g_{k+1}^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} \right) \|g_{k+1}\|^2 \\ &\quad + \frac{y_k^{\mathrm{T}} g_{k+1}}{g_k^{\mathrm{T}} g_k} g_{k+1}^{\mathrm{T}} s_k \\ &= \frac{g_k^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} \|g_{k+1}\|^2 + \frac{y_k^{\mathrm{T}} g_{k+1}}{g_k^{\mathrm{T}} g_k} g_{k+1}^{\mathrm{T}} s_k . \end{aligned}$$

But, $y_k^{\mathrm{T}} s_k > 0$ by (5) and since $g_k^{\mathrm{T}} s_k \le 0$, it follows that

$$\frac{g_k^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} \|g_{k+1}\|^2 \le 0 .$$

Therefore, from (9), it follows that $g_{k+1}^{\mathrm{T}} d_{k+1} \le 0$, i. e., the direction d_{k+1} is a descent one. □

Theorem 2 *Suppose that $(g_{k+1}^{\mathrm{T}} y_k)(g_{k+1}^{\mathrm{T}} s_k) \le 0$. If $0 < \theta_k < 1$, then the direction d_{k+1} given by (8) satisfies the sufficient descent condition*

$$g_{k+1}^{\mathrm{T}} d_{k+1} \le -\left(1 - \theta_k \frac{g_{k+1}^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} \right) \|g_{k+1}\|^2 . \tag{10}$$

Proof From (8) we have

$$\begin{aligned} g_{k+1}^{\mathrm{T}} d_{k+1} &= -\|g_{k+1}\|^2 + (1-\theta_k)\frac{g_{k+1}^{\mathrm{T}} y_k}{g_k^{\mathrm{T}} g_k} g_{k+1}^{\mathrm{T}} s_k \\ &\quad + \theta_k \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k} g_{k+1}^{\mathrm{T}} s_k \\ &= -\|g_{k+1}\|^2 + \theta_k \frac{g_{k+1}^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} \|g_{k+1}\|^2 \\ &\quad + (1-\theta_k)\frac{(g_{k+1}^{\mathrm{T}} y_k)(g_{k+1}^{\mathrm{T}} s_k)}{g_k^{\mathrm{T}} g_k} \\ &\le -\left(1 - \theta_k \frac{g_{k+1}^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} \right) \|g_{k+1}\|^2 \le 0 . \end{aligned}$$

Observe that since $y_k^{\mathrm{T}} s_k > 0$ by (5) and since $g_{k+1}^{\mathrm{T}} s_k = y_k^{\mathrm{T}} s_k + g_k^{\mathrm{T}} s_k < y_k^{\mathrm{T}} s_k$, then $y_k^{\mathrm{T}} s_k / g_{k+1}^{\mathrm{T}} s_k > 1$. Therefore, if $0 < \theta_k < 1$, it follows that $\theta_k < y_k^{\mathrm{T}} s_k / g_{k+1}^{\mathrm{T}} s_k$. Therefore,

$$1 - \theta_k \frac{g_{k+1}^{\mathrm{T}} s_k}{y_k^{\mathrm{T}} s_k} > 0 ,$$

proving the theorem. □

To select the parameter θ_k we consider the following two possibilities. In the first hybrid conjugate gradient algorithm the parameter θ_k is selected in such a manner that the *conjugacy condition* $y_k^{\mathrm{T}} d_{k+1} = 0$ is satisfied at every iteration, independent of the line search. Hence,

from $y_k^{\mathrm{T}} d_{k+1} = 0$ after some algebra, using (8), we get

$$\theta_k = \theta_k^{\mathrm{CCOMB}} \equiv \frac{(y_k^{\mathrm{T}} g_{k+1})(y_k^{\mathrm{T}} s_k) - (y_k^{\mathrm{T}} g_{k+1})(g_k^{\mathrm{T}} g_k)}{(y_k^{\mathrm{T}} g_{k+1})(y_k^{\mathrm{T}} s_k) - \|g_{k+1}\|^2 \|g_k\|^2} . \tag{11}$$

In the second algorithm the parameter θ_k is selected in such a manner that the direction d_{k+1} from (8) is the Newton direction, i. e.,

$$-\nabla^2 f(x_{k+1})^{-1} g_{k+1} = -g_{k+1} + (1-\theta_k)\frac{y_k^{\mathrm{T}} g_{k+1}}{g_k^{\mathrm{T}} g_k} s_k + \theta_k \frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k} s_k . \tag{12}$$

Having in view that $\nabla^2 f(x_{k+1}) s_k = y_k$, from (12) we get

$$\theta_k = \theta_k^{\mathrm{NDOMB}} \equiv \frac{(y_k^{\mathrm{T}} g_{k+1} - s_k^{\mathrm{T}} g_{k+1})\|g_k\|^2 - (g_{k+1}^{\mathrm{T}} y_k)(y_k^{\mathrm{T}} s_k)}{\|g_{k+1}\|^2 \|g_k\|^2 - (g_{k+1}^{\mathrm{T}} y_k)(y_k^{\mathrm{T}} s_k)} . \tag{13}$$

Observe that the parameter θ_k given by (11) or (13) can be outside the interval $[0, 1]$. However, in order to have a real convex combination in (7) the following rule is considered: if $\theta_k \le 0$, then set $\theta_k = 0$ in (7), i. e., $\beta_k^{\mathrm{N}} = \beta_k^{\mathrm{PRP}}$; if $\theta_k \ge 1$, then take $\theta_k = 1$ in (7), i. e., $\beta_k^{\mathrm{N}} = \beta_k^{\mathrm{DY}}$. Therefore, under this rule for θ_k selection, the direction d_{k+1} in (8) combines the properties of the PRP and DY algorithms.

The New Hybrid Algorithms (CCOMB, NDOMB)

Step 1 *Initialization.* Select $x_0 \in R^n$ and the parameters $0 < \rho \le \sigma < 1$. Compute $f(x_0)$ and g_0. Consider $d_0 = -g_0$ and set the initial guess: $\alpha_0 = 1/\|g_0\|$.

Step 2 *Test for continuation of iterations.* If $\|g_k\|_\infty \le 10^{-6}$, then stop.

Step 3 *Line search.* Compute $\alpha_k > 0$ satisfying the Wolfe line search conditions (4) and (5) and update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, g_{k+1} and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Step 4. θ_k *parameter computation.* If $(y_k^{\mathrm{T}} g_{k+1})(y_k^{\mathrm{T}} s_k) - \|g_{k+1}\|^2 \|g_k\|^2 = 0$, then set $\theta_k = 0$, otherwise compute θ_k as follows:

CCOMB algorithm (θ_k from the conjugacy condition): $\theta_k = \theta_k^{\mathrm{CCOMB}}$.

NDOMB algorithm (θ_k from the Newton direction): $\theta_k = \theta_k^{\mathrm{NDOMB}}$.

Step 5 β_k^{N} *conjugate gradient parameter computation.* If $0 < \theta_k < 1$, then compute β_k^{N} as in (7). If $\theta_k \ge 1$, then set $\beta_k^{\mathrm{N}} = \beta_k^{\mathrm{DY}}$. If $\theta_k \le 0$, then set $\beta_k^{\mathrm{N}} = \beta_k^{\mathrm{PRP}}$.

Step 6 *Direction computation.* Compute $d = -g_{k+1} + \beta_k^{\mathrm{N}} s_k$. If the restart criterion of Powell,

$$\left|g_{k+1}^{\mathrm{T}} g_k\right| \ge 0.2 \|g_{k+1}\|^2 , \tag{14}$$

is satisfied, then set $d_{k+1} = -g_{k+1}$; otherwise define $d_{k+1} = d$. Compute the initial guess $\alpha_k = \alpha_{k-1}\|d_{k-1}\| / \|d_k\|$, set $k = k+1$ and continue with step 2.

It is well known that if f is bounded along the direction d_k then there exists a step size α_k satisfying the Wolfe line search conditions (4) and (5). In our algorithm when the Powell restart condition is satisfied, we restart the algorithm with the negative gradient $-g_{k+1}$. More sophisticated reasons for restarting the algorithms have been proposed in the literature [10], but we are interested in the performance of a conjugate gradient algorithm that uses this restart criterion, associated with a direction satisfying the conjugacy condition or that is equal to the Newton direction. Under reasonable assumptions, conditions (4), (5) and (14) are sufficient to prove the global convergence of the algorithm. We consider this aspect in the next section.

The first trial of the step length crucially affects the practical behavior of the algorithm. At every iteration $k \ge 1$ the starting guess for step α_k in the line search is computed as $\alpha_{k-1}\|d_{k-1}\|_2 / \|d_k\|_2$. This selection was considered for the first time by Shanno and Phua [27] in CONMIN. It is also considered in the packages SCG by Birgin and Martínez [5] and in SCALCG by Andrei [2,3,4].

Convergence Analysis

Throughout this section we assume that

1. *The level set* $S = \{x \in R^n : f(x) \le f(x_0)\}$ *is bounded.*
2. *In a neighborhood* N *of* S, *the function* f *is continuously differentiable and its gradient is Lipschitz-continuous, i. e., there exists a constant* $L > 0$

such that $\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|$, *for all* $x, y \in N$.

Under these assumptions for f, there exists a constant $\Gamma \ge 0$ such that $\|\nabla f(x)\| \le \Gamma$, for all $x \in S$.

It was proved in [11] that for any conjugate gradient method with strong Wolfe line search the following lemma holds.

Lemma 1 *Suppose that assumptions 1 and 2 hold and consider any conjugate gradient method (2) and (3), where d_k is a descent direction and α_k is obtained by the strong Wolfe line search. If*

$$\sum_{k\ge 1} \frac{1}{\|d_k\|^2} = \infty\,, \tag{15}$$

then

$$\liminf_{k\to\infty} \|g_k\| = 0\,. \tag{16}$$

For uniformly convex functions which satisfy the above assumptions we can prove that the norm of d_{k+1} generated by (8) is bounded above. Thus, by Lemma 1 we have the following result.

Theorem 3 *Suppose that assumptions 1 and 2 hold. Consider the algorithm (2) and (8), where d_{k+1} is a descent direction and α_k is obtained by the strong Wolfe line search.*

$$f(x_k + \alpha_k d_k) - f(x_k) \le \rho\alpha_k g_k^{\mathrm{T}} d_k\,, \tag{17}$$

$$\left|g_{k+1}^{\mathrm{T}} d_k\right| \le -\sigma g_k^{\mathrm{T}} d_k\,. \tag{18}$$

If for $k \ge 0$, $\|s_k\|$ tends to zero and there exist the nonnegative constants η_1 and η_2 such that

$$\|g_k\|^2 \ge \eta_1 \|s_k\|^2 \text{ and } \|g_{k+1}\|^2 \le \eta_2 \|s_k\|\,, \tag{19}$$

and the function f is a uniformly convex function, i. e., there exists a constant $\mu \ge 0$ such that for all $x, y \in S$

$$(\nabla f(x) - \nabla f(y))^{\mathrm{T}}(x - y) \ge \mu\|x - y\|^2\,, \tag{20}$$

then

$$\lim_{k\to\infty} g_k = 0\,. \tag{21}$$

Proof From (20) it follows that $y_k^{\mathrm{T}} s_k \ge \mu\|s_k\|^2$. Now, since $0 \le \theta_k \le 1$, from uniform convexity and (19) we have

$$\left|\beta_k^{\mathrm{N}}\right| \le \left|\frac{g_{k+1}^{\mathrm{T}} y_k}{g_k^{\mathrm{T}} g_k}\right| + \left|\frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k}\right| \le \frac{\|g_{k+1}\|\,\|y_k\|}{\eta_1\|s_k\|^2} + \frac{\eta_2\|s_k\|}{\mu\|s_k\|^2}\,.$$

But $\|y_k\| \le L\|s_k\|$; therefore,

$$\left|\beta_k^{\mathrm{N}}\right| \le \frac{\Gamma L}{\eta_1\|s_k\|} + \frac{\eta_2}{\mu\|s_k\|}\,.$$

Hence,

$$\|d_{k+1}\| \le \|g_{k+1}\| + \left|\beta_k^{\mathrm{N}}\right|\|s_k\| \le \Gamma + \frac{\Gamma L}{\eta_1} + \frac{\eta_2}{\mu}\,,$$

which implies that (15) is true. Therefore, by Lemma 1 we have (16), which for uniformly convex functions is equivalent to (21). □

Powell [25] showed that for general functions the PRP method is globally convergent if the step lengths $\|s_k\| = \|x_{k+1} - x_k\|$ tend to zero, i. e., $\|s_k\| \le \|s_{k-1}\|$ is a condition of convergence. For convergence of our algorithms from (19) we see that along the iterations, for $k \ge 1$, the gradient must be bounded as $\eta_1\|s_k\|^2 \le \|g_k\|^2 \le \eta_2\|s_{k-1}\|$. If the Powell condition is satisfied, i. e., $\|s_k\|$ tends to zero, then $\|s_k\|^2 \ll \|s_{k-1}\|$ and therefore the norm of the gradient can satisfy (19). In the numerical experiments we observed that (19) is always satisfied in the last part of the iterations.

For general nonlinear functions the convergence analysis of our algorithm exploits insights developed by Gilbert and Nocedal [17], Dai and Liao [9] and Hager and Zhang [18]. Global convergence proof of these new hybrid conjugate gradient algorithms is based on the Zoutendijk condition combined with the analysis showing that the sufficient descent condition holds and $\|d_k\|$ is bounded. Suppose that level set S is bounded and function f is bounded from below.

Lemma 2 *Assume that d_k is a descent direction and ∇f satisfies the Lipschitz condition $\|\nabla f(x) - \nabla f(x_k)\| \le L\|x - x_k\|$ for all x on the line segment connecting x_k and x_{k+1}, where L is a constant. If the line search satisfies the second Wolfe condition (5), then*

$$\alpha_k \ge \frac{1-\sigma}{L}\frac{\left|g_k^{\mathrm{T}} d_k\right|}{\|d_k\|^2}\,. \tag{22}$$

Proof Subtracting $g_k^{\mathrm{T}} d_k$ from both sides of (5) and using the Lipschitz condition, we have

$$(\sigma - 1) g_k^{\mathrm{T}} d_k \le (g_{k+1} - g_k)^{\mathrm{T}} d_k \le L \alpha_k \|d_k\|^2 . \quad (23)$$

Since d_k is a descent direction and $\sigma < 1$, (22) follows immediately from (23). □

Theorem 4 *Suppose that assumptions 1 and 2 hold, $0 < \theta_k \le 1, (g_{k+1}^{\mathrm{T}} y_k)(g_{k+1}^{\mathrm{T}} s_k) \le 0$, for every $k \ge 0$ there exists a positive constant ω, such that $1 - \theta_k (g_{k+1}^{\mathrm{T}} s_k)/(y_k^{\mathrm{T}} s_k) \ge \omega > 0$, and there exist the constants γ and Γ, such that for all $k, \gamma \le \|g_k\| \le \Gamma$. Then for the computational scheme (2) and (8), where α_k is determined by the Wolfe line search (4) and (5), either $g_k = 0$ for some k or*

$$\liminf_{k \to \infty} \|g_k\| = 0 . \quad (24)$$

Proof By the Wolfe condition (5) we have

$$y_k^{\mathrm{T}} s_k = (g_{k+1} - g_k)^{\mathrm{T}} s_k \ge (\sigma - 1) g_k^{\mathrm{T}} s_k = -(1 - \sigma) g_k^{\mathrm{T}} s_k . \quad (25)$$

By Theorem 2, and the assumption $1 - \theta_k (g_{k+1}^{\mathrm{T}} s_k)/(y_k^{\mathrm{T}} s_k) \ge \omega$, it follows that

$$g_k^{\mathrm{T}} d_k \le -\left(1 - \theta_{k-1} \frac{g_k^{\mathrm{T}} s_{k-1}}{y_{k-1}^{\mathrm{T}} s_{k-1}}\right) \|g_k\|^2 \le -\omega \|g_k\|^2 .$$

Therefore,

$$-g_k^{\mathrm{T}} d_k \ge \omega \|g_k\|^2 . \quad (26)$$

Combining (25) with (26), we get

$$y_k^{\mathrm{T}} s_k \ge (1 - \sigma) \omega \alpha_k \gamma^2 .$$

On the other hand $\|y_k\| = \|g_{k+1} - g_k\| \le L \|s_k\|$; hence,

$$\left|g_{k+1}^{\mathrm{T}} y_k\right| \le \|g_{k+1}\| \|y_k\| \le \Gamma L \|s_k\| .$$

With these, from (7) we get

$$\left|\beta_k^{\mathrm{N}}\right| \le \left|\frac{g_{k+1}^{\mathrm{T}} y_k}{g_k^{\mathrm{T}} g_k}\right| + \left|\frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k}\right| .$$

But

$$\left|\frac{g_{k+1}^{\mathrm{T}} y_k}{g_k^{\mathrm{T}} g_k}\right| \le \frac{\|g_{k+1}\| \|y_k\|}{\gamma^2} \le \frac{\Gamma L \|s_k\|}{\gamma^2} \le \frac{\Gamma L D}{\gamma^2} ,$$

where $D = \max\{\|y - z\| : y, z \in S\}$ is the diameter of the level set S.

On the other hand,

$$\left|\frac{g_{k+1}^{\mathrm{T}} g_{k+1}}{y_k^{\mathrm{T}} s_k}\right| \le \frac{\Gamma^2}{(1 - \sigma) \omega \alpha_k \gamma^2} .$$

Therefore,

$$\left|\beta_k^{\mathrm{N}}\right| \le \frac{\Gamma L D}{\gamma^2} + \frac{\Gamma^2}{(1 - \sigma) \omega \alpha_k \gamma^2} \equiv E . \quad (27)$$

Now, we can write

$$\|d_{k+1}\| \le \|g_{k+1}\| + \left|\beta_k^{\mathrm{N}}\right| \|s_k\| \le \Gamma + E D . \quad (28)$$

Since the level set S is bounded and the function f is bounded from below, using Lemma 2, from (4) it follows that

$$0 < \sum_{k=0}^{\infty} \frac{(g_k^{\mathrm{T}} d_k)^2}{\|d_k\|^2} < \infty , \quad (29)$$

i. e., the Zoutendijk condition holds. Therefore, from Theorem 2 using (29), the descent property yields

$$\sum_{k=0}^{\infty} \frac{\gamma^4}{\|d_k\|^2} \le \sum_{k=0}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} \le \sum_{k=0}^{\infty} \frac{1}{\omega^2} \frac{(g_k^{\mathrm{T}} d_k)^2}{\|d_k\|^2} < \infty ,$$

which contradicts (28). Hence, $\gamma = \liminf_{k \to \infty} \|g_k\| = 0$. □

Therefore, when $0 < \theta_k \le 1$ our hybrid conjugate gradient algorithms are globally convergent, meaning that either $g_k = 0$ for some k or (24) holds. Observe that in the conditions of Theorem 2 the direction d_{k+1} satisfies the sufficient descent condition independent of the line search.

Numerical Experiments

In this section we present the computational performance of a Fortran implementation of the CCOMB and NDOMB algorithms for a set of 750 unconstrained optimization test problems. The test problems are the unconstrained problems in the CUTE [6] library, along with other large-scale optimization problems presented in [1]. We selected 75 large-scale unconstrained optimization problems in extended or generalized form. Each problem was tested ten times for a gradually increasing number of variables: $n = 1000, 2000, \ldots,$

10000. At the same time we present comparisons with other conjugate gradient algorithms, including the performance profiles of Dolan and Moré [14].

All algorithms implement the Wolfe line search conditions with $\rho = 0.0001$ and $\sigma = 0.9$, and the same stopping criterion $\|g_k\|_\infty \leq 10^{-6}$, where $\|.\|_\infty$ is the maximum absolute component of a vector.

The comparisons of algorithms are given in the following context. Let f_i^{ALG1} and f_i^{ALG2} be the optimal values found by ALG1 and ALG2, for problem $i = 1, \ldots, 750$, respectively. We say that, in the particular problem i, the performance of ALG1 was better than the performance of ALG2 if

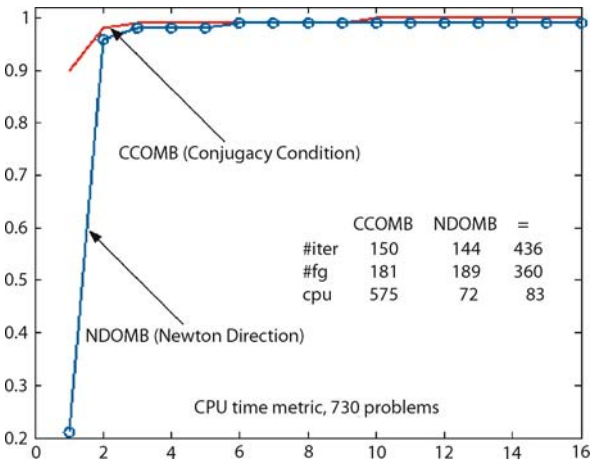
$$\left| f_i^{ALG1} - f_i^{ALG2} \right| < 10^{-3} \tag{30}$$

and the number of iterations, or the number of function-gradient evaluations, or the CPU time of ALG1 was less than the number of iterations, or the number of function-gradient evaluations, or the CPU time corresponding to ALG2, respectively.

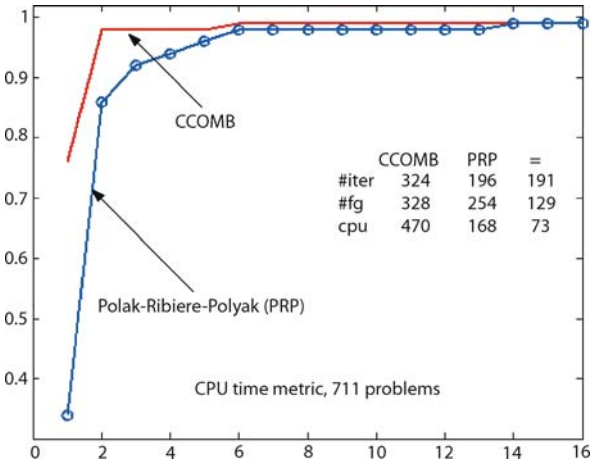
All codes were written in double-precision Fortran and compiled with f77 (default compiler settings) on an Intel Pentium 4, 1.8 GHz workstation. All these codes were authored by Andrei. The performances of these algorithms were evaluated using the profiles of Dolan and Moré [14]. That is, for each algorithm we plotted the fraction of problems for which the algorithm is within a factor of the best CPU time. The left side of these figures gives the percentage of the test problems, out of 750, for which an algorithm is more performant; the right side gives the percentage of the test problems that were successfully solved by each of the algorithms. Mainly, the right side represents a measure of an algorithm's robustness.

In the first set of numerical experiments, we compared the performance of the CCOMB algorithm with the performance of the NDOMB algorithm. Figure 1 shows the Dolan and Moré CPU performance profile of the CCOMB algorithm compared with that of the NDOMB algorithm.

Observe that the CCOMB algorithm outperforms the NDOMB algorithm in the vast majority of problems. Only 730 problems out of 750 satisfy criterion (30). Referring to the CPU time, the CCOMB algorithm was better in 575 problems, in contrast to the NDOMB algorithm, which solved only 72 problems in a better CPU time.



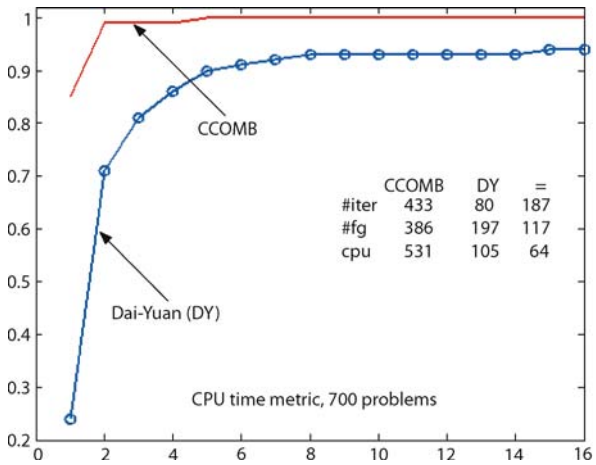
New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 1
Performance based on CPU time: CCOMB algorithm compared with the NDOMB algorithm



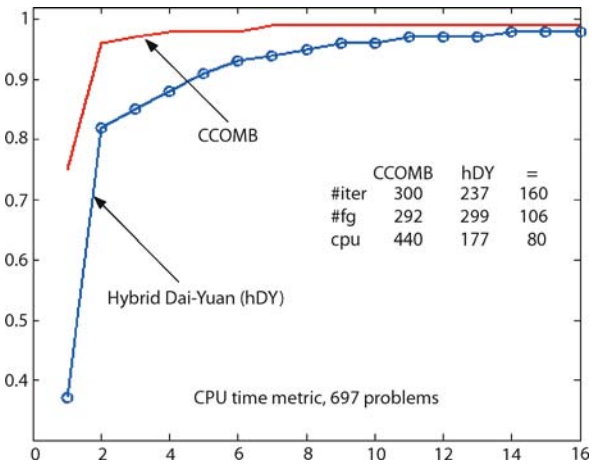
New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 2
Performance based on CPU time: CCOMB algorithm compared with the Polak–Ribière–Polyak (*PRP*) algorithm

In the second set of numerical experiments we compared the performance of the CCOMB algorithm with the performances of the PRP and DY conjugate gradient algorithms. Figures 2 and 3 show the Dolan and Moré CPU performance profiles of the CCOMB algorithm compared with those of the PRP and DY algorithms, respectively.

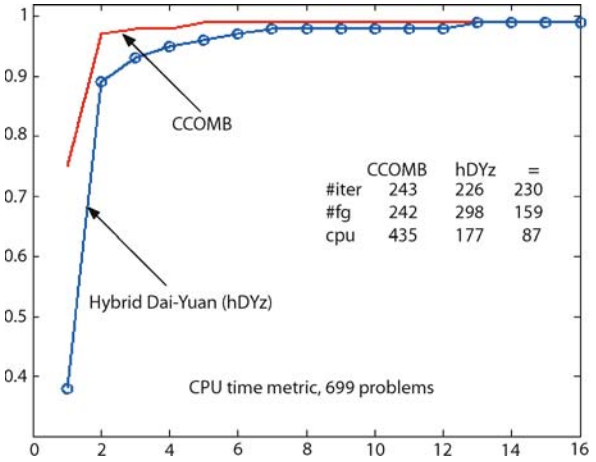
When comparing the CCOMB and PRP algorithms (Fig. 2), subject to the number of iterations, we see that the CCOMB algorithm was better in 324 problems (i. e.,



New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 3
Performance based on CPU time: CCOMB algorithm compared with the Dai–Yuan (*DY*) algorithm



New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 4
Performance based on CPU time: CCOMB algorithm compared with the hybrid Dai–Yuan (*hDY*) algorithm



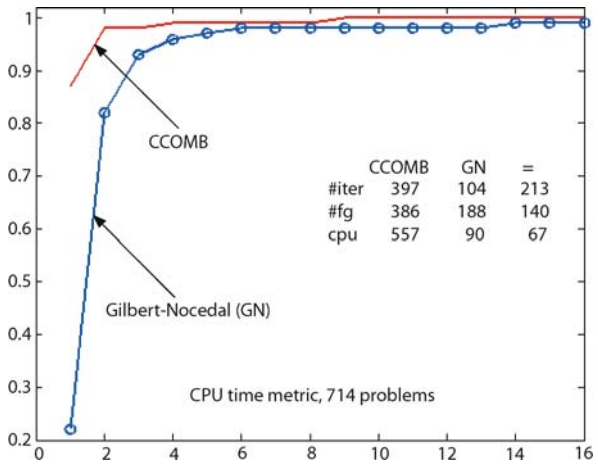
New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 5
Performance based on CPU time: CCOMB compared with the hybrid Dai–Yuan (*hDYz*) algorithm

it achieved the minimum number of iterations in 324 problems), the PRP algorithm was better in 196 problems and they achieved the same number of iterations in 191 problems, etc. Out of 750 problems, only for 711 problems does criterion (30) holds. Similarly, in Fig. 3 we see the number of problems for which the CCOMB algorithm was better than the DY algorithm. Observe that the convex combination of the PRP and DY algorithms, expressed as in (7), is far more successful than the PRP or the DY algorithm.

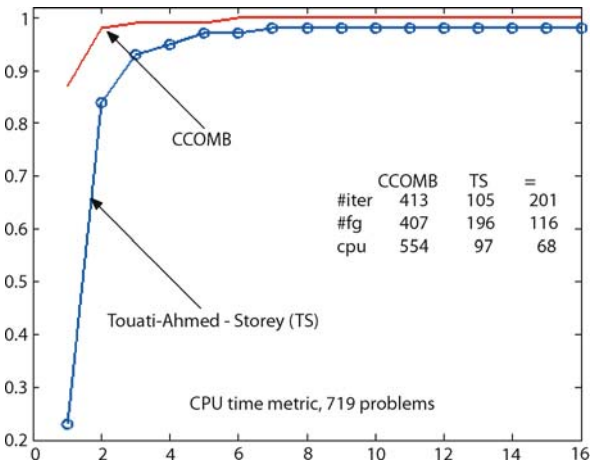
The third set of numerical experiments refers to the comparisons of the CCOMB algorithm with hybrid conjugate gradient algorithms: hDY, hDYz, GN, HuS, TS and LS-CD. Figures 4–9 presents the Dolan and Moré CPU performance profiles of these algorithms, as well as the number of problems solved by each algorithm in the minimum number of iterations, the minimum number of function evaluations and the minimum CPU time, respectively.

From these figures we see that the CCOMB algorithm is the top performer. Since these codes use the same Wolfe line search and the same stopping criterion they differ in their choice of the search direction. Hence, among these conjugate gradient algorithms we considered here, the CCOMB algorithm appears to generate the best search direction.

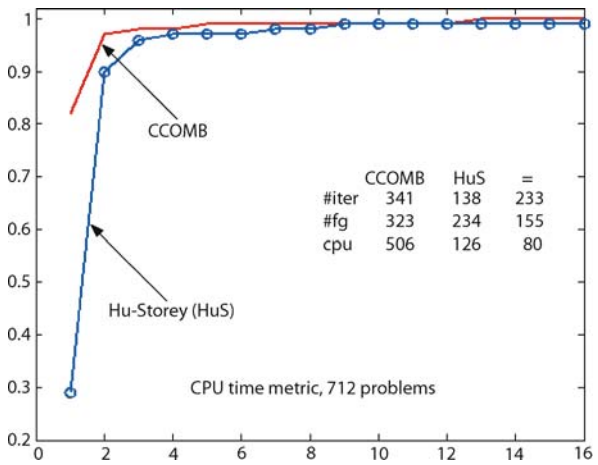
In the fourth set of numerical experiments we compared the CCOMB algorithm with the CG_DESCENT conjugate gradient algorithm of Hager and Zhang [18]. The computational scheme implemented in the CG_DESCENT algorithm is a modification of the HS method which satisfies the sufficient-descent condition, independent of the accuracy of the line search. The CG_DESCENT code, authored by Hager and Zhang, contains the variant CG_DESCENT (HZw) implementing the Wolfe line search and the variant CG_DESCENT (HZaw) implementing an approximate Wolfe line search. There are two main points associated



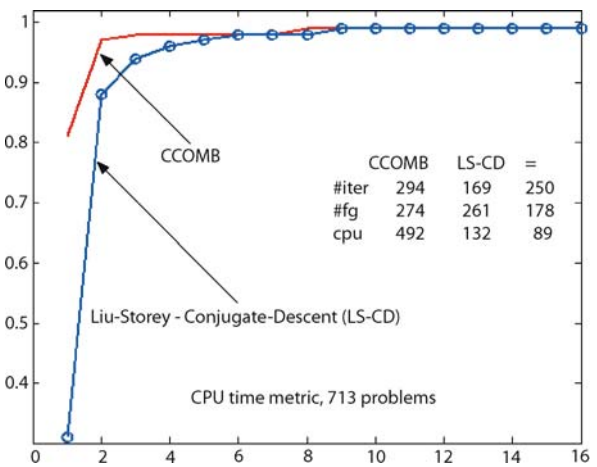
New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 6
Performance based on CPU time: CCOMB compared with the Gilbert–Nocedal (*GN*) algorithm



New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 8
Performance based on CPU time: CCOMB algorithm compared with the Touati-Ahmed–Storey (*TS*) algorithm



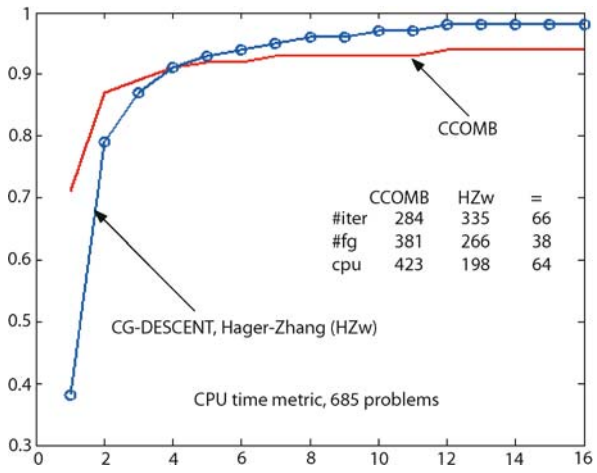
New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 7
Performance based on CPU time: CCOMB algorithm compared with the Hu–Storey (*HuS*) algorithm



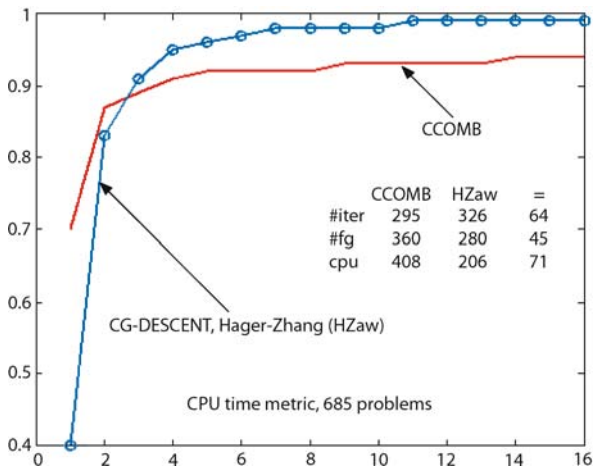
New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 9
Performance based on CPU time: CCOMB algorithm compared with the Liu–Storey–conjugate descent (*LS-CD*) algorithm

with the CG_DESCENT algorithm. Firstly, the scalar products are implemented using the loop unrolling of depth 5. This is efficient for large-scale problems (over 10^6 variables). Secondly, the Wolfe line search is implemented using a very fine numerical interpretation of the first Wolfe condition (4). The Wolfe conditions implemented in the CCOMB and CG_DESCENT (HZw) algorithms can compute a solution with accuracy of the order of the square root of the machine epsilon.

In contrast, the approximate Wolfe line search implemented in the CG_DESCENT (HZaw) algorithm can compute the solution with accuracy of the order of machine epsilon. Figures 10 and 11 present the performance profile of these algorithms in comparison with that of the CCOMB algorithm. We see that the CG_DESCENT algorithm is more robust than the CCOMB algorithm.



New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 10
Performance based on CPU time: CCOMB algorithm compared with the CG_DESCENT algorithm with Wolfe line search (*HZw*)



New Hybrid Conjugate Gradient Algorithms for Unconstrained Optimization, Figure 11
Performance based on CPU time: CCOMB algorithm compared with the CG_DESCENT algorithm with approximate Wolfe line search (*HZaw*)

Conclusion

We know a large variety of conjugate gradient algorithms. The known hybrid conjugate gradient algorithms are based on projection of classical conjugate gradient algorithms. In this chapter we have proposed new hybrid conjugate gradient algorithms in which the famous parameter β_k is computed as a convex combination of β_k^{PRP} and β_k^{DY}, i. e., $\beta_k = (1-\theta_k)\beta_k^{\mathrm{PRP}} + \theta_k\beta_k^{\mathrm{DY}}$. The parameter θ_k is computed in such a manner that the conjugacy condition is satisfied, or the corresponding direction in the hybrid conjugate gradient algorithm is the Newton direction. For uniformly convex functions if the step size s_k approaches zero, the gradient is bounded in the sense that $\eta_1 \|s_k\|^2 \leq \|g_k\|^2 \leq \eta_2 \|s_{k-1}\|$ and the line search satisfies the strong Wolfe conditions, then our hybrid conjugate gradient algorithms are globally convergent. For general nonlinear functions if the parameter θ_k from the β_k^{N} definition is bounded, i. e., $0 < \theta_k < 1$, then our hybrid conjugate gradient is globally convergent. The Dolan and Moré CPU performance profile of the hybrid conjugate gradient algorithm based on the conjugacy condition (CCOMB algorithm) is better than the performance profile corresponding to the hybrid algorithm based on the Newton direction (NDOMB algorithm). The performance profile of the CCOMB algorithm was better than those of the well-established conjugate gradient algorithms (hDY, hDYz, GN, HuS, TS and LS-CD) for a set consisting of 750 unconstrained optimization test problems, some of them from CUTE library. Additionally the proposed hybrid conjugate gradient algorithm CCOMB is more robust than the PRP and DY conjugate gradient algorithms. However, subject to robustness the CCOMB algorithm is outperformed by the CG_DESCENT algorithm.

References

1. Andrei N (2004) Test functions for unconstrained optimization. Available via http://www.ici.ro/camo/neculai/SCALCG/evalfg.for
2. Andrei N (2007) Scaled conjugate gradient algorithms for unconstrained optimization. Comput Optim Appl 38:401–416
3. Andrei N (2007) Scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization. Optim Method Softw 22:561–571
4. Andrei N (2007) A scaled BFGS preconditioned conjugate gradient algorithm for unconstrained optimization. Appl Math Lett 20:645–650
5. Birgin E, Martínez JM (2001) A spectral conjugate gradient method for unconstrained optimization. Appl Math Optim 43:117–128
6. Bongartz I, Conn AR, Gould NIM, Toint PL (1995) CUTE: constrained and unconstrained testing environments. ACM Trans Math Softw 21:123–160

7. Dai YH (1997) Analysis of conjugate gradient methods. Ph.D. Thesis, Institute of Computational Mathematics and Scientific/Engineering Computing. Chinese Academy of Science, Beijing
8. Dai YH (2001) New properties of a nonlinear conjugate gradient method. Numer Math 89:83–98
9. Dai YH, Liao LZ (2001) New conjugacy conditions and related nonlinear conjugate gradient methods. Appl Math Optim 43:87–101
10. Dai YH, Liao LZ, Li D (2004) On restart procedures for the conjugate gradient method. Numer Algorithm 35:249–260
11. Dai YH, Han JY, Liu GH, Sun DF, Yin X, Yuan Y (1999) Convergence properties of nonlinear conjugate gradient methods. SIAM J Optim 10:348–358
12. Dai YH, Yuan Y (1999) A nonlinear conjugate gradient method with a strong global convergence property. SIAM J Optim 10:177–182
13. Dai YH, Yuan Y (2001) An efficient hybrid conjugate gradient method for unconstrained optimization. Annu Oper Res 103:33–47
14. Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. Math Program 91:201–213
15. Fletcher R (1987) Practical Methods of Optimization, vol 1: Unconstrained Optimization. Wiley, New York
16. Fletcher R, Reeves C (1964) Function minimization by conjugate gradients. Comput J 7:149–154
17. Gilbert JC, Nocedal J (1992) Global convergence properties of conjugate gradient methods for optimization. SIAM J Optim 2:21–42
18. Hager WW, Zhang H (2005) A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM J Optim 16:170–192
19. Hager WW, Zhang H (2006) A survey of nonlinear conjugate gradient methods. Pac J Optim 2:35–58
20. Hestenes MR, Stiefel EL (1952) Methods of conjugate gradients for solving linear systems. J Res Nat Bur Stand 49:409–436
21. Hu YF, Storey C (1991) Global convergence result for conjugate gradient methods. J Optim Theory Appl 71:399–405
22. Liu Y, Storey C (1991) Efficient generalized conjugate gradient algorithms, Part 1: Theory. JOTA 69:129–137
23. Polak E, Ribière G (1969) Note sur la convergence de directions conjuguée. Rev Fr Inf Rech Oper, 3e Année 16:35–43
24. Polyak BT (1969) The conjugate gradient method in extreme problems. USSR Comp Math Math Phys 9:94–112
25. Powell MJD (1977) Restart procedures of the conjugate gradient method. Math Program 2:241–254
26. Powell MJD (1984) Nonconvex minimization calculations and the conjugate gradient method. In: Watson GA (ed) Numerical Analysis Proceedings, Dundee, 1983. Lecture Notes in Mathematics, vol 1066. Springer, Berlin, pp 122–141
27. Shanno DF, Phua KH (1976) Algorithm 500, Minimization of unconstrained multivariate functions. ACM Trans Math Softw 2:87–94
28. Touati-Ahmed D, Storey C (1990) Efficient hybrid conjugate gradient techniques. J Optim Theory Appl 64:379–397
29. Yuan Y (1993) Analysis on the conjugate gradient method. Optim Method Softw 2:19–29

Nonconvex Energy Functions: Hemivariational Inequalities

Georgios E. Stavroulakis[1,2],
Euripidis Mistakidis[3]
[1] Carolo Wilhelmina Technische Universität, Braunschweig, Germany
[2] Technical University of Crete, Chania, Greece
[3] University Thessaly, Volos, Greece

MSC2000: 49J40, 70-XX, 80-XX, 49J52, 49Q10, 74K99, 74Pxx

Article Outline

Keywords

Hemivariational inequalities; Nonsmooth modeling; Generalized subdifferential of F.H. Clarke; Nonconvex energy function

A large number of problems in mechanics, engineering and economics can be defined by means of appropriate differentiation of an *energy function*. This function is sometimes called a *potential*. On the assumption of

differentiability one may write the first order optimality condition that the gradient of the function is equal to zero at some point. All points which satisfy this condition are called *critical points*. The set of nonlinear equations derived this way are the governing equations (the model) of the studied problem. Equivalently, one may consider the requirement that the directional derivative of the function at this critical point and in all directions emanating from this point is equal to zero, or that all possible small variations of the energy function around the critical point is equal to zero. This way the critical point is described by a *variational inequality* and one speaks about a variational formulation of the problem. Whether a critical point is also a minimum of the considered energy function, or, for example, a maximum or a saddle point requires the consideration of second order optimality conditions. One should mention in passing that (possibly local) minima are of outmost importance in applications. For instance, in mechanics they provide stable equilibria of the studied mechanical systems. Convexity and coercivity of the energy function guarantees that a critical point is a minimum while strict convexity ensures its uniqueness as well.

Lack of differentiability of the energy function, or the consideration of inequality constraints in the minimization problem changes the picture. As far as the convexity property holds, one may use the powerful tools of *convex analysis* to study the problem. The gradient of the nonsmooth but convex function is replaced by the subgradient, the differential by a set-valued *subdifferential* (in the sense of J.-J. Moreau and R.T. Rockafellar) and the critical point equation by the differential inclusion: zero must be an element of the subdifferential of the energy function at minimum. Accordingly, in the variational formulation a variational equation is replaced by a variational inequality. Analogous considerations hold true for the case of inequality constraints. Here, one has, in principle, two ways to study the problem. Either all admissible variations are taken into account in the derivation of the optimality conditions, or the inequality constraints are included by means of the indicator function of the admissible set in the set of the problem's variables. Following Moreau, who introduced and studied convex analysis and applied it for the solution of mechanical problems, a convex nondifferentiable potential energy function is called a *superpotential*.

Admittedly, convexity is a convenient assumption too good to be true in real life applications. The study of nonconvex energy functions requires new tools and methods, which are being developed within the area of nonconvex analysis. Among them, the notion of the generalized gradients in the sense of F.H. Clarke, Rockafellar [2] has found several applications. Concerning the search for minima of a nonconvex energy function one may formulate critical point problems. Under certain assumptions the generalized gradient of Clarke provides a useful tool for the formulation and the study of nonconvex and nonsmooth energy function problems. In the area of mechanics, P.D. Panagiotopoulos has been the first to introduce and use this notion. He called the resulting nonconvex variational inequalities *hemivariational inequalities*. Initially, the notion of Clarke, which is suitable for locally Lipschitz energy functions, has been used. Later, the extended notion of Rockafellar, which roughly speaking includes infinite vertical branches, has been considered. Later, he extended this analysis by incorporating inequality constraints or convex superpotential terms (as in the case of convex problems), which have been inspired from the engineering applications he studied. Thus, he formulated and studied *variational-hemivariational inequality* problems. On the analogy of convex analysis, the nonconvex and possibly nondifferentiable energy functions have been called by Panagiotopoulos *nonconvex superpotentials*. Once more, it should be emphasized here that hemivariational inequalities are not equivalent to minimum problems but to substationarity problems. Nevertheless, they constitute a consistent extension of variational inequalities, and they include them for the case of convex energy functions. Furthermore, and this is important for some numerical algorithms, as with the subdifferential of convex analysis the generalized subdifferential of Clarke involves one convex set in the set-valued approximation of the differential of a nonconvex and nonsmooth energy function. Consequently, the development of theory and algorithms runs in parallel with the ones developed for problems of convex analysis and of variational inequalities. It should be mentioned that propositions of substationary potential and complementary energy generalize the classical minimum energy propositions of mechanics (where, for historical reasons, they are known as principles). Moreover, following the example of *nonsmooth analy-*

sis, Panagiotopoulos named the whole area of mechanics which deals with nondifferentiable functions *nonsmooth mechanics*.

In this article the notions of generalized derivative of Clarke and Rockafellar and the definition of the substationarity points are first given. After that, the hemivariational and the variational-hemivariational inequalities of Panagiotopoulos are presented. Short comments on the theoretical tools which are used for their study, examples of applications and numerical algorithms which have been proposed for their solution complete the paper. Details on all previous issues can be found in the cited publications. See also ► hemivariational inequalities: applications in mechanics.

Clarke's Generalized Derivative

Let a function f be locally Lipschitz at $x \in X$ and let y be a vector in X. The *directional differential in the sense of Clarke* of f at x in the direction y, denoted by $f^0(x, y)$, is defined by the relation:

$$f^0(x, y) = \limsup_{\substack{\mu \to 0+ \\ h \to 0}} \frac{f(x + h + \mu y) - f(x + h)}{\mu}.$$

$f^0(x, y)$ is also called a *generalized directional differential*.

By means of the directional differential $f^0(x, y)$ one can now define the *generalized gradient* $\overline{\partial} f(x)$: $X \to X^*$:

$$\overline{\partial} f(x) = \left\{ x^* \in X^* : \begin{array}{c} f^0(x, x_1 - x) \geq (x^*, x_1 - x) \\ \forall x_1 \in X \end{array} \right\}. \tag{1}$$

One may also use the definition

$$\overline{\partial} f(x) = \left\{ x^* \in X^* : \ (x^*, -1) \in N_{\operatorname{epi} f}(x, f(x)) \right\}, \tag{2}$$

where $N_C(x)$ denotes the normal cone to a set C at point x and epi f is the epigraph of the function f.

Note that $\overline{\partial} f(\cdot)$ is a multivalued mapping; it is a nonempty convex, closed and bounded subset of $X^\star$ and the following relation holds true:

$$f^0(x, y) = \max \left\{ \langle y, x^* \rangle : \ x^* \in \overline{\partial} f(x) \right\}. \tag{3}$$

For didactical reasons the notation $\overline{\partial}$ is used here (and in most of the work of Panagiotopoulos). In honor of Clarke who proposed it, the notation ∂_{CL} is sometimes used. When misunderstanding is not expected, the notation ∂, which is usually reserved for the convex analysis subdifferential, is also used. It should also be noted here that $\overline{\partial}$ should not be confused with the superdifferential used in the theory of quasidifferentiability in the sense of V.F. Demyanov.

Relation (1) can be used to define the generalized gradient $\overline{\partial} f(x)$ for any type of function $f: X \to \overline{\mathbb{R}}$ which is finite at the point x. Note that $\overline{\partial} f(x)$ may be empty. The above definition of $\overline{\partial} f(x)$ for any function $f: X \to \overline{\mathbb{R}}$ makes sense, because the normal cone $N_C(x)$ can be defined with respect to any set epi f. The generalized directional differential $f^\uparrow(x; y)$ at x in the direction y is defined by the relation:

$$f^\uparrow(x, y) = \sup \left\{ \langle y, x^* \rangle : \ x^* \in \overline{\partial} f(x) \right\}. \tag{4}$$

Thus one may write the relation:

$$\overline{\partial} f(x) = \left\{ x^* \in X^* : \begin{array}{c} f^\uparrow(x, x_1 - x) \geq \langle x^*, x_1 - x \rangle \\ \forall x_1 \in X \end{array} \right\}. \tag{5}$$

The directional differential $f^\uparrow(x; y)$ is also called *directional differential in the sense of Rockafellar* [15]. Note that $\overline{\partial} f(x) = \emptyset$ if $f^\uparrow(x, 0) = -\infty$, while if $f^\uparrow(x, y)$ is finite for every y, then $\overline{\partial} f(x) \neq \emptyset$.

It should be noted that for a convex function f one has

$$f^\uparrow(x, y) = \liminf_{\widetilde{y} \to y} f'(x, \widetilde{y}), \quad \forall y \in X, \tag{6}$$

where $f'(\cdot, \cdot)$ denotes the one-sided directional Gâteaux differential. Moreover, for a locally Lipschitz function f at point x one has

$$f^\uparrow(x, y) = f^0(x, y), \quad \forall y \in X, \tag{7}$$

and for a continuously differentiable f:

$$\overline{\partial} f(x) = \{\operatorname{grad} f(x)\}. \tag{8}$$

Examples

For a convex function f one has

$$\overline{\partial} f(x) = \partial f(x), \tag{9}$$

and for a concave and bounded below on a neighborhood of x function f:

$$\overline{\partial} f(x) = -\partial(-f)(x) \tag{10}$$

at every point x where f is finite. The *indicator function* I_C of a (generally nonconvex) set C is defined by $I_C(x) = 0$ if $x \in C$, and $I_C(x) = \infty$ otherwise. It can be proved that

$$\overline{\partial} I_C(x) = N_C(x) \tag{11}$$

and

$$I_C^{\uparrow}(x, y) = I_{T_{C(x)}}(y). \tag{12}$$

In the finite-dimensional case $X \equiv \mathbf{R}^n$, for a locally Lipschitz function f at a point x, $\overline{\partial} f(x)$ is the convex hull of all points $y \in \mathbf{R}^n$ of the form

$$y = \lim_{i \to \infty} \operatorname{grad} f(x_i), \tag{13}$$

where x_i converges as $i \to \infty$ to x, avoiding the nondifferentiability points and any other points of a set of measure zero (in the sense of Lebesgue) and such that $\operatorname{grad} f(x_i)$ converges.

For a *maximum-type function* f, i. e., when the function is defined by means of continuously differentiable functions $\varphi_i = \varphi_i(x)$, $i = 1, \ldots, m$, $x \in \mathbf{R}^n$ by the relation: $f = \max\{\varphi_i, \ldots, \varphi_m\}$, one has

$$\overline{\partial} f(x) = \operatorname{co}\{\operatorname{grad} \varphi_i(x) : \ i \in I(x)\}, \tag{14}$$

where $I(x) = \{\mathrm{i}: \varphi_i(x) = f(x)\}$ is the active index set.

The *normal cone* to a set defined by: $C = \{x \in \mathbf{R}^n : f(x) \leq 0\}$ at a point x_0 with $f(x_0) = 0$ is described by the relation

$$N_C(x_0) \subset \left\{\lambda x^* : \ x^* \in \mathbb{R}^n, \ \lambda \geq 0, \ x^* \in \overline{\partial} f(x_0)\right\},$$

whenever f is Lipschitzian on a neighborhood of x_0 and $0 \notin \overline{\partial} f(x_0)$. The notion of $\overline{\partial}$-regularity assures that directional derivative information can be regained from the Clarke's notion. For a locally Lipschitz function one requires that $f^0(x, y) = f'(x, y)$, $\forall y \in X$, holds at x. This definition is equivalent to the statement that epi f is regular at $(x, f(x))$. For instance, a convex function and a maximum type function are $\overline{\partial}$-regular at a point x where they are finite. For example, for the max-type function $f = \max\{\varphi_1, \ldots, \varphi_m\}$ one has

$$N_C(x_0) = \overline{\partial} I_C(x_0)$$
$$= \left\{ z = \sum_{i=1}^{m} \lambda_i \operatorname{grad} \varphi_i(x_0) : \begin{array}{c} \lambda_i \geq 0, \\ \varphi_i(x_0) \leq 0, \\ \lambda_i \varphi_i(x_0) = 0 \end{array} \right\}, \tag{15}$$

if $0 \notin \overline{\partial} f(x_0)$. The above relation permits the extension of the Lagrange multiplier rule for optimization problems subjected to the nonconvex inequality constraints $\varphi_i(x) \leq 0$, $i = 1, \ldots, m$. This becomes obvious, e. g. if one considers the search for a local minimum problem of a continuously differentiable function g: $\mathbf{R}^n \to \mathbf{R}$ over $C = \{x \in \mathbf{R}^n : \varphi_i(x) \leq 0, \ i = 1, \ldots, m\}$. A necessary condition is $0 \in \overline{\partial}(g + I_C)(x)$, which implies that

$$-\operatorname{grad} g(x) \in \overline{\partial} I_C(x), \tag{16}$$

which together with (15) leads to the Lagrange multiplier rule.

Critical Points and Substationarity

The notion of *substationarity* plays an important role in the theory of hemivariational inequalities because it permits the formulation of the propositions of substationary potential and complementary energy which generalize the corresponding classical minimum energy propositions in mechanics [12,13,14]. Point x_0 is a *substationarity point of a functional* $f : X \to \overline{\mathbb{R}}$ if

$$0 \in \overline{\partial} f(x_0). \tag{17}$$

Equivalently one has:

$$f^{\uparrow}(x_0, y) \geq 0, \quad \forall y \in X. \tag{18}$$

Substationarity points are all the classical stationarity points, all the local minima, a large class of local maxima, as well as all the saddle points. Point x is said to be a *substationarity point of f with respect to a set C*, if $f + I_C$ is substationary at x.

Nonconvex Energy Functions

The nonconvex superpotentials resulting by integrating discontinuous functions $\beta \in L^{\infty}_{\mathrm{loc}}(\mathbf{R})$ play an important

role in the formulation of hemivariational inequalities for several types of mechanical problems. After some technical details about filling in gaps in a multifunction, nonmonotone laws in mechanics which admit a nonconvex energy function will be introduced.

Filling the Gaps in a Multifunction

Suppose that β: $\mathbf{R} \to \mathbf{R}$ is a function such that $\beta \in L^{\infty}_{\text{loc}}(\mathbf{R})$, i. e. a function which is essentially bounded on any bounded interval of $\mathbf{R}$. For any $\rho > 0$ and $\xi \in \mathbf{R}$ let us define

$$\overline{\beta}_{\rho}(\xi) = \operatorname*{essinf}_{\|\xi_1 - \xi\| \le \rho} \beta(\xi_1) \tag{19}$$

and

$$\overline{\overline{\beta}}_{\rho}(\xi) = \operatorname*{esssup}_{\|\xi_1 - \xi\| \le \rho} \beta(\xi_1). \tag{20}$$

Obviously, the monotonicity properties of $\rho \to \overline{\beta}_{\rho}(\xi)$ and $\rho \to \overline{\overline{\beta}}_{\rho}(\xi)$ imply that the limits as $\rho \to 0_+$ exist. Therefore one may write that: $\overline{\overline{\beta}}(\xi) = \lim_{\rho \to 0_+} \overline{\overline{\beta}}_{\rho}(\xi)$. Furthermore, one defines the multivalued function:

$$\widetilde{\beta}(\xi) = \left[\overline{\beta}(\xi), \overline{\overline{\beta}}(\xi)\right], \tag{21}$$

where $[\cdot, \cdot]$ denotes an interval between the two given arguments. Then, a locally Lipschitz function j can be determined, up to an additive constant, by the relation

$$j(\xi) = \int_0^{\xi} \beta(\xi_1)\, d\xi_1, \tag{22}$$

such that $\overline{\partial} j(\xi) \subset \widetilde{\beta}(\xi)$. If moreover $\beta(\xi_{\pm})$ exist for each $\xi \in \mathbf{R}$, then one has $\overline{\partial} j(\xi) = \widetilde{\beta}(\xi)$.

Superpotential Nonmonotone Laws

Let us assume that one has an one-dimensional mechanical law which is described by the graph of a discontinuous function. For instance, a force-displacement law $(S - u)$ is considered, which may correspond to an one-dimensional nonlinear spring law, to a nonlinear boundary condition, etc. The law is considered to be of the form β: $u \to -S$ where $u \in \mathbf{R}$ and $S \in \mathbf{R}$. The procedure of (19)–(22) is used in order to define a locally Lipschitz nonconvex superpotential energy function $j(u)$. The mechanical law is produced, in turn, by using the previously introduced generalized subdifferential operator $\partial_{CL} = \overline{\partial}$ and the nonconvex superpotential j as follows:

$$-f \in \partial_{\text{CL}} j(u). \tag{23}$$

By definition, (23) is equivalent to the inequality

$$j^{\uparrow}(u, u^* - u) \ge \langle -f, u^* - u \rangle, \quad \forall u^* \in U, \tag{24}$$

for $u \in U$, which Panagiotopoulos has called a hemivariational inequality, and to the inclusion

$$(-f, -1) \in N_{\text{epi}\, j}(u, j(u)). \tag{25}$$

For j Lipschitzian, $j^{\uparrow}$ in (23) is replaced by j^0 Obviously, if j is a convex superpotential, one has superpotential laws which can be described by monotone graphs with complete vertical branches. The procedure would be identical in that case as well, where ∂_{CL} is replaced by the subdifferential of convex analysis. The result would be a variational inequality. Note also that extensions to multidimensional laws (e. g., material constitutive relations) can be considered within this formulation.

Hemivariational Inequalities

An abstract coercive hemivariational inequality is written first. Let V be a real Hilbert space with the property that $V \subset [L^2(\Omega)]^n \subset V^{\star}$, where $V^{\star}$ denotes the dual space of V, and the injections are continuous and dense. Let moreover a boundary value problem be defined in an open, bounded subset Ω of $\mathbf{R}^n$. Here $(\cdot, \cdot)$ denotes the $[L^2(\Omega)]^n$ inner product and the duality pairing, $\|\cdot\|$ is the norm of V and $|\cdot|_2$ is the $[L^2(\Omega)]^n$-norm. One should recall that the form $(\cdot, \cdot)$ extends uniquely from $V \times L^2\ [(\Omega)]^n$ to $V \times V^{\star}$. Moreover let L: $V \to L^2(\Omega)$, $Lu = \widehat{u}$, $\widehat{u}(x) \in \mathbb{R}$ be a linear continuous mapping. Further, assume that $l \in V^{\star}$, that L: $V \to L^2(\Omega)$ and that $\widetilde{V} = \left\{v \in V : \widehat{v} \in L^{\infty}(\Omega)\right\}$ is dense in V for the V-norm, and has a Galerkin base in V. It is also assumed that $a(\cdot, \cdot)$: $V \times V \to \mathbf{R}$ is a bilinear symmetric continuous form which is *coercive*, i. e. there exists a constant c

> 0 such that

$$a(v,v) \geq c\,\|v\|^2, \quad \forall v \in V. \tag{26}$$

A *coercive hemivariational inequality problem* reads: Find $u \in V$ such that

$$a(u, v-u) + \int_\Omega j^0(\widehat{u}, \widehat{v}-\widehat{u})\, d\Omega \geq (l, v-u), \quad \forall v \in V. \tag{27}$$

For example, a linear elastostatic structural analysis problem with additional nonlinear elements of nonmonotone type which admit a nonconvex superpotential $j(u)$ can be written in the hemivariational inequality form (25). In this context u are the displacements at the various points of the structure, which occupies Ω in its undeformed configuration. For a plate problem one has $\Omega \subset \mathbf{R}^2$, for a three-dimensional continuum $\Omega \subset \mathbf{R}^3$, etc. Moreover the functional space V is dictated from the kind of the assumed application and from the (natural, support) boundary conditions of the structure. The operator L and the energy form (u, u) depend on the mechanical theory used for the elastic part of the structure, while l denotes the external loading. Finally, coercivity usually means that a well-defined mechanical theory has been assumed and sufficient boundary conditions are assigned so that, for example, no rigid body motions of the structure are allowed. Finally, one should note that the familiar form of the principle of virtual work (i. e., a variational equality) can be obtained back from the hemivariational inequality (27) if the effect of the nonlinear terms is neglected, i. e., if the second term on the left-hand side of (27) is absent and if an equality is assumed in the place of the inequality.

Theoretical Studies

One recalls that the theory of the existence of solution of variational inequalities is a well-developed theory in mathematics which is closely connected with the convexity of the energy functionals involved. Indeed the existence theory of variational inequalities is based on monotonicity arguments. On the contrary the study of hemivariational inequalities, due to the absence of convexity is based on compactness arguments. Existence and approximation results for hemivariational inequalities have been studied for several applications. See [11,14] for details and citations to related references.

Semicoercive Case

Noncoercive problems arise in mechanics, for example, when the existing classical boundary conditions are not sufficient to prevent (even infinitesimal) rigid body displacements and rotations of the structure. In classical, equality mechanics, one may write conditions that guarantee the existence of a solution to the considered boundary value problem. Nevertheless, uniqueness of some quantities may be lost. One may consider, as an example, a free elastic body subjected to self-equilibrated external forces. In this case stresses and deformations of the elastic body can be determined, but its displacements are only determined modulo some rigid body displacements and rotations. In the presence of inequality (e. g., unilateral contact) constraints analogous relations have also been provided by G. Fichera (see, e. g., [13, Chap. 4]). It is interesting to observe that some inequality constraints may be activated and stabilize the body, a result that has certain applications in robotics [1].

For hemivariational inequalities, analogous results have been obtained by Panagiotopoulos and coworkers (see also [4]). One such result for an abstract hemivariational inequality is given here, without proof. Here $a(\cdot, \cdot)$ is assumed to be *semicoercive*, i. e., $a(\cdot, \cdot)$ is continuous and symmetric but it has a nonzero kernel, i. e. $\ker a(\cdot, \cdot) = \{q\colon a(q, q) = 0\} \neq \{0\}$. Moreover let $\ker a$ be finite dimensional. Let us assume that the norm $\|v\|$ on V is equivalent to $\|\|v\|\| = p(\widetilde{v}) + \|q\|_2$, where $v = \widetilde{v} + q$, $q \in \ker a$, $\widetilde{v} \in \ker a^\perp$ (i. e. $(\widetilde{v}, q) = 0$, $\forall\, q \in \ker a$) and $p(\widetilde{v})$ is a seminorm on V such that $p(v) = p(v + q)$, $\forall\, v \in V$, $q \in \ker a$, and let $a(v, v) \geq c(p(v))^2$, $\forall v \in V$, $c = \text{const} > 0$. This semicoercivity inequality replaces (26) for the coercive case. Moreover, let q_+ and q_- be the positive and negative parts of $\widehat{q}$, where $\widehat{q} = Lq$, i. e. $q_+ = \max\{0, \widehat{q}\}$, $q_- = \max\{0, -\widehat{q}\}$. The following quantities will also be used: $\beta(-\infty) = \lim \sup_{\xi \to -\infty} \beta(\xi)$ and $\beta(\infty) = \lim \inf_{\xi \to \infty} \beta(\xi)$. On the assumption that

$$\beta(-\infty) \leq \beta(\xi) \leq \beta(\infty), \quad \forall \xi \in \mathbb{R}, \tag{28}$$

a necessary condition for the existence of a solution $u \in V$ of a semicoercive hemivariational inequality problem

is the following inequality:

$$\int_\Omega [\beta(-\infty)q_+ - \beta(\infty)q_-]\, d\Omega \le (l, q) \le \int_\Omega [\beta(\infty)q_+ - \beta(-\infty)q_-]\, d\Omega, \quad \forall q \in \ker a. \tag{29}$$

If (28) holds strictly (with $<$ instead of $\le$), then (29) also holds strictly.

Substationarity of the Energy

Equivalent to the hemivariational inequality is the following *substationarity problem*: Find $u \in V$ such that the superpotential energy functional

$$\Pi(v) = \frac{1}{2}a(v, v) + \int_\Omega j(\widehat{v})\, d\Omega - (l, v) \tag{30}$$

is substationary at $v = u$. Here, the integral $\int_\Omega j(\widehat{v})\, d\Omega$ is set equal to ∞ if it is not defined.

Recall that the latter problem is, by definition, equivalent to: Find $u \in V$ which is a solution of the inclusion

$$0 \in \partial\Pi(u). \tag{31}$$

The equivalence of the above defined substationarity problem with the hemivariational inequality can be proved, on the assumption that j is locally Lipschitz and ∂-regular

Applications

Several types of hemivariational inequalities have already been studied (see, for example, [13,14], and the references given there) with respect to certain engineering problems, e. g. in nonmonotone semipermeability problems, in the theory of multilayered plates (delamination), in the theory of composite structures, in the theory of partial debonding of adhesive joints etc.

Numerical Algorithms

Till now (1998), the following methods have been investigated for the numerical solution of hemivariational inequality problems in mechanics.

- Nonsmooth optimization algorithms (in particular, of the *bundle algorithm* optimization type), for the solution of the inclusion (31). See [6,9] and, for the bundle nonsmooth optimization concept, e. g., [7].
- Decomposition into a series of convex subproblems (for instance, into a number of variational inequalities). For the numerical treatment of problems in elastostatics this approach has been the most fruitful, according to the experience of the authors. The decomposition has been based either on engineering motivated methods and heuristics (cf., [8]), or on mathematical programming techniques. In the last case results from the theory of quasidifferentiability [3], of difference-convex optimization [10] and of enumeration-type or branch and bound procedures [16] have been tested. More details can be found in [3,5,10,14].

See also

- ▶ Bariational Inequalities
- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex-Nonsmooth Calculus of Variations
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities

- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Al-Fahed AM, Stavroulakis GE, Panagiotopoulos PD (1992) A linear complementarity approach to the frictionless gripper problem. Internat J Robotics Res 11(2):112–122
2. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, Hoboken. reprinted as Classics Appl. Math. (1990) vol 5. SIAM, Philadelphia
3. Dem'yanov VF, Stavroulakis GE, Polyakova LN, Panagiotopoulos PD (1996) Quasidifferentiability and nonsmooth modelling in mechanics, engineering and economics. Kluwer, Dordrecht
4. Goeleven D (1996) Noncoercive variational problems and related results. Addison-Wesley, Indianapolis
5. Haslinger J, Stavroulakis GE (eds) (2006) Nonsmooth Mechanics of Solids. CISM Courses and Lectures, vol 485. Springer, Wien, New York
6. Haslinger J, Miettinen M, Panagiotopoulos PD (1999) Finite element method for hemivariational inequalities. Kluwer, Dordrecht
7. Hiriart-Urruty J-B, Lemaréchal C (1993) Convex analysis and minimization algorithms. Springer, Berlin
8. Koltsakis EK, Mistakidis ES, Tzaferopoulos MA (1995) On the numerical treatment of nonconvex energy problems in mechanics. J Glob Optim 6:427–448
9. Miettinen M, Mäkelä MM, Haslinger J (1995) On numerical solution of hemivariational inequalities by nonsmooth optimization methods. J Glob Optim 8(4):401–425
10. Mistakidis ES, Stavroulakis GE (1998) Nonconvex optimization in mechanics. Algorithms, heuristics and engineering applications by the F.E.M. Kluwer, Dordrecht
11. Naniewicz Z, Panagiotopoulos PD (1995) Mathematical theory of hemivariational inequalities and applications. Dekker, London
12. Panagiotopoulos PD (1983) Nonconvex energy functions. Hemivariational inequalities and substationary principles. Acta Mechanics 42:160–183
13. Panagiotopoulos PD (1985) Inequality problems in mechanics and applications. Convex and nonconvex energy functions. Birkhäuser, Basel. Russian translation, MIR, Moscow
14. Panagiotopoulos PD (1993) Hemivariational inequalities. Applications in mechanics and engineering. Springer, Heidelberg
15. Rockafellar RT (1981) The theory of subgradients and its applications to problems of optimization: Convex and nonconvex functions. Heldermann, Berlin
16. Tzaferopoulos MA, Liolios A (1997) On a branch and bound algorithm for the solution of a family of softening material problems of mechanics with applications to the analysis of metallic structures. J Glob Optim 11:133–149

Nonconvex Network Flow Problems

NNFP

BRUCE W. LAMAR
Economic and Decision Analysis Center,
The MITRE Corp., Bedford, USA

MSC2000: 90C26, 90C30, 90C35, 90B10

Article Outline

Keywords

Economy of scale; Global optimization; Nonconvex programming; Flows in networks; Fixed charge networks; Network design

The basic *network flow problem* can be formulated as

$$\begin{cases} \min & \displaystyle\sum_{(i,j)\in A} \phi_{ij}(x_{ij}) \\ \text{s.t.} & \displaystyle\sum_{(n,j)\in A} x_{nj} - \sum_{(i,n)\in A} x_{in} = b_n, \\ & \quad \forall n \in N, \\ & l_{ij} \le x_{ij} \le u_{ij}, \quad \forall (i,j) \in A, \end{cases} \tag{1}$$

where A is the (directed) arc set with generic element (i, j); N is the node set with generic element n; b_n is the *net supply* (if $b_n > 0$) or *net demand* (if $b_n < 0$) at node n; u_{ij} (respectively, l_{ij}) is the flow upper (respectively, lower) bound for arc (i, j); x_{ij} is the *flow decision variable* for arc (i, j); and ϕ_{ij} (x_{ij}) is the *cost function* for arc (i, j).

It is assumed that $\sum_{n \in N} b_n = 0$ (otherwise, a dummy supply node or demand node can be added to N to enforce this condition). The *objective function* in (1) minimizes total costs in the network; the first constraints are *node flow balance equations*; and the second constraints are *arc flow bounds*. Extensions to the basic network flow problem given above include *multicommodity networks*, *generalized networks* (i. e., networks with arc flow gains or losses), and *augmented networks* (i. e., formulations with additional constraints and/or decision variables in addition to x_{ij}) (see, for example, [1]).

If the cost function $\phi_{ij}(x_{ij})$ is a *nonconvex function* for one or more of the arcs in set A, then (1) is referred to as a *nonconvex network flow problem* (NNFP). The most commonly used cost function in a NNFP is a *fixed charge function*, of the form

$$\phi_{ij}(x_{ij}) = \begin{cases} 0 & \text{if } x_{ij} = 0, \\ \alpha_{ij} + \beta_{ij} \cdot x_{ij} & \text{if } x_{ij} > 0, \end{cases} \tag{2}$$

where α_{ij} and β_{ij} are coefficients with $\alpha_{ij} \geq 0$ (and $l_{ij} = 0$). By incurring the quantum of cost α_{ij} before flow can be carried on arc (i, j), fixed charge NNFPs can be used to model a variety of network design and expansion [29], lot sizing [17], and facility location [6,30] problems. Classical combinatorial problems, such as traveling salesperson and 0–1 knapsack problems, can also be represented as fixed charge NNFPs. In fact, *any* mathematical programming problem that can be formulated as an integer program with integer coefficients can be recast as a fixed charge NNFP [28].

Another common form of cost function in a NNFP is a concave quadratic function, of the form

$$\phi_{ij}(x_{ij}) = \alpha_{ij} + \beta_{ij} \cdot \left(x_{ij} - \gamma_{ij}\right)^2, \tag{3}$$

where α_{ij}, β_{ij}, and γ_{ij} are coefficients with $\beta_{ij} \leq 0$. Concave quadratic NNFPs are used to model arc flow economies of scale which are present in many communication [32], water resource [10], and physical distribution [22] problems.

A third type of cost function used in NNFPs is the 'sawtooth' function. A simple two-piece sawtooth function is given by

$$\phi_{ij}(x_{ij}) = \begin{cases} \alpha_{ij} \cdot x_{ij} & \text{if } x_{ij} < \gamma_{ij}, \\ \beta_{ij} \cdot x_{ij} & \text{if } x_{ij} \geq \gamma_{ij}, \end{cases} \tag{4}$$

where α_{ij}, β_{ij}, and γ_{ij} are coefficients with $\alpha_{ij} \geq \beta_{ij}$ and $l_{ij} \leq \gamma_{ij} \leq u_{ij}$. Functions of the form (4) are used to represent price-breaks and other types of all-units discounting [7] that occur, for instance, in network representations of inventory [33] and cash flow management problems [24].

The NNFP is in the class of NP-hard problems [14]. Thus, determining the *global minimum* to an NNFP is challenging because of the existence of many feasible points that are locally, but not globally, optimal. G.M. Guisewite [18] distinguishes between NNFPs with concave cost functions (such as (2) and (3)) and indefinite cost functions (that is, functions that are neither concave nor convex, such as (4)). For concave NNFPs, if the constraints in (1) are feasible, then there exists an optimal extreme point solution. Further, if the coefficients b_n, l_{ij}, and u_{ij} in (1) are integer-valued, then the optimal arc flows x^*_{ij} will also be integer-valued [4].

One of the earliest solution methods for concave NNFPs was proposed by B. Yaged [32]. This method uses successive linearizations of the concave cost function. It quickly converges to a local (but not necessarily global) minimum. Other approximate methods for concave NNFPs involve dual ascent [2,9], Lagrangian relaxation [11], local extreme point search techniques [13,21], and tabu search methods [16].

Exact methods for concave NNFPs generally rely on the underlying network topology. For problems with an arbitrary network topology, branch and bound procedures using rectangular partitioning are the most widely used techniques [3,5,26,30]. If the number of supply nodes or demand nodes in the set N is limited, then dynamic programming approaches are tractable [8,17,20]. Alternatively, if there are only a small number of arcs in the set A with a nonlinear cost function, then parametric programming techniques may be employed [25].

For the second type of NNFP – with an indefinite form of cost function – the optimal solution to (1) is not necessarily at an extreme point of the feasible region. For cases where the indefinite cost function is piecewise-linear but not necessarily continuous (as is the case in (4)), then the problem can be formulated and solved as a mixed integer program. If the indefinite function is continuous (and satisfies certain regularity properties), then it can be converted to a difference convex function (d.c. function) and the indefinite NNFP

can be solved as a specialized difference convex programming problem [31]. For more general functions, the indefinite NNFP can be converted (at least in principle) to a concave NNFP on an expanded network. Then, the concave NNFP solution techniques can be applied the problem on the expanded network [27].

See [12,15,18,19,22,23] for additional discussion of applications and solution methods for NNFPs.

See also

- ▶ Auction Algorithms
- ▶ Communication Network Assignment Problem
- ▶ Directed Tree Networks
- ▶ Dynamic Traffic Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Generalized Networks
- ▶ Global Supply Chain Models
- ▶ Inventory Management in Supply Chains
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ Multicommodity Flow Problems
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonoriented Multicommodity Flow Problems
- ▶ Operations Research Models for Supply Chain Management and Design
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs
2. Balakrishnan A, Magnanti TL, Wong RT (1989) A dual-ascent procedure for large scale uncapacitated network design. Oper Res 37:716–740
3. Bell GB, Lamar BW (1997) Solution methods for nonconvex network problems. In: Pardalos PM, Hearn DW, Hager WW (eds) Network Optimization. Lectures Notes Economics and Math Systems. Springer, Berlin, pp 32–50
4. Charnes A, Cooper WW (1961) Management models and industrial applications of linear programming. Wiley, New York
5. Cruz FRB, Macgregor Smith J, Mateus GR (1998) Solving to optimality the uncapacitiated fixed-charge network flow problem. Comput Oper Res 25:67–81
6. Daskin MS (1995) Network and discrete location. Wiley, New York
7. Dolan RJ (1987) Quantity discounts: Managerial issues and research opportunities. Marketing Sci 6:1–22
8. Erickson RE, Monna CL, Veinott AF (1987) Send-and-split method for minimum concave-cost network flows. Math Oper Res 12:634–664
9. Erlenkotter D (1978) A dual-based procedure for uncapacitated facility location. Oper Res 26:992–1009
10. Feltenmark S, Lindberg PO (1997) Network methods for head-dependent hydro power scheduling. In: Pardalos PM, Hearn DW, Hager WW (eds) Network Optimization, Lectures Notes Economics and Math Systems. Springer, Berlin, pp 249–264
11. Fisher ML (1985) An applications oriented guide to Lagrangian relaxation. Interfaces 15:10–21
12. Florian M (1986) Nonlinear cost network models in transportation analysis. In: Gallo G, Sandi C (eds) Netflow at Pisa, Math Program Stud. North-Holland, Amsterdam, pp 167–196
13. Gallo G, Sandi C, Sodini C (1980) An algorithm for the min concave cost flow problem. Europ J Oper Res 4:248–255
14. Garey MR, Johnson DS (1979) Computers and intractability. A guide to the theory of NP-completeness. Freeman, New York
15. Glover F, Klingman D, Phillips NV (1992) Network models in optimization and their applications in practice. Wiley, New York
16. Glover F, Laguna M (1997) Tabu search. Kluwer, Dordrecht
17. Graves SC, Orlin JB (1985) A minimum concave-cost dynamic network flow problem with applications to lot sizing. Networks 15:59–71
18. Guisewite GM (1995) Network models. In: Horst R, Pardalos PM (eds) Handbook Global Optim. Kluwer, Dordrecht, pp 609–648
19. Guisewite GM, Pardalos PM (1990) Minimum concave-cost network flow problems: Applications, complexity, and algorithms. Ann Oper Res 25:75–100
20. Guisewite GM, Pardalos PM (1991) Algorithms for the single-source uncapacitated minimum concave-cost network flow problem. J Global Optim 1:245–265
21. Guisewite GM, Pardalos PM (1992) Performance of local search in minimum concave-cost network flow problems. In: Floudas CA, Pardalos PM (eds) Recent Advances in Global Optimization. Princeton Univ. Press, Princeton, pp 50–75
22. Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer, Dordrecht

23. Jensen PA, Barnes JW (1980) Network flow programming. Wiley, New York
24. Jorjani S, Lamar BW (1994) Cash flow management network modelling with quantity discounting. OMEGA Internat J Management Sci 22:149–155
25. Klinz B, Tuy H (1993) Minimum concave-cost network flow problems with a single nonlinear arc cost. In: Du D-Z, Pardalos PM (eds) Network Optimization Problems: Algorithms, Applications, and Complexity. World Sci., Singapore, pp 125–146
26. Lamar BW (1993) An improved branch and bound algorithm for minimum concave cost network flow problems. J Global Optim 3:261–287
27. Lamar BW (1993) A method for solving network flow problems with general nonlinear arc costs. In: Du D-Z, Pardalos PM (eds) Network Optimization Problems: Algorithms, Applications, and Complexity. World Sci., Singapore, pp 147–167
28. Lamar BW (1996) A note on formulating nonconvex network optimisation problems. Techn Report Dept Management Univ Canterbury
29. Magnanti TL, Wong RT (1984) Network design and transportation planning: models and algorithms. Transport Sci 18:1–55
30. Soland RM (1974) Optimal facility location with concave costs. Oper Res 22:373–382
31. Tuy H (1995) D.C. optimization: Theory, methods, and algorithms. In: Horst R, Pardalos PM (eds) Handbook Global Optim. Kluwer, Dordrecht, pp 149–216
32. Yaged B (1971) Minimum cost routing for static network models. Networks 1:139–172
33. Zangwill WI (1968) Minimum concave cost flows in certain networks. Managem Sci 14:429–450

Nonconvex-nonsmooth Calculus of Variations

DANIEL GOELEVEN[1], DUMITRU MOTREANU[2]

[1] I.R.E.M.I.A. University de la Réunion, Saint-Denis, France

[2] Department Mat., University Al.I.Cuza, Iasi, Romania

MSC2000: 49J40

Article Outline

Keywords

Variational-hemivariational inequality; Generalized critical point; Unilateral mechanics

Problems in *unilateral mechanics* involving both monotone and nonmonotone unilateral boundary conditions have as variational formulations the so-called variational-hemivariational inequalities [5]. Solutions of these mathematical models can be determined as the critical points of some energy functionals which can usually be expressed as the sum of a locally Lipschitz function $\Phi\colon X \to \mathbf{R}$ and a proper, convex and lower semicontinuous function $\Psi\colon X \to \mathbf{R} \cup \{+\infty\}$. The critical points of $I := \Phi + \Psi$ are here defined as the solutions of the *variational-hemivariational inequality*:

$$\begin{aligned} &u \in X : \\ &\Phi^o(u; v-u) + \Psi(v) - \Psi(u) \geq 0, \quad \forall v \in X . \end{aligned} \tag{1}$$

For example, let us consider a linear elastic body identified with an open bounded subset $\Omega \subset \mathbf{R}^3$. The boundary Γ of the body Ω is assumed to consist of three open disjoint parts Γ_1, Γ_2 and Γ_3, i. e. $\Gamma = \overline{\Gamma}_1 \cup \overline{\Gamma}_2 \cup \overline{\Gamma}_3$. Let us denote by $u = (u_i)$, $\sigma = (\sigma_{ij})$, $\varepsilon = (\varepsilon_{ij})$, $S = (S_i)$ the displacement vector, the stress tensor, the strain tensor and the stress vector, respectively. It is supposed that the body is subject to a body density force $f \in L^2(\Omega; \mathbf{R}^3)$. Moreover, one compels the part Γ_1 of the body to satisfy the constraint

$$u(x) \in Q(x), \quad \forall x \in \Gamma_1 , \tag{2}$$

where $Q(x)$ is defined as follows:

$$Q(x) := \left\{ y \in \mathbf{R}^3 : \ g(x, y) \leq 0 \right\} , \tag{3}$$

where $g\colon \Gamma_1 \times \mathbf{R}^3 \to \mathbf{R}$ is continuous and convex in the second variable. Moreover, we assume that $g(x, 0) \leq 0$, $\forall x \in \Gamma_1$. Thus $Q(x)$ is a nonempty, closed and convex subset for all $x \in \Gamma_1$. The formulation of these unilateral constraints has to encompass the associated forces of constraints $r(x)$. We assume a normal reaction law of the form

$$-r(x) \in N_{Q(x)}(u(x)), \quad \forall x \in \Gamma_1 , \tag{4}$$

where for a vector $v \in \mathbf{R}^3$, $N_{Q(x)}(v)$ denotes the normal cone of $Q(x)$ at v. Note that the system (2–4) is equivalent to the variational inequality

$$\begin{aligned} &u(x) \in Q(x) : \\ &r(x)^\top (v - u(x)) \geq 0, \quad \forall v \in Q(x), \quad \forall x \in \Gamma_1 . \end{aligned} \tag{5}$$

The part Γ_2 (it is assumed that meas$(\Gamma_2) > 0$) of the boundary is assumed to be fixed, that is,

$$u(x) = 0, \quad \forall x \in \Gamma_2 . \tag{6}$$

On Γ_3, we consider conditions of a unilateral contact between the body and a Winkler-type support which may sustain only limited values of traction, that is (we consider the usual decompositions $v = v_N n + v_T$, $S = S_N n + S_T$, v_N, $S_N \in \mathbf{R}$, v_T, $S_T \in \mathbf{R}^3$, where n denotes the unit outward normal vector on Γ):

$$-S_N \in \partial j(x, u_N(x)), \quad \forall x \in \Gamma_3 , \tag{7}$$

where

$$j(x, y) := \begin{cases} 0 & \text{if} \quad y < 0, \\ \frac{k_o(x) y^2}{2} & \text{if} \quad 0 \le y < \varepsilon, \\ \frac{k_o(x) \varepsilon^2}{2} & \text{if} \quad y \ge \varepsilon, \end{cases}$$

and

$$S_T = C_T \quad \text{on} \quad \Gamma_3 \tag{8}$$

with C_T given in $L^2(\Gamma_3; \mathbf{R}^3)$. We refer the reader to [4] for the details concerning the formulation of the subdifferential law (7). From (7), (8) and the orthogonality between $S_N n$ and u_T and between S_T and $u_N n$, we obtain the hemivariational inequality

$$S^\top v + j^o_y(x, u_N(x); v_N) - C_T^\top v_T \ge 0,$$
$$\forall v \in \mathbf{R}^3, \quad \forall x \in \Gamma_3 .$$

Here $\partial\, j(x, \cdot)$ stands for the *Clarke subdifferential* of j with respect to the second variable while $j^o_y(x, u; v)$ denotes the generalized directional derivative of j at u in the direction v [2]. In the framework of a small deformation theory, one has the equilibrium equations

$$\sigma_{ij,j} + f_i = 0 , \tag{9}$$

$$\varepsilon_{ij}(u) = \frac{1}{2}(u_{i,j} + u_{j,i}) , \tag{10}$$

$$\sigma_{ij} = C_{ijkl} \varepsilon_{kl}(u) , \tag{11}$$

where $C_{ijkl} \in L^\infty(\Omega)$ denotes the elasticity tensor assumed to satisfy the usual symmetry and ellipticity properties. Suppose now that all the data are sufficiently smooth to justify the following computations. From (9) and for a virtual displacement $v - u$ we obtain the equation

$$-\int_\Omega \sigma_{ij,j}(v_i - u_i)\, dx = \int_\Omega f^\top (v - u)\, dx.$$

Using the Green–Gauss theorem and relations (10) and (11), we get

$$\int_\Omega C_{ijkl} \varepsilon_{ij}(u) \varepsilon_{kl}(v - u)\, dx$$
$$= \int_\Omega f^\top (v - u)\, dx + \int_\Gamma S^\top (v - u)\, ds .$$

Using now the boundary conditions, we derive for v and u satisfying (6):

$$\int_\Omega C_{ijkl} \varepsilon_{ij}(u) \varepsilon_{kl}(v - u)\, dx$$
$$= \int_\Omega f^\top (v - u)\, dx + \int_{\Gamma_3} S_N (v_N - u_N)\, ds$$
$$+ \int_{\Gamma_3} C_T^\top (v_T - u_T)\, ds + \int_{\Gamma_1} r^\top (v - u)\, ds .$$

A combination of this last expression expressing the principle of virtual work with the mechanical laws (4) and (7), we obtain the variational-hemivariational inequality problem: Find u satisfying (2) and (6) such that

$$\int_\Omega C_{ijkl} \varepsilon_{ij}(u) \varepsilon_{kl}(v - u)\, dx$$
$$+ \int_{\Gamma_3} j^o(x, u_N; v_N - u_N)\, ds$$
$$- \int_\Omega f^\top (v - u)\, dx - \int_{\Gamma_3} C_T^\top (v_T - u_T)\, ds \ge 0 ,$$

for all v satisfying (2) and (6). This mathematical model expresses the principle of virtual works in its inequality form. Let us now present a suitable framework to formulate this last inequality model. The boundary Γ of the body is assumed sufficiently regular and we denote by γ: $H^1(\Omega, \mathbf{R}^3) \to H^{1/2}(\Gamma; \mathbf{R}^3)$, γ_N: $H^1(\Omega; \mathbf{R}^3) \to H^{1/2}(\Gamma)$ and γ_T: $H^1(\Omega; \mathbf{R}^3) \to H_T$ (see e. g., [6] for the notations) the usual trace mapping, normal trace mapping and tangential trace mapping, respectively. We set

$$X = \{u \in H^1(\Omega; \mathbf{R}^3) : \gamma(u(x)) = 0 \text{ a.e. } x \in \Gamma_2\}$$

and

$$C = \{u \in X : \gamma(u(x)) \in Q(x) \text{ a.e. } x \in \Gamma_1\} .$$

We define the operator $A: X \to X'$, the element $h \in X'$ and the functional $J: X \to \mathbf{R}$ by the formulas:

$$\langle Au, v\rangle := \int_\Omega C_{ijkl}(x)\varepsilon_{ij}(u)\varepsilon_{kl}(v)\ \mathrm{d}x, \quad \forall u, v \in X,$$

$$\langle h, v\rangle := \int_\Omega f_i v_i\ \mathrm{d}x + \int_{\Gamma_3} C_T^\top v_T\ \mathrm{d}s, \quad \forall v \in X,$$

$$J(v) := \int_{\Gamma_3} j(x, \gamma_N(v(x)))\ \mathrm{d}s, \quad \forall v \in X\,.$$

Using the rules concerning the subdifferentiation of integral functionals and composite mappings [2], we can show that

$$J^o(u, v) \le \int_{\Gamma_3} j^o(x, \gamma_N(u(x)); \gamma_N(u(x)))\ \mathrm{d}s, \quad \forall u, v \in X\,. \tag{12}$$

Setting

$$\Phi(u) = \frac{1}{2}\langle Au, u\rangle - \langle h, u\rangle + J(u)$$

and using the relation (12) we see that the equilibriums of our system can be determined by means of the variational-hemivariational inequality:

$$u \in C: \quad \Phi^o(u; v - u) \ge 0, \quad \forall v \in C,$$

or equivalently

$$u \in X: \quad \Phi^o(u; v - u) + \Psi_C(v) - \Psi_C(u) \ge 0, \quad \forall v \in X\,, \tag{13}$$

where Ψ_C denotes the indicator function of C. It is now natural to introduce a concept of *generalized critical point* of the energy functional $I = \Phi + \Psi_C$ as a solution of the variational-hemivariational inequality (13) (which is a particular case of the general model given in (1)).

Various other examples in mechanics leading to a variational-hemivariational inequality like (1) are described in [5]. As a rule, the formulation of concrete problems involving both monotone and nonmonotone boundary conditions introduces a general nonsmooth and nonconvex energy functional (expressed as the sum of a locally Lipschitz function $\Phi: X \to \mathbf{R}$ and a proper, convex and lower semicontinuous function $\Psi: X \to \mathbf{R} \cup \{+\infty\}$) whose critical points are defined as the solutions of the variational-hemivariational inequality (1). If $\Phi \in C^1(X; \mathbf{R})$ then problem (1) reduces to a classical variational inequality. A critical point theory to deal with such case has been developed by A. Szulkin [7]. Defining a compactness condition of Palais–Smale type related to the variational inequality model and using the famous Ekeland principle, Szulkin provided a deformation lemma and extended the well-known mountain pass theorem, the saddle point theorem, the main results for even functionals, etc. Several examples including variational inequalities are given in [7]. If $\Psi = 0$, the problem (1) reduces to the problem studied by K.-C. Chang [1]. Motivated by the study of partial differential equations involving discontinuous nonlinearities, Chang extended the concept of critical point, the Palais–Smale condition and the deformation lemma so as to be applicable to locally Lipschitz functionals. The minimax method developed by Chang is now one of the most efficient and appreciated tools to deal with hemivariational inequalities arising in unilateral mechanics and partial differential equations involving discontinuous nonlinearities. It has been in particular extensively used by D. Goeleven, D. Motreanu, and P.D. Panagiotopoulos [3] so as to develop in several directions the theory of hemivariational inequalities (see for instance the references cited in [3]). So, the theory of Szulkin and the one of Chang has been shown very efficient to deal with problems in unilateral mechanics involving some given classes of boundary conditions. However, one has seen it above, to deal efficiently with problems in unilateral mechanics involving both monotone and nonmonotone boundary conditions it is necessary to deal with a critical point theory for functions which can be written as the sum of a locally Lipschitz function and a proper, convex and lower semicontinuous function. Such theory has now recently been developed by Goeleven, Motreanu, and Panagiotopoulos [3]. The approach combines the approach of Szulkin and the one of Chang in a nontrivial way. A general linking theorem is obtained and then used to generalize the mountain pass theorem, the saddle point theorem and the main results for even functionals. This last theory constitutes a contribution in the field of nonconvex-nonsmooth calculus of variations that unifies the theory of Szulkin

and Chang. Moreover it can be used to develop the theory of variational-hemivariational inequalities and consequently to study various concrete problems in mechanics involving different types of unilateral boundary conditions.

See also

- ▶ Composite Nonsmooth Optimization
- ▶ Generalized Monotonicity: Applications to Variational Inequalities and Equilibrium Problems
- ▶ Hemivariational Inequalities: Applications in Mechanics
- ▶ Hemivariational Inequalities: Eigenvalue Problems
- ▶ Hemivariational Inequalities: Static Problems
- ▶ Nonconvex Energy Functions: Hemivariational Inequalities
- ▶ Nonsmooth and Smoothing Methods for Nonlinear Complementarity Problems and Variational Inequalities
- ▶ Quasidifferentiable Optimization
- ▶ Quasidifferentiable Optimization: Algorithms for Hypodifferentiable Functions
- ▶ Quasidifferentiable Optimization: Algorithms for QD Functions
- ▶ Quasidifferentiable Optimization: Applications
- ▶ Quasidifferentiable Optimization: Applications to Thermoelasticity
- ▶ Quasidifferentiable Optimization: Calculus of Quasidifferentials
- ▶ Quasidifferentiable Optimization: Codifferentiable Functions
- ▶ Quasidifferentiable Optimization: Dini Derivatives, Clarke Derivatives
- ▶ Quasidifferentiable Optimization: Exact Penalty Methods
- ▶ Quasidifferentiable Optimization: Optimality Conditions
- ▶ Quasidifferentiable Optimization: Stability of Dynamic Systems
- ▶ Quasidifferentiable Optimization: Variational Formulations
- ▶ Quasivariational Inequalities
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Solving Hemivariational Inequalities by Nonsmooth Optimization Methods
- ▶ Variational Inequalities
- ▶ Variational Inequalities: F. E. Approach
- ▶ Variational Inequalities: Geometric Interpretation, Existence and Uniqueness
- ▶ Variational Inequalities: Projected Dynamical System
- ▶ Variational Principles

References

1. Chang K-C (1981) Variational methods for non-differentiable functionals and their applications to partial differential equations. J Math Anal Appl 80:102–129
2. Clarke FH (1984) Nonsmooth analysis and optimization. Wiley, New York
3. Goeleven D, Motreanu D, Panagiotopoulos PD (1997) General minimax methods for variational-hemivariational inequalities. Preprint
4. Naniewicz Z (1989) On some nonmonotone subdifferential boundary conditions in elastostatics. Ingen Archiv 60:31–40
5. Naniewicz Z, Panagiotopoulos PD (1995) The mathematical theory of hemivariational inequalities and applications. M. Dekker, New York
6. Panagiotopoulos PD (1993) Hemivariational inequalities. Applications in mechanics and engineering. Springer, Berlin
7. Szulkin A (1986) Minimax principles for lower semicontinuous functions and applications to nonlinear boundary value problems. Ann Inst Henri Poincaré 3:77–109

Nondifferentiable Optimization

Nonsmooth Optimization, NDO

Samir Elhedhli[1], Jean-Louis Goffin[1], Jean-Philippe Vial[2]

[1] GERAD/Fac. Management, McGill University, Montreal, Canada

[2] Logilab/Department Management Stud., University Genève, Geneva, Switzerland

MSC2000: 90-00, 90C47, 46N10

Article Outline

Keywords

Nondifferentiable optimization; Nonsmooth optimization

Nondifferentiable, or nonsmooth, optimization (NDO) is concerned with problems where the smoothness assumption on the functions involved is relaxed. 'Nondifferentiability' means that the gradient does not exist, implying that the function may have kinks or corner points. Consequently, the function cannot be approximated locally by a tangent hyperplane, or by a quadratic approximation. In NDO, the smoothness assumption is usually replaced by weaker ones, which at least guarantees the existence of directional derivatives.

NDO problems arise in a variety of contexts, and methods designed for smooth optimization may fail to solve them. This justifies developing specialized theory and methods that are the object of this short introduction. In the sequel, we will often refer to *convex NDO*, a subclass of nondifferentiable optimization, in which functions are further assumed to be convex. Due to its global property, convexity allows stronger convergence results and finer analyses. Yet, the difficulties linked with the presence of kinks remain an important aspect, justifying special interest for this class of problems.

In the following Section, we give some basic definitions, then discuss examples of nondifferentiable optimization problems and finally, describe a few different solution techniques.

Basic Definitions

The basic nondifferentiable optimization problem takes the form

$$[\text{NDP}] \quad \min_{x \in \mathbf{R}^n} f(x), \tag{1}$$

where f is a real valued, continuous, nondifferentiable function. Convexity of f implies that it has at least one supporting hyperplane at every point of $\mathbf{R}^n$. The slopes of such hyperplanes form the set of *subgradients*, which is known as the *subdifferential set* or the *generalized gradient* [7]. At differentiable points there is a unique supporting hyperplane whose slope is the gradient. At nondifferentiable points, there is an infinite set of subgradients and, hence, an infinite set of supporting hyperplanes.

A supporting hyperplane to f at a point x_0 is given by

$$y = f(x_0) + \xi_0^\top (x - x_0),$$

where ξ_0 is any element of the subdifferential $\partial f(x_0)$ of f at x_0. Recalling the fact that it is a supporting hyperplane leads to the *subgradient inequality*

$$f(x_0) + \xi_0^\top (x - x_0) \le f(x). \tag{2}$$

Subgradients are defined by this inequality.

Determining the whole subdifferential set is generally an extremely difficult, or impossible, task. If the function f is polyhedral, the number of extreme points of the subdifferential may be exponential in the dimension of the underlying space. A complete description of the subdifferential can be accomplished for simple situations, such as the one when f is the maximum of a finite number of convex differentiable functions: $f(x) = \max_{i \in I} f_i(x)$. The subdifferential $\partial f(x_0)$ is then given by

$$\partial f(x_0) = \left\{ \sum_{i \in I(x_0)} \alpha_i \nabla f_i(x_0) : \begin{array}{c} \sum_{i \in I(x_0)} \alpha_i = 1, \\ \alpha_i \ge 0 \end{array} \right\}$$

$$I(x_0) = \{ i : \ f_i(x_0) = f(x_0) \} \ .$$

When f is a Lipschitz function, the subdifferential set can be defined as being the set of cluster points of the gradients $\nabla f(x_i)$ as a sequence of differentiable points x_i approaches x [7]. The precise definition of $\partial f(x_0)$ is given by

$$\text{conv} \{ \lim \nabla f(x_i) : x_i \to x_0 : \ \nabla f(x_i) \text{ exists} \} \ .$$

In nondifferentiable optimization, the whole subdifferential set is never calculated. Subgradients are calculated when needed and often a single element suffices. It is common practice to isolate the procedures for calculating subgradients into an oracle. The number of calls to the oracle can be a basis for comparing different NDO methods.

A natural solution method in nonsmooth analysis is an iterative method, where a search is done following descent directions. A descent direction is one along

which a small movement of f leads to a strict improvement. In other words

$$f'(x_0; d) = \lim_{t \to 0} \frac{f(x + td) - f(x)}{t}$$

should be strictly negative. $f'(x_0;d)$ is called the *directional derivative* and it is related to the subgradient through the relation

$$f'(x_0; d) = \sup \left\{ \xi^\top d : \ \xi \in \partial f(x_0) \right\}. \tag{3}$$

This relation implies that for d to be a descent direction, $-d$ has to make an acute angle with every subgradient of f at x_0.

Sources of NDO Problems

Nonsmooth problems are encountered in many disciplines. In some instances, they occur naturally and in others they result from mathematical transformations.

In statistics, for example, rectilinear data fitting, which was long discovered to be superior to the Euclidean one – it has the advantage of overcoming the effect of outliers, [25] – results directly in an NDO problem. Similarly, functions involving ℓ_1 or ℓ_∞ norms, Euclidean or Chebyshev distances, a maximum of convex functions are typical NDO problems. As an example, the ℓ_∞ solution of an overdetermined linear system is found by solving the nondifferentiable convex function:

$$\min_{x \in \mathbf{R}^n} \|Ax - b\|_\infty = \min_{x \in \mathbf{R}^n} \max_{i=1,\ldots,m} \left| a_i^\top x - b_i \right|, \tag{4}$$

where $x \in \mathbf{R}^n$, $b \in \mathbf{R}^m$ and $A \in \mathbf{R}^{m \times n}$ with rows $a_i^\top$. This problem can be traced back to the Russian mathematician P.L. Chebyshev who studied it in the 1850s [25].

Among the mathematical transformations that lead to NDO problems is the technique that changes constrained problems into unconstrained ones through the use of exact penalty functions [11] (cf. also ► Quasidifferentiable optimization: Exact penalty methods). Equality constraints, $\phi(x) = 0$ and inequality constraints $\varphi(x) \leq 0$ are placed in the objective using penalty parameters and nondifferentiable functions $|\phi(x)|$ and $\max\{0, \varphi(x)\}$, respectively. In other word, a solution to the constrained problem

$$\begin{cases} \min & f(x) \\ \text{s.t.} & \phi(x) = 0, \\ & \varphi(x) \leq 0, \end{cases} \tag{5}$$

is determined by solving

$$\min f(x) + t_1 |\phi(x)| + t_2 \max\{0, \varphi(x)\}$$

for large enough values of t_1 and t_2.

Still, the major source of optimization problems are master problems resulting from the application of relaxation/restriction techniques such as Lagrangian relaxation [14] (cf. also ► Integer programming: Lagrangian relaxation), [12], Benders decomposition (cf. also ► Generalized Benders decomposition) [4,13] and Dantzig–Wolfe decomposition [9,10].

These different approaches are conceptually similar, at least in the linear case, and end up solving the same NDO problem. To show that, let us consider the linear program

$$[LP] \begin{cases} \min & c^\top x \\ \text{s.t.} & Ax \geq b, \\ & Dx \geq d, \\ & x \geq 0. \end{cases}$$

Where, we assume for the ease of exposition that {x: $Ax \geq b; x \geq 0$} is a bounded, nonempty polytope. The dual of [LP] is

$$[LD] \begin{cases} \max & b^\top u + d^\top v \\ \text{s.t.} & A^\top u + D^\top v \leq c, \\ & u, v \geq 0. \end{cases}$$

Applying Lagrangian relaxation to [LP] is equivalent to relaxing $Dx \geq d$ using positive dual multipliers v, leading to

$$\max_{v \geq 0} \left\{ \min_{x \geq 0} c^\top x + v^\top (d - Dx) : \ Ax \geq b \right\}. \tag{6}$$

Benders decomposition applied to [LD] results in

$$\max_{v \geq 0} \left\{ \max_{u \geq 0} b^\top u + d^\top v : \ A^\top u \leq c - D^\top v \right\},$$

where v is assumed to be the complicating variable. Replacing the inside problem by its dual leads to

$$\max_{v \geq 0} \left\{ \min_{x \geq 0} c^\top x + v^\top (d - Dx) : \ Ax \geq b \right\}. \tag{7}$$

Dantzig–Wolfe decomposition replaces [LP] by its convex representation in terms of the convex points of

$\{x: Ax \geq b; x \geq 0\}$ that is indexed by $\mathcal{E}$, to get

$$\begin{cases} \min & \sum_{h\in\mathcal{E}} \alpha_h (c^\top x^h) \\ \text{s.t.} & \sum_{h\in\mathcal{E}} \alpha_h (Dx^h) \geq d, \\ & \sum_{h\in\mathcal{E}} \alpha_h = 1, \\ & \alpha_h \geq 0, \quad h \in \mathcal{E}. \end{cases}$$

Taking the dual results in

$$\begin{cases} \max\limits_{v\geq 0, v_0} & v_0 + d^\top v \\ \text{s.t.} & c^\top x^h + v^\top D x^h \geq v_0, \quad \forall h \in \mathcal{E}, \end{cases}$$

which is equivalent to

$$\max_{v\geq 0} \left\{ \min_{h\in\mathcal{E}} c^\top x^h + v^\top (d - Dx^h) \right\}. \tag{8}$$

The equivalence between (6) and (7) is obvious. Using the fact that there is always an extreme point solution to a linear program, the equivalence between (6), (7) and (8) is established. Therefore, Lagrangian relaxation applied to the primal is exactly Benders decomposition applied to the dual, and is equivalent by duality to Dantzig–Wolfe decomposition. Furthermore, all three solve (8), which is the maximum of a concave piecewise linear function that is nondifferentiable at intersection points.

F.H. Clarke [7] and [8] discusses further examples from physics, engineering, economics and optimal control. Other mathematical problems leading to NDO optimization include semi-infinite programming, eigenvalue optimization and variational inequalities [16].

Solution Approaches

Due to the existence of successful solution methods for differentiable optimization, one other solution approach tries to transform nonsmooth problem into smooth ones. As an example, the absolute value function $|x|$, which is nondifferentiable at zero can be approximated by

$$\begin{cases} -x, & x \geq t, \\ \frac{x^2}{t}, & -t \leq x \leq t, \\ x, & x \geq t, \end{cases}$$

for small values of the parameter t. For these transformations to be successful, the right transformation should be found and the the nondifferentiable points should be known. A solution approach based on this transformation is discussed in [21].

Other solution approaches that try to eliminate nondifferentiability, do so by transforming an unconstrained nonsmooth problem into a constrained smooth one. This approach is highly efficient for problems that can be transformed into easily solvable constrained problems such as linear programs. The ℓ_∞ optimization problem described in (4) is equivalent to the linear program

$$\begin{cases} \min & y \\ \text{s.t.} & Ax - ye \leq b, \\ & Ax + ye \geq b, \end{cases}$$

where e is the appropriate dimension vector whose entries are all ones. Being a linear program with a special type of matrix, most linear programming techniques were modified to solve (4). This includes the simplex-like algorithm in [3] and the interior point algorithms in [23] and [27].

Subgradient Methods

The first methods for nondifferentiable optimization tried to extend the gradient-based methods that were successful for smooth optimization. The transition from gradients to subgradients is not straightforward as some subgradient-based search direction are not necessarily improving directions. P. Wolfe [26] gives an example where the extension of the steepest descent method fails. To overcome that, some designed methods [18,20] will only take a serious step when the next iterate is a better one.

Subgradient methods (cf. also ► Nondifferentiable optimization: Subgradient optimization methods) were developed by N.Z. Shor [24] in the 1960s. They are basically an iterative technique where iterates are updated using a current subgradient and a carefully-chosen stepsize. Applied to (1), iterates are given by

$$x_{k+1} = x_k + t_k \xi_k,$$

where x_k is the current point, ξ_k is a subgradient of f at x_k and t_k is a stepsize. Shor [24] states that a constant

stepsize does not converge, even for the simple function $|x|$. He proposes the use of a stepsize that satisfies

$$\sum_{k=0}^{\infty} t_k = \infty, \quad t_k \to 0.$$

In practice, the most widely used stepsize is $\theta[f(x_k) - f^*]/\| \xi_k \|$ where $\theta \in (0, 2]$ and f^* is the best estimate of the optimal value $f(x^*)$.

Steepest Descent and *E*-Subgradient Methods

Subgradient methods are not monotonic, as they do not guarantee to improve the value of the minimized function. Descent methods are designed to overcome this drawback. As an example we discuss the steepest descent method which chooses its search direction by solving

$$\min_{\|d\| \le 1} f'(x; d).$$

Using relation (3), the steepest descent direction, at a point x_k, is given by

$$d_k = \frac{\xi_k}{\|\xi_k\|}; \quad \xi_k = \arg \max_{\xi \in \partial f(x_0)} \|\xi\| .$$

The method proceeds iteratively, updating the iterates by

$$x_{k+1} = x_k + \alpha_k \xi_k$$

and choosing the steplength α_k so that $f(x_{k+1}) < f(x_k)$.

The main difficulty with the steepest descent resides in the calculation of the direction d_k which necessitates the knowledge of the whole differential set $\partial f(x)$. To overcome that, *ϵ-subgradient methods* prefer to calculate approximate steepest descent direction by searching through subgradients of neighboring points through the use of the *ϵ-subdifferential set*

$$\partial_\epsilon f(x) = \{\xi : \ f(x_0) + \xi(x - x_0) + \epsilon \le f(x), \ \forall x\} .$$

Details of the method can be found in [5].

Cutting Plane Methods

J.E. Kelley [17] and W. Cheney and A.A. Goldstein [6] were the first to realize the potential of such methods for convex programming. Applied to (1), cutting plane algorithms use the subgradient inequality to approximate f by

$$f(x) \cong \max_{i \in I} f(x_i) + \xi_i^\top (x - x_i),$$

where ξ_i^f, $i \in I$ are subgradients of f at x_i, $i \in I$. Thus, (1) is replaced by

$$\min_x \left\{ \max_{i \in I} f(x_i) + \xi_i^\top (x - x_i) \right\},$$

which is equivalent to,

$$\begin{cases} \min & v \\ \text{s.t.} & f(x_i) + \xi_i^\top (x - x_i) \le v, \quad \forall i \in I. \end{cases} \tag{9}$$

Problem (9) is a linear program that is easier to deal with than the original problem. It is to note, however, that this is only an approximation of (1), which gets better as more constraints are added. Let us denote by [MP$_k$] the relaxed master problem (9) with index set I_k.

By transforming (1) to (9), a nondifferentiable problem is replaced by a constrained problem having a large number of constraints. Cutting plane methods use only a subset of these constraints and generate the rest as needed. In fact, they would solve a series of relaxed master problems [MP$_k$] and stop when an optimal (satisfactory) solution to (1) is reached.

Various cutting plane methods were proposed over the years. Each variant generates cuts at a different point called the *query point*. *Kelley's classical cutting plane method* [17] chooses the minimum of the relaxed master problem [MP$_k$] as a query point. Although, it may work well for some problems, this method suffers from slow convergence [22]. The *analytic center cutting plane method* (ACCPM) [15,16], on the other hand, chooses the analytic center as its query point. Its calculation makes use of interior point concepts and has shown promising results for a number of applications [1,2]. *Bundle methods* [19,20], choose the query point by solving a quadratic program that contains a small number of cutting planes. The information (bundle of cutting planes) is updated regularly and kept moderately small.

Conclusion

Nondifferentiable optimization tackles a class of problems that are intractable to classical optimization meth-

ods. Most of the theory is based on the notion of subgradients and most of the work is done for the convex case. It has an abundance of applications in real life, because the nondifferentiability aspect captures some of the inherent complexity in real-life problems. Like all disciplines, favoring an easily implementable and understood method will not necessarily lead to a good solution method. This corresponds to the subgradient method in NDO. Although it is easily implementable, it has slow convergence. More sophisticated methods, such as bundle methods or ACCPM are more promising from a computational point of view but require more knowhow of the method and of numerical linear algebra.

See also

- ▶ Dini and Hadamard Derivatives in Optimization
- ▶ Discontinuous Optimization
- ▶ Global Optimization: Envelope Representation
- ▶ Nondifferentiable Optimization: Cutting Plane Methods
- ▶ Nondifferentiable Optimization: Minimax Problems
- ▶ Nondifferentiable Optimization: Newton Method
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Nondifferentiable Optimization: Relaxation Methods
- ▶ Nondifferentiable Optimization: Subgradient Optimization Methods

References

1. Bahn O, Du Merle O, Goffin JL, Vial JP (1995) A cutting plane method from analytic centers for stochastic programming. Math Program B 69:45–73
2. Bahn O, Goffin JL, Vial JP, Du Merle O (1994) Implementation and behavior of an interior point cutting plane algorithm for convex programming: An application to geometric programming. Discrete Appl Math 49:3–23
3. Barrodale I, Phillips C (1974) An improved algorithm for discrete Chebychev linear approximation. In: Proc. Fourth Manitoba conference on Numerical Mathematics. Univ Manitoba, Winnipeg
4. Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Math 4:238–252
5. Bertsekas DP (1995) Nonlinear programming. Athena Sci., Belmont
6. Cheney W, Goldstein AA (1959) Newton's method for convex programming and Chebyshev approximation. Numerische Math 1–5:253–268
7. Clarke FH (1989) Optimization and nonsmooth analysis. Les Publ. CRM, Montreal
8. Clarke FH, Ledyaev Yu, Ledyaev S, Stern RJ, Wolenski PR (1998) Nonsmooth analysis and control theory. Grad Texts Math. Springer, Berlin
9. Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. Oper Res 8:101–111
10. Dantzig GB, Wolfe P (1961) The decomposition algorithm for linear programming. Econometrica 29(4):767–778
11. Fiacco AV, McCormick GP (1968) Nonlinear programming: Sequential unconstrained minimization techniques. Wiley, New York, Reprint: (1990) SIAM Classics Appl Math, vol 4. SIAM, Philadelphia
12. Fisher ML (1981) The Lagrangian relaxation method for solving integer programming problems. Managem Sci 27:1–18
13. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
14. Geoffrion AM (1974) Lagrangean relaxation for integer programming. Math Program Stud 2:82–114
15. Goffin J-L, Haurie A, Vial J-P (1992) Decomposition and nondifferentiable optimization with the projective algorithm. Managem Sci 38(2):284–302
16. Goffin J-L, Vial J-P (1998) Interior point methods for nondifferentiable optimization. In: Kishka P, Lorenz HW, Derigs U, Domschke W, Kleinschmidt P, Moehring R (eds) Operations Research Proc. 1997. Springer, Berlin, pp 35–49
17. Kelley JE (1960) The cutting plane method for solving convex programs. J SIAM 8:703–712
18. Kiwiel KC (1985) Methods of descent for nondifferentiable optimization. Lecture Notes Math, vol 1133. Springer, Berlin
19. Kiwiel KC (1990) Proximity control in Bundle methods for convex nondifferentiable optimization. Math Program 46:105–122
20. Lemaréchal C (March 28-April 8, 1977 1978) Bundle methods in nonsmooth optimization. In: Lemaréchal C, Mifflin R (eds) Nonsmooth Optimization: Proc. IIASA Workshop. Pergamon, Oxford
21. Madsen K, Nielsen HB, Pnar MÇ (1993) New characterizations of $\ell 1$ solutions of over-determined linear systems. Oper Res Lett 16
22. Nemirovskii AS, Yudin DB (1983) Problem complexity and method efficiency in optimization. Wiley, New York
23. Ruzinsky SA, Olsen ET (1989) $\ell 1$ and $\ell\infty$ minimization via a variant of Karmarkar's algorithm. IEEE Trans Acoustics, Speech and Signal Processing 37
24. Shor NZ (March 28-April 8, 1977 1978) Subgradient methods: A survey of soviet research. In: Lemaréchal C, Mifflin R (eds) Nonsmooth optimization: Proc. IIASA workshop. Pergamon, Oxford

25. Watson GA (1980) Approximation theory and numerical methods. Wiley, New York
26. Wolfe P (1975) A method of conjugate subgradients for minimizing nondifferentiable functions. Math Program Stud 3:145–173
27. Zhang Y (1993) Primal-dual interior point approach for computing the $\ell 1$-solutions and $\ell\infty$-solutions of overdetermined linear systems. J Optim Th Appl 77(2)

Nondifferentiable Optimization: Cutting Plane Methods

Samir Elhedhli[1], Jean-Louis Goffin[1], Jean-Philippe Vial[2]
[1] McGill University, Montreal, Canada
[2] Logilab/Department Management Stud., University Genève, Geneva, Switzerland

MSC2000: 49M20, 90C25, 90-08

Article Outline

Keywords

Nondifferentiable optimization; Nonsmooth optimization; Cutting planes

Cutting plane methods were proposed independently by J.E. Kelley [27] and W. Cheney and A.A. Goldstein [5] as a solution technique for constrained convex optimization problems. Although they may not compete with some of the efficient methods for smooth optimization, they are one of the first and fundamental solution approaches for nondifferentiable optimization (cf. ▶ Nondifferentiable optimization).

The fact that they rely on polyhedral approximations of convex functions, makes the technique suitable for nondifferentiable optimization. It forms with subgradient optimization [38] the two major solution approaches for nondifferentiable problems. The cutting plane algorithms that are being designed over the years are getting more sophisticated and ultimately, showing promising computational results and stable numerical properties.

In this article, we give an overview of cutting plane methods for nondifferentiable optimization problems. We start with a brief introduction to cutting planes. We then give a generic cutting plane algorithm, that will be used to describe some of the main variants.

A Generic Cutting Plane Algorithm

To describe the general cutting plane approach, we consider the following nondifferentiable problem

$$[NDP] \begin{cases} \min & f(x) \\ \text{s.t.} & g(x) \le 0\,, \end{cases}$$

where f and g are real-valued, continuous, nondifferentiable, convex functions. Convexity implies that there is at least one supporting hyperplane to f at every point x_0 of the domain, whose equation is given by

$$y = f(x_0) + \xi_0^f (x - x_0)\,,$$

where ξ_0^f is any element of the subdifferential $\partial f(x_0)$ of f at x_0. (For ease of notation we assume that subgradients are row vectors.) Recalling that a supporting hyperplane gives an under-estimate of f, the *subgradient inequality*

$$f(x_0) + \xi_0^f (x - x_0) \le f(x) \tag{1}$$

can be used to approximate f by the maximum of a set of piecewise linear functions. Therefore, given a set of points x_i, $i \in I$ and their corresponding subgradients ξ_i^f, f is tangentially approximated by

$$\overline{f}(x) = \max_{i \in I} \left\{ f(x_i) + \xi_i^f (x - x_i) \right\}\,. \tag{2}$$

Inequality (1) implies that $\overline{f}(x) \le f(x)$ for any index set I. Larger sets will give better approximations.

The same can be said about g, leading to the polyhedral approximation

$$\overline{g}(x) = \max_{j\in J} g(x_j) + \xi_j^g (x - x_j)\,,$$

where ξ_j^g are subgradients of g at a collection of points x_j indexed by J. Thus, [NDP] can be approximated by,

$$\begin{cases} \min & \left\{ \max\limits_{i\in I} f(x_i) + \xi_i^f (x - x_i) \right\} \\ \text{s.t.} & \max\limits_{j\in J} g(x_i) + \xi_j^g (x - x_j) \le 0\,, \end{cases}$$

which is equivalent to,

$$\begin{cases} \min & v \\ \text{s.t.} & f(x_i) + \xi_i^f (x - x_i) \le v, \quad \forall i \in I, \\ & g(x_j) + \xi_j^g (x - x_j) \le 0, \quad \forall j \in J. \end{cases} \tag{3}$$

Problem (3) is a linear program that is easier to deal with than the original problem. It is to note, however, that this is only an approximation of [NDP], that gets better as more constraints are added. Let us denote by [MP_k] the *relaxed master problem* (3) with index sets I_k and J_k whose optimal solution is denoted by (x_k, v_k).

By transforming [NDP] into (3), nondifferentiability is eliminated at the cost of having a problem with a very large number of constraints. To get around that, cutting plane methods use only a subset of these constraints and generate the rest as needed. In fact, they would handle a series of relaxed master problems [MP_k] and stop when an optimal (satisfactory) solution to [NDP] is reached. [MP_k] is a relaxation of [NDP] as, by convexity of g,

$$\left\{ x\colon \max_{j\in J_k} \left\{ g(x_j) + \xi_j^g (x - x_j) \right\} \le 0 \right\}$$

contains $\{x\colon g(x) \le 0\}$, while, by convexity of f,

$$\max_{i\in I_k} \left\{ f(x_i) + \xi_i^f (x - x_i) \right\} \le f(x)$$

for all $x \in \{x\colon g(x) \le 0\}$. This implies that the optimal value v_k is a lower bound on the optimal value $f(x^*)$. The relaxation gets tighter as more points are included in the index sets I and J. The hope is that termination occurs before the sets I and J get enormously big.

The idea described above can be put into the generic cutting plane algorithm below.

In the course of the cutting plane algorithm, upper bounds are usually achieved through the objective evaluation at a feasible point. Lower bounds are either given by the optimal solution of the relaxed master problem, or by evaluating the dual objective at a feasible dual solution. For the stopping criteria, the algorithm may be stopped if the bound on the duality gap drops below a certain threshold. This is given by the difference between the best known upper and lower bounds.

Initialize:

1 Get an initial upper bound v_u and a lower bound v_l for the optimal solution $f(x^*)$.
2 Get an initial relaxed master problem [MP_0].

Iterate (k)

1 Get a *query point* x_k and a corresponding lower bound v_l^+.
2 If x_k is infeasible to [NDP] ($g(x_k) > 0$), then the oracle of g generates a *feasibility cut* of type $g(x_k) + \xi_j^g (x - x_k) \le 0$.
3 If x_k is feasible to [NDP] ($g(x_k) \le 0$), then the oracle of f generates an *optimality cut* of type $f(x_k) + \xi_i^f (x - x_k) \le v$ and an upper bound v_u^+.
4 Update the bounds:
a) if x_k is feasible to [NDP], then $v_u = \max\{v_u, v_u^+\}$.
b) $v_l = \min\{v_l, v_l^+\}$
5 Update index sets:
a) If a feasibility cut is added then $I_{k+1} := I_k$ and $J_{k+1} := J_k \cap \{k\}$.
b) If an optimality cut is added then $I_{k+1} := I_k \cap \{k\}$ and $J_{k+1} := J_k \cap \{k\}$.
6 Either STOP or $k := k + 1$.

At each iteration of the method, we can construct a bounded polyhedral set, namely the *localization set*. Given an upper bound v_u^k to [MP_k], it is given by

$$F_k(v_u^k) = \left\{ (x, v)\colon \begin{array}{c} v \le v_u^k; \\ f(x_i) + \xi_i^f (x - x_i) \le v, \\ i \in I_k; \\ g(x_j) + \xi_j^g (x - x_j) \le 0, \\ j \in J_k \end{array} \right\}.$$

The localization set $F_k(v_u)$ is a bounded polyhedral subset of the feasible region of [MP_k] that contains any optimal solution $(x^*, f(x^*))$ to [NDP]. To see that, $f(x^*)$

$\leq v_u^k$ as v_u^k is an upper bound on the optimal value. Furthermore, by the feasibility of x^* and the subgradient inequality $g(x_j) + \xi_j^g(x^* - x_j) \leq g(x^*) \leq 0$ for all $j \in J_k$ and by the fact that $[\mathrm{MP}_k]$ is a relaxation of [NDP], $f(x_i) + \xi_i^f(x^* - x_i) \leq v_k \leq f(x^*)$ for all $i \in I_k$.

Although every cutting plane method follows the above general scheme, different query point choices produce different algorithms. Prior to discussing a few of them, we would like to stress that some methods are explicitly based on the localization set, and require that this set be bounded. This might not hold in particular in the initialization phase, since no cut is yet present, and no upper and lower bounds on the objective are known. Therefore, one has to introduce box constraints. This may not even suffice if the first generated cut is not an optimality one, because the localization set remains unbounded in the z variable. One must then proceed with a an auxiliary phase I problem.

Kelley's Cutting Plane Method

This is the classical cutting plane method that was originally described in [27] and is present in the original *Dantzig–Wolfe decomposition* [7,8] and Benders decomposition [4]. At iteration k, the method solves the relaxed master $[\mathrm{MP}_k]$ and uses its solution x_k to generate further cuts.

As $[\mathrm{MP}_k]$ is a relaxation of [NDP], v_k gives a lower bound to $f(x^*)$ which is monotonically increasing. In addition, when x_k is feasible, $f(x_k)$ gives an upper bound that, unfortunately, is not monotonically decreasing. The difference between the updated bounds gives an estimate of the duality gap and can be used as a practical stopping criteria. The exact optimal solution of [NDP] is detected when x_k is feasible and $f(x_k) = v_k$, which is equivalent to having $0 \in \partial f(x_k)$.

This optimal point strategy assumes that the relaxation is a good approximation of the original problem, but this is only true when a big number of cuts has been generated. The method is globally convergent [47], but in practice, it sometimes shows a slow pattern of convergence [35].

Center of Gravity Method

This method was first proposed in [32] as the first cutting plane method that generates query points at the center of the localization set. Choosing the *center of gravity* to generate query points seems to be the natural choice. In fact cutting through the center of gravity of the a localization set of volume V and dimension n produces two sets, each with a volume of at least

$$\left[1 - (1 - \frac{1}{n+1})^n\right] V.$$

Therefore, after k iterations the volume of the localization set shrinks at a constant rate of $1/(1 - \exp(1))$. This rate of convergence is the best that can be obtained [46]. Unfortunately, finding a single center of gravity could be as difficult as solving the original problem.

For some simple convex bodies such as ellipsoids, cubes or spheres, calculating the center of gravity is relatively easy. This idea is the motivation behind the largest inscribed sphere method of [11], the volumetric method of [43] and the analytic center cutting plane method (ACCPM) of [18,19,21].

Largest Inscribed Sphere Method

The query point of this method is chosen as the center of the largest inscribed sphere in the localization set. Its calculation is based on work of G.L. Nemhauser and W.B. Widhelm [34], who showed that the minimization of the simple linear program

$$\begin{cases} \min & \sigma \\ \text{s.t.} & a_i^\top x + \left\| a_i^\top \right\| \sigma \leq b_i, \quad i \in I, \end{cases}$$

gives the radius σ and the center x of the largest inscribed sphere in the bounded polyhedron $\{x\colon a_i^\top x \leq b_i, i \in I\}$. The method is detailed in [11].

Volumetric Method

P.M. Vaidya [43] proposed the volumetric center as a query point. It is the maximizer of the determinant of the Hessian matrix of the logarithmic barrier function. This choice is motivated by the observation that for every point of the localization set, an inscribed ellipsoid can be constructed. The point that gives the maximum-volume inscribed ellipsoid is the minimizer of

$$\frac{1}{2} \log \left(\det \left(\sum \frac{a_i a_i^\top}{(b_i - a_i^\top x)^2} \right) \right),$$

where a_i are the columns of the matrix defining the localization set.

Bundle Methods

These methods were first proposed by C. Lemaréchal [30]. The method has developed over the years, [28,31], building upon the pioneering work [30].

The method adds a regularization term to the estimation of f and solves

$$\begin{cases} \min & \left\{ \max\limits_{i \in I_k} f(x_i) + \xi_i^f (x - x_i) \right. \\ & \qquad \left. + \frac{1}{2\alpha_k} \|x - x_{k-1}\|^2 \right\} \\ \text{s.t.} & \max\limits_{j \in J_k} \left\{ g(x_j) + \xi_j^g (x - x_j) \right\} \le 0, \end{cases} \tag{4}$$

to get the next query point x_k. The main idea is based on the fact that (2) is a good approximation of f when the next iterate is not far from the previous ones. Problem (4) is equivalent to the quadratic program

$$\begin{cases} \min & v + \frac{1}{2\alpha_k} \|x - x_{k-1}\|^2 \\ \text{s.t.} & f(x_i) + \xi_i^f (x - x_i) \le v, \quad i \in I_k, \\ & g(x_j) + \xi_j^g (x - x_j) \le 0, \quad j \in J_k. \end{cases}$$

The nice feature of bundle methods is that they limit the number of hyperplanes that are used to approximate f. The set of subgradients (the bundle) is updated at each iteration and kept moderately small. As a result, (4) is solved very quickly.

Techniques to estimate α_k are treated in [28] where three different values are proposed.

Analytic Center Cutting Plane Method (ACCPM)

To overcome the difficulty associated with the calculation of centers of polyhedrons and still use a central point strategy, ACCPM uses concepts form the interior point literature that have proven to be highly efficient.

The query point for ACCPM is the *analytic center* of the localization set $F_k(v_u)$. This new notion of center was first introduced in [40,41,42] as the the unique pair (x_a^k, v_a^k) that minimizes

$$\log(v_u^k - v) + \sum_{i \in I_k} \log[v - f(x_i) - \xi_i^f (x - x_i)] + \sum_{i \in J_k} \log[-g(x_i) - \xi_i^g (x - x_i)].$$

This function is a *potential function* similar to the one used by N.K. Karmarkar [26] when presenting the first interior point algorithm. Its minimization is equivalent to the solution of a linear program by any interior point method. The primal projective algorithm (cf. also ► Linear programming: Karmarkar projective algorithm) is favored due to its ability to deal with dual infeasibilities when new cuts are added. It is, in principle, a modified Newton method applied to the minimization of the potential function [16].

The *Newton procedure* for computing a central point (x_a^k, v_a^k) of $F_k(v_u)$ identifies, upon termination, a dual feasible solution to [MP$_k$]. Evaluating the dual objective at this point, gives a lower bound to the optimal solution of [NDP]. In addition, if x_a^k is feasible to [NDP], i. e. $g(x_a^k) \le 0$, then $f(x_a^k)$ is an upper bound. Unfortunately, x_a^k never coincides with the original optimizer x^*, as it can never be a vertex, so the stopping criteria can only be based on the difference between the upper a lower bounds. The algorithm would stop if that gap is sufficiently small.

The convergence analysis of the method is done in [20] and [36]. ACCPM is readily implemented [25] and has shown promising results in solving different large scale problems [2,3,17,33]. The method was modified to use weighted analytic centers as a query point [22], to add multiple cuts at once [24], to use a primal-dual interior point algorithm for the calculation of analytic centers [9] and to accommodate quadratic cuts [10].

Concluding Remarks

Cutting plane methods are an effective solution approach for nondifferentiable optimization. This has proven to be true when advanced methods such as bundle and analytic center methods were designed. Not only they make use of recent advances in optimization theory, they also resulted in efficient computer implementations.

Manipulating the set of cuts is an important enhancement factor to cutting plane methods. In the addition of cuts, introducing a whole set at once is more effective than introducing single cuts as it allows the fast accumulation of information about the problem. This is possible for certain classes of problems where disaggregation is possible. This is true for Dantzig–Wolfe and

Benders decompositions with primal and dual block diagonal matrices respectively.

The second improvement is in the manipulation of number of cuts in the relaxed master problem. Keeping all cuts may lead to a better approximation but it also requires huge amounts of storage space and solution time. Thus well defined cut-dropping strategies will considerably improve the performance of the method.

A final remark concerns the use of heuristics with cutting planes. They can be used to initialize the method so that sufficient cutting planes are obtained to start the method, or take over the search for optimal solutions when the method stalls.

See also

- ► Dini and Hadamard Derivatives in Optimization
- ► Global Optimization: Envelope Representation
- ► Nondifferentiable Optimization
- ► Nondifferentiable Optimization: Minimax Problems
- ► Nondifferentiable Optimization: Newton Method
- ► Nondifferentiable Optimization: Parametric Programming
- ► Nondifferentiable Optimization: Relaxation Methods
- ► Nondifferentiable Optimization: Subgradient Optimization Methods

References

1. Atkinson DS, Vaidya PM (1995) A cutting plane algorithm that uses analytic centers. Math Program B 69:1–43
2. Bahn O, Goffin JL, Vial JP, du Merle O (1994) Implementation and behavior of an interior point cutting plane algorithm for convex programming: An application to geometric programming. Discrete Appl Math 49:3–23
3. Bahn O, Merle OD, Goffin JL, Vial JP (1995) A cutting plane method from analytic centers for stochastic programming. Math Program B 69:45–73
4. Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Math 4:238–252
5. Cheney W, Goldstein AA (1959) Newton's method for convex programming and Chebyshev approximation. Numerische Math 1–5:253–268
6. Clarke FH (1989) Optimization and nonsmooth analysis. Les Publ. CRM, Montreal
7. Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. Oper Res 8:101–111
8. Dantzig GB, Wolfe P (1961) The decomposition algorithm for linear programming. Econometrica 29(4):767–778
9. Denault M, Goffin J-L (1997) On a primal-dual analytic center cutting plane method for variational inequalities. GERAD Techn Report G-97-56
10. Denault M, Goffin J-L (1998) Variational inequalities with quadratic cuts. GERAD Techn Report G-98-69
11. Elzinga J, Moore TG (1975) A central cutting plane algorithm for the convex programming problem. Math Program 8:134–145
12. Fiacco AV, McCormick GP (1968) Nonlinear programming: Sequential unconstrained minimization techniques. Wiley, New York. Reprint: SIAM Classics Appl Math, vol 4, 1990
13. Fisher ML (1981) The Lagrangian relaxation method for solving integer programming problems. Managem Sci 27:1–18
14. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Th Appl 10:237–260
15. Geoffrion AM (1974) Lagrangean relaxation for integer programming. Math Program Stud 2:82–114
16. de Ghellinck G, Vial JP (1986) A polynomial Newton method for linear programming. Algorithmica 1:425–453
17. Goffin J-L, Gondzio J, Sarkissian R, Vial J-P (1997) Solving nonlinear multicommodity flow problem by the analytic centre cutting plane method. Math Program B 76:131–154
18. Goffin J-L, Haurie A, Vial J-P (1992) Decomposition and nondifferentiable optimization with the projective algorithm. Managem Sci 38(2):284–302
19. Goffin J-L, Haurie A, Vial J-P, Zhu DL (1993) Using central prices in the decomposition of linear programs. Europ J Oper Res 64:393–409
20. Goffin J-L, Luo Z-Q, Ye Y (1996) Complexity analysis of an interior cutting plane method for convex feasibility problems. SIAM J Optim 6:638–652
21. Goffin J-L, Vial J-P (1990) Cutting planes and column generation techniques with the projective algorithm. J Optim Th Appl 65:409–429
22. Goffin J-L, Vial J-P (1993) On the computation of weighted analytic centers and dual ellipsoids with the projective algorithm. Math Program 60:81–92
23. Goffin J-L, Vial J-P (1998) Interior point methods for nondifferentiable optimization. In: Kishka P, Lorenz HW, Derigs U, Domschke W, Kleinschmidt P, Moehring R (eds) Operations Research Proc. 1997. Springer, Berlin, pp 35–49
24. Goffin J-L, Vial J-P (1998) Multiple cuts in the analytic center cutting plane method. HEC/Logilab Techn Report 98.10 and G-98-26, Dept. Management Stud Univ Geneva
25. Gondzio J, du Merle O, Sarkissian R, Vial JP (1994) ACCPM-A library for convex optimization based on analytic centre cutting plane method. Europ J Oper Res 94:206–211
26. Karmarkar NK (1984) A new polynomial-time algorithm for linear programming. Combinatorica 4:373–395
27. Kelley JE (1960) The cutting plane method for solving convex programs. J SIAM 8:703–712
28. Kiwiel KC (1990) Proximity control in Bundle methods for convex nondifferentiable optimization. Math Program 46:105–122

29. Lasdon L (1970) Optimization theory for large scale systems. MacMillan, New York
30. Lemaréchal C (1978) Bundle methods in nonsmooth optimization. In: Lemarechal C, Mifflin R (eds) Nonsmooth Optimization, Proc. IIASA Workshop, March 28–April 8 1977. Pergamon, Oxford
31. Lemaréchal C, Nemrirovskii A, Nesterov Y (1995) New variants of bundle methods. Nonsmooth Optim, Math Program 69:111–147
32. Levin AY (1965) On an algorithm for the minimization of convex functions over convex functions. Soviet Math Dokl 6:286–290
33. du Merle O, Hansen P, Jaumard B, Mladenovic N (1997) An interior point algorithm for minimum sum of squares clustering. GERAD Techn Report G97-53
34. Nemhauser GL, Widhelm WB (1971) A modified linear program for columnar methods in mathematical programming. Oper Res 19:1051–1060
35. Nemirovskii AS, Yudin DB (1983) Problem complexity and method efficiency in optimization. Wiley, New York
36. Nesterov Y (1996) Cutting plane methods from analytic centers: efficiency estimates. Math Program B 69:149–176
37. Roos C, Terlaky T, Vial J-P (1997) Interior point methods: Theory and algorithms. Springer, Berlin
38. Shor NZ (1978) Subgradient methods: A survey of Soviet research. In: Lemaréchal C, Mifflin R (eds) Nonsmooth optimization: Proc. IIASA workshop, March 28–April 8 1977. Pergamon, Oxford
39. Shor NZ (1985) Minimization methods for nondifferentiable functions. Springer, Berlin
40. Sonnevend G (1985) An analytical center for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming. Lecture Notes Control Inform Sci, vol 84. Springer, Berlin, pp 866–876
41. Sonnevend G (1988) New algorithms in convex programming based on a notion of centre (for systems of analytic inequalities) and on rational extrapolation. In: Hoffmann KH, Hiriat-Urruty JB, Lemaréchal C, Zowe J (eds) Trends in Mathematical Optimization; Proc. 4th French–German Conf. Optimization, Irsee, April 1986. Internat Ser Numer Math. Birkhäuser, Basel, pp 311–327
42. Sonnevend G (1989) Applications of the notion of analytic center in approximation (estimation) problem. J Comput Appl Math 28:349–358
43. Vaidya P (1996) A new algorithm for minimizing convex functions over convex sets. Math Program 73:291–341
44. Wolfe P (1975) A method of conjugate subgradients for minimizing nondifferentiable functions. Math Program Stud 3:145–173
45. Ye Y (1992) A potential reduction algorithm allowing column generation. SIAM J Optim 2:7–20
46. Ye Y (1997) Interior point algorithms: Theory and analysis. Wiley, New York
47. Zangwill WI (1969) Nonlinear programming: A unified approach. Prentice-Hall, Englewood Cliffs

Nondifferentiable Optimization: Minimax Problems

VLADIMIR F. DEMYANOV
St. Petersburg State University, St. Petersburg, Russia

MSC2000: 90C30, 65K05

Article Outline

Keywords

Minimax problem; Nondifferentiable optimization; Inf-stationary point; Extremal basis method; Hypodifferential; Subdifferential

Max-Type Functions

Let $S \subset \mathbf{R}^n$ be an open set. A standard *minimax problem* (MMP) is the problem of minimizing a function

$$f(x) = \max_{y \in G} \varphi(x, y), \tag{1}$$

where G is a compact set of some space Y, the function $\varphi: S \times G \to \mathbf{R}$ is continuous jointly in $[x, y]$ on $S \times G$ and continuously differentiable in x (the function $\varphi'_x(x, y)$ is continuous in $[x, y]$ on $S \times G$). The function f is, in general, nonsmooth though φ is smooth in x.

Remark 1 The function

$$f(x) = \max_{y \in G} |\varphi(x, y)|$$

can be rewritten in the form (1) since

$$\begin{aligned} f(x) &= \max_{y \in G} \max\{\varphi(x, y), -\varphi(x, y)\} \\ &= \max_{\overline{y} \in \overline{G}} \overline{\varphi}(x, \overline{y}), \end{aligned}$$

where

$$\overline{G} = \{[1, y]\colon y \in G\} \bigcup \{[-1, y]\colon y \in G\} \subset \mathbf{R} \times Y = \overline{Y},$$

$$\overline{\varphi}(x, \overline{y}) = \begin{cases} \varphi(x, y), & \overline{y} = [1, y], \\ -\varphi(x, y), & \overline{y} = -[1, y]. \end{cases}$$

The first serious study of minimax problems was performed by P.L. Chebyshev who may be considered as Godfather of nonsmooth analysis. He treated the problem of minimizing the function

$$f(x) = \max_{y \in [a,b]} |\Psi(y) - P(x, y)|, \tag{2}$$

where $x = (x_0, \ldots, x_{n-1}) \in \mathbf{R}^n$, $P(x, y) = \sum_{i=0}^{n-1} x_i y^i$. The function $P(x, y)$ is a polynomial (in y). If $x^* \in \mathbf{R}^n$ and

$$f(x^*) = \min_{x \in \mathbf{R}^n} f(x)$$

then $P(x^*, y)$ is called the *polynomial of best approximation* (for the function $\Psi(y)$). If $\Psi(y) = y^n$, $[a, b] = [-1, 1]$ then

$$P_n^*(y) = y^n - P(x^*, y) = \frac{1}{2^{n-1}} T_n(y),$$

where $T_n(y)$ is the famous *Chebyshev polynomial*:

$$T_0(y) = 1, \quad T_1(y) = y,$$
$$T_{m+1}(y) = 2yT_m(y) - T_{m-1}(y), \quad \forall m \geq 2.$$

On the interval $[-1, 1]$ the relation $T_n(y) = \cos(n \arccos y)$ holds.

Let $\varphi_1(x, y) = \Psi(y) - P(x, y)$. The following condition holds (see [4,9]): For a point $x^* \in \mathbf{R}^n$ to be a minimizer of f (see (2)) it is necessary and sufficient that $n + 1$ points $y_0, \ldots, y_n$ exist such that

$$y_i \in G, \quad \forall i \in 0, \ldots, n; \quad y_0 < \cdots < y_n,$$

$$\varphi_1(x^*, y_{i+1}) = -\varphi_1(x^*, y_i), \tag{3}$$

$$\forall i \in 0, \ldots, n-1, \tag{4}$$

$$|\varphi_1(x^*, y_{i+1})| = \varphi(x^*, y_i) = f(x^*). \tag{5}$$

The set $\{y_0, \ldots, y_n\}$ satisfying (3)–(5) is called a *Chebyshev alternation* (or *alternance*).

The following properties of a max-function will be used in the sequel (see [1,4,5]).

1) Let a function f be described by (1). Then it is directionally differentiable at any point $x \in S$ and

$$f'(x, g) = \lim_{\alpha \downarrow 0} \frac{f(x + \alpha g) - f(x)}{\alpha} = \max_{v \in \partial f(x)} (v, g), \tag{6}$$

where

$$\partial f(x) = \operatorname{co} \{\varphi'(x, y)\colon y \in R(x)\}, \quad g \in \mathbf{R}^n,$$
$$R(x) = \{y \in G\colon \varphi(x, y) = f(x)\}.$$

(6) means that f is subdifferentiable, $\partial f(x)$ is the *subdifferential* of f at x (it is a convex compact set). The subdifferential mapping ∂f is not, in general, Hausdorff continuous.

2) For a point $x^* \in S$ to be a (local or global) minimizer of f on S it is necessary that

$$0_n \in \partial f(x^*). \tag{7}$$

Here $0_n = (0, \ldots, 0) \in \mathbf{R}^n$.

A point $x^* \in S$ satisfying (7) is called an *inf-stationary point* of f.

Remark 2 Note that for a point $x^{**} \in S$ to be a (local or global) maximizer of f on S it is necessary that

$$\partial f(x^{**}) = \{0_n\}. \tag{8}$$

If the function f is convex (it is the case, e. g., if φ is convex in x for any $y \in G$), then condition (7) is also sufficient for x^* to be a global minimizer of f on an open set S. Therefore condition (8) implies that a convex function f does not attain its maximal value on an open convex set S.

3) If x_0 is not inf-stationary then the direction

$$g(x_0) = -\frac{v_0}{\|v_0\|}, \tag{9}$$

where

$$\|v_0\| = \max_{v \in \partial f(x_0)} \|v\|,$$

is the steepest descent direction of f at x_0, i. e.

$$f'(x_0, g(x_0)) = \min_{\|g\|=1} f'(x_0, g).$$

4) The following expansion holds

$$f(x+\Delta) = f(x) + \max_{y\in G}[\varphi(x,y) - f(x) + (\varphi'_x(x,y),\Delta)] + o_x(\Delta),$$

where

$$\frac{o_x(\Delta)}{\|\Delta\|} \overset{\|\Delta\|\to 0}{\longrightarrow} 0.$$

This expression can be rewritten as

$$f(x+\Delta) = f(x) + \max_{[a,v]\in \mathrm{d}f(x)} [a + (v,\Delta)] + o_x(\Delta), \tag{10}$$

where

$$\mathrm{d}f(x) = \mathrm{co}\left\{[a,v]\in \mathbf{R}\times\mathbf{R}^n : \begin{array}{c} a=\varphi(x,y)-f(x),\\ v=\varphi'_x(x,y),\\ y\in G\end{array}\right\} \tag{11}$$

is the *hypodifferential* of f at x. Note that $a \geq 0$ and $\max_{[a,v]\in df(x)} a = 0$. The mapping df is Hausdorff-continuous on S.

5) Necessary condition (7) is equivalent to

$$0_{n+1} \in \mathrm{d}f(x^*). \tag{12}$$

6) If x_0 is not inf-stationary, then let us find

$$\min_{z\in \mathrm{d}f(x^*)} \|z\| = \|z_0\|,$$

where $z_0 = [a_0, v_0]$. In this case $v_0 \neq 0_n$. The direction

$$\widetilde{g}(x_0) = -\frac{v_0}{\|v_0\|} \tag{13}$$

is a descent (not necessary steepest descent) direction of f at x_0.

The vector-function $\widetilde{g}(x)$ (see (13)) is continuous.

7) If the function φ in (1) is twice continuously differentiable at x then the following relation holds:

$$f(x+\Delta) = f(x) + \max_{y\in G}\left[\varphi(x,y) - f(x) + (\varphi'_x(x,y),\Delta) + \frac{1}{2}(\varphi_{xx}(x,y)\Delta,\Delta)\right] + o_x(\Delta^2), \tag{14}$$

where

$$\frac{o_x(\Delta^2)}{\|\|\Delta\|^2} \overset{\|\Delta\|\to 0}{\longrightarrow} 0.$$

(14) can be rewritten as

$$f(x+\Delta) = f(x) + \max_{[a,v,A]\in d^2 f(x)}\left[a + (v,\Delta) + \frac{1}{2}(A\Delta,\Delta)\right] + o_x(\Delta^2), \tag{15}$$

where

$$d^2 f(x) = \mathrm{co}\left\{[a,v,A]\in \mathbf{R}\times\mathbf{R}^n\times\mathbf{R}^{n\times n} : \begin{array}{c} a=\varphi(x,y)-f(x),\\ v=\varphi'_x(x,y),\\ A=\varphi_x(x,y),\\ y\in G\end{array}\right\}. \tag{16}$$

Here $\mathbf{R}^{n\times n}$ is the space of real $(n\times n)$-matrices. The set $d^2 f(x)$ is called the *second hypodifferential* of f at x. It is closed, bounded and convex. The mapping $d^2 f$ is Hausdorff continuous on S. In this case f is twice continuously hypodifferentiable.

Algorithms for Unconstrained Minimization

Assume that in (1) $S = \mathbf{R}^n$. Then the problem of minimizing f is an unconstrained minimax problem. There are a lot of numerical methods to solve this problem based on the properties of max-functions.

Method of Steepest Descent

Let $x_0 \in \mathbf{R}^n$ be arbitrary. Assume that x_k has already been defined. If $0_n \in \partial f(x_k)$ then x_k is an inf-stationary point and the process terminates. If $0_n \notin \partial f(x_k)$ then let us take $g_k = g(x_k)$ (see (9)) and find

$$\min_{\alpha\geq 0} f(x_k + \alpha g_k) = f(x_k + \alpha_k g_k). \tag{17}$$

Now, put $x_{k+1} = x_k + \alpha_k g_k$. Continuing in the same manner we construct a sequence $\{x_k\}$ such that

$$f(x_{k+1}) < f(x_k). \tag{18}$$

If this sequence $\{x_k\}$ is finite, then, by construction, the last point is a stationary one.

Let $\{x_k\}$ be not finite. Assume that the set $Q(x_0) = \{x \in \mathbf{R}^n: f(x) \le f(x_0)\}$ is bounded (then it is closed). Due to (18) $x_k \in Q(x_0)$ and, hence, there exist a point $x^* \in Q(x_0)$ and a subsequence $\{x_{k_s}\}$ such that $x_{k_s} \to x^*$. One may expect that x^* is inf-stationary. However, in general, this is not the case and the reason is the discontinuity of the mapping ∂f.

To ensure the convergence it is necessary to overcome the discontinuity of ∂f.

Let us introduce the set $R_\varepsilon(x) = \{y \in G: f(x) - \varphi(x, y) \le \varepsilon\}$ where $\varepsilon \ge 0$. Put

$$L_\varepsilon(x) = \operatorname{co}\left\{\varphi'_x(x, y): y \in R_\varepsilon(x)\right\}.$$

Find

$$\min_{v \in L_\varepsilon(x)} \|v\| = \|v_\varepsilon(x)\|.$$

If $\|v_\epsilon(x)\| = 0$, the point x is called an *ε-stationary point*. Choose any $\varepsilon > 0$. Let us construct the following method. Take any $x_0 \in \mathbf{R}^n$. Let x_k have already been found. If $\|v_\varepsilon(x_k)\| = 0$ then x_k is ε-stationary. If $\|v_\varepsilon(x_k)\| > 0$ then let us find

$$\min_{\alpha \ge 0} f(x_k - \alpha v_\varepsilon(x_k)) = f(x_k - \alpha_k v_\varepsilon(x_k))$$

and put $x_{k+1} = x_k - \alpha_k v_\varepsilon(x_k)$. Continuing analogously we get a sequence $\{x_k\}$.

Proposition 3 *If the set $Q(x_0)$ is bounded then in a finite number of steps we arrive at a point x_k such that*

$$0_n \in L_{2\varepsilon}(x_k).$$

Thus, in a finite number of steps we shall find a 2ε-stationary point.

Now it is not difficult to modify this method to get an inf-stationary point of f.

Choose any $\varepsilon_0 > 0$ and $x_0 \in \mathbf{R}^n$. Assume that $Q(x_0)$ is bounded. Applying the above method, in a finite number of steps we shall find a point $\overline{x}_0$ such that $0_n \in L_{2\varepsilon_0}(\overline{x}_0)$. Let $\overline{x}_k \in Q(x_0)$ be found such that $0_n \in L_{2\varepsilon_k}(\overline{x}_k)$ where $\varepsilon_k = 2^{-k}\varepsilon_0$. Take $\varepsilon_{k+1} = \varepsilon_k/2$ and $x_k = \overline{x}_k$. Applying the above method, in a finite number of steps a point $\overline{x}_{k+1} \in Q(x_0)$ will be found such that $0_n \in L_{2\varepsilon_{k+1}}(\overline{x}_{k+1})$. Clearly, $\overline{x}_{k+1} \in Q(x_0)$. The sequence $\{\overline{x}_k\}$ is bounded.

Proposition 4 *Any limit point of the sequence $\{x_k\}$ is an inf-stationary point of f.*

Hypodifferential Descent

Another method is based on expansion (10). Take $x_0 \in \mathbf{R}^n$. Let x_k have been found. Compute

$$\min_{z \in df(x_k)} \|z\| = \|z_k\|,$$

where $z_k = [a_k, v_k]$. If $\| z_k \| = 0$ then the point x_k is inf-stationary and the process terminates.

If $\| z_k \| > 0$ then $v_k \ne 0_n$ and let us find

$$\min_{\alpha \ge 0} f(x_k - \alpha v_k) = f(x_k - \alpha_k v_k)$$

and put $x_{k+1} = x_k - \alpha_k v_k$.

If the sequence $\{x_k\}$ thus constructed is finite then the last point is inf-stationary. If $\{x_k\}$ is infinite then the following result holds:

Proposition 5 *If the set $Q(x_0)$ is bounded then any limit point of the sequence $\{x_k\}$ is inf-stationary.*

Remark 6 The two algorithms described above are 'conceptual' (according to the terminology of E. Polak). These algorithms are computationally effective in the case where the set G (see (1)) contains only a finite number of points. Different practical implementations of the above ideas can be found in [2,4,10,11,12].

Newton-Type Method

If the function φ in (1) is twice continuously differentiable, one can employ the expansion (14)–(15).

Take any $x_0 \in \mathbf{R}^n$. Let x_k have already been defined. Find

$$\min_{\Delta \in \mathbf{R}^n} F_k(\Delta) = F_k(\Delta_k),$$

where

$$F_k(\Delta) = \max_{[a,v,A] \in d^2 f(x_k)} \left[a + (v, \Delta) + \frac{1}{2}(A\Delta, \Delta)\right].$$

Now put $x_{k+1} = x_k + \Delta_k$.

Under some additional conditions (see [3]) the sequence $\{x_k\}$ thus constructed converges at least to a local minimizer of f (and the rate of convergence is quadratic).

A Convex Max-Function

Extremal Basis Method

Now let us consider the case where f is described by (1) and the function φ in (1) is strongly convex in x with

a constant $m > 0$ for every fixed $y \in G$, i. e.

$$\varphi(x + \Delta, y) \geq \varphi(x, y) + (\varphi_x'(x, y), \Delta) + m\|\Delta\|^2, \quad \forall [x, y] \in \mathbf{R}^n \times G, \quad \forall \Delta \in \mathbf{R}^n. \tag{19}$$

The function f is also strongly convex and has a unique minimizer.

Choose an arbitrary set of $n + 2$ points from G:

$$\tau_0 = \{y_{01}, \ldots, y_{0,n+2}\}, \quad y_{0i} \in G, \quad \forall i \in 1, \ldots, n + 2.$$

The set τ_0 will be called a *basis*.

Assume that the points $x_0, \ldots, x_{k-1}$ and the basis $\tau_k = \{y_{k1}, \ldots, y_{k,n+2}\}$ have already been found.

Let us define the function

$$f_k(x) = \max_{i \in 1, \ldots, n+2} \varphi(x, y_{ki})$$

and choose a point $x_k \in \mathbf{R}^n$ such that

$$f_k(x_k) = \min_{x \in \mathbf{R}^n} f_k(x). \tag{20}$$

If $f(x_k) = f_k(x_k)$ then x_k is the minimizer of f, and the process terminates.

The minimization problem in (20) is simpler than that of minimizing f.

Consider the case $f(x_k) > f_k(x_k)$. By the necessary and (in our convex case) sufficient condition (7)

$$0_n \in L_k, \tag{21}$$

where $L_k = \operatorname{co} H_k$, $H_k = \{\varphi_x'(x_k, y_{ki}) : i \in R_k\}$, $R_k = \{i \in 1, \ldots, n + 2 : \varphi(x_k, y_{ki}) = f_k(x_k)\}$. By the Carathéodory theorem [7] every point of L_k can be represented as a convex combination of not more than $n + 1$ points of H_k. Therefore, there exists at least one index $i_k \in 1, \ldots, n+2$ such that either $i_k \notin R_k$ or the origin in (21) may be 'constructed' without the vector $\varphi_x'(x_k, y_{ki_k})$.

Let $\overline{y}_k \in G$ be such that

$$\varphi(x_k, \overline{y}_k) = f(x_k) = \max_{y \in G} \varphi(x_k, y).$$

Now let us construct a new basis

$$\tau_{k+1} = \{y_{k+1,1}, \ldots, y_{k+1,n+2}\}$$

where

$$y_{k+i,i} = \begin{cases} y_{ki}, & i \neq i_k, \\ \overline{y}_k, & i = i_k. \end{cases}$$

The basis τ_{k+1} differs from τ_k by one point and also contains $n + 2$ points. For the basis τ_{k+1} again define the function $f_{k+1}(x)$ and the point x_{k+1}.

As a result, a sequence $\{x_k\}$ is constructed. If this sequence is finite, its last point is the minimizer. If not, the following property holds.

Proposition 7 (See [6 Sect. III.10].) *The sequence $\{x_k\}$ converges to the minimum point of f.*

Remark 8 The extremal basis method can be extended (with necessary adjustments) to the case where φ is just convex at x, not necessarily strongly convex.

Remark 9 If the function φ in (1) is not convex then condition (7) and the Carathéodory theorem produce the following properties:

1) There exist points $y_1, \ldots, y_{k+1}$ such that $y_i \in Y$ for any $i \in 1, \ldots, k + 1$ and a minimizer x^* of f is an inf-stationary point of the function

$$F(x) = \max_{i \in 1, \ldots, k+1} \varphi(x, y_i).$$

2) There exist points $y_1, \ldots, y_{k+1}$ and coefficients $\alpha_1, \ldots, \alpha_{k+1}$ such that

$$y_i \in Y, \quad \alpha_i \geq 0, \quad \forall i \in 1, \ldots, k + 1, \quad \sum_{i \in 1, \ldots, k+1} \alpha_i = 1,$$

and a minimizer x^* is a stationary point of the smooth function

$$L(x) = \sum_{i \in 1, \ldots, k+1} \alpha_i \varphi(x, y_i).$$

These properties can be used to derive corresponding numerical algorithms.

Constrained Minimax Problems

Let f be defined by (1) on an open set $S \subset \mathbf{R}^n$ and $\Omega \subset S$ be a closed set. The problem is to find

$$\min_{x \in \Omega} f(x) = f^*.$$

Take $x \in \Omega$. The set

$$\Gamma(x, \Omega) = \left\{ g \in \mathbf{R}^n : \begin{array}{c} \exists \{[\alpha_k, g_k]\} : \\ [\alpha_k, g_k] \to [+0, g], \\ x + \alpha_k g_k \in \Omega, \\ \forall k \end{array} \right\}$$

is the *Bouligand cone* to Ω at x. It is nonempty and closed.

The function f is subdifferentiable (see (6)).

The following necessary condition holds ([4,5]):

For a point $x^* \in \Omega$ to be a minimizer of f on Ω it is necessary that

$$f'(x^*, g) \geq 0, \quad \forall g \in \Gamma(x^*, \Omega). \tag{22}$$

The cone $\Gamma(x^*, \Omega)$ may be represented in the form

$$\Gamma(x^*, \Omega) = \bigcup_{i \in I} A_i, \tag{23}$$

where the A_i are closed convex cones, I is some set (e. g., $\Gamma(x^*, \Omega)$ can always be given as the union of all its rays). Of course, from practical considerations we are interested to have as few elements in I as possible (the best case is the one where Ω is convex, then $\Gamma(x^*, \Omega)$ is also convex).

Taking into account (6), condition (22) can be rewritten in the equivalent form

$$\partial f(x^*) \cap A_i^+ \neq \emptyset, \quad \forall i \in I, \tag{24}$$

where A_i^+ is the cone conjugate to A_i:

$$A_i^+ = \{v \in = \mathbf{R}^n : \ (v, g) \geq 0, \ \forall g \in A_i\}.$$

A point $x^* \in \Omega$ satisfying (24) is called an *inf-stationary point* of f on Ω.

Let $x \in \Omega$ be not inf-stationary. For every $i \in I$ let us find

$$\min_{\substack{v \in \partial f(x) \\ w \in A_i^+}} \|v - w\| = \|v_i - w_i\| = \rho_i \tag{25}$$

and

$$\max_{i \in I} \rho_i = \rho_{i_0} = \|v_{i_0} - w_{i_0}\|. \tag{26}$$

Then the direction

$$g_{i_0}(x_0) = \frac{w_{i_0} - v_{i_0}}{\rho_{i_0}}$$

is a direction of steepest descent of the function f at the point x_0 (on the set Ω):

$$f'(x, g_{i_0}(x_0)) = \min_{\substack{g \in \Gamma(x_0, \Omega) \\ \|g\| = 1}} f'(x, g).$$

It may happen that there exist several steepest descent directions (s.d.d.). If $\Gamma(x_0, \Omega)$ is convex then such a direction is unique.

Many numerical methods for minimizing f on Ω employ condition (24) (see [2,6,10,11,12]).

Let us discuss in detail the case where Ω is described in the form

$$\Omega = \{x \in S : \ h(x) \leq 0\}, \tag{27}$$

where

$$h(x) = \max_{z \in G_1} \psi(x, z),$$

G_1 is a compact set of some space Z, $\psi : S \times G_1 \to \mathbf{R}^n$ is continuous jointly in x and z on $S \times G_1$ and is continuously differentiable in x. Assume that Ω is nonempty. Note that Ω is closed.

The function h is also subdifferentiable and

$$\partial h(x) = \operatorname{co}\{\psi'_x(x, z) : \ z \in Q(x)\},$$
$$Q(x) = \{z \in G_1 : \ \psi(x, z) = h(x)\}.$$

Note that if $h(x) = 0$ and $0_n \notin \partial h(x)$ then

$$\Gamma^+(x, \Omega) = \{v = \lambda w : \ \lambda \leq 0, \ w \in \partial h(x)\}.$$

If $h(x) < 0$ the $\Gamma^+(x, \Omega) = \{0_n\}$.

The cone $\Gamma(x, \Omega)$ is convex and closed and, hence, there exists only one steepest descent direction.

The following necessary condition is true: Let $x^* \in \Omega$, $h(x^*) = 0$. For a point x^* to be a minimizer of f on Ω it is necessary that

$$0_n \in \operatorname{co}\{\partial f(x^*), \partial h(x^*)\} = L(x^*). \tag{28}$$

If $0_n \notin \partial h(x^*)$ then conditions (22) and (28) are equivalent. If $0_n \in \partial h(x^*)$ then (28) holds automatically but (22) does not.

If $0_n \notin L(x_0)$, $h(x_0) = 0$ then the direction

$$g(x_0) = -\frac{v(x_0)}{\|v(x_0)\|}$$

where

$$\|v(x_0)\| = \min_{v \in L(x_0)} \|v\|$$

is a descent direction and it is an admissible direction, i. e. there exists $\alpha_0 > 0$ such that $x_0 + \alpha\, g(x_0) \in \Omega$, $\forall \alpha$

$\in [0, \alpha_0]$. Observe that the steepest descent direction is not necessarily admissible.

Note also that for any $\lambda > 0$ condition (28) is equivalent to

$$0_n \in \operatorname{co}\{\partial f(x^*), \lambda \partial h(x^*)\} = L_\lambda(x^*).$$

Method of Hypodifferential Descent

The function h (as well as f) is continuously hypodifferentiable with the hypodifferential (cf. (11))

$$dh(x) = \operatorname{co}\left\{ [a, v] : \begin{array}{c} a = \psi(x, z) - h(x), \\ v = \psi'_x(x, z), \\ z \in G_1 \end{array} \right\}. \quad (29)$$

Let

$$L(x) = \operatorname{co}\{df(x), dh(x) + [h(x), 0_n]\}.$$

Proposition 10 *For a point $x^* \in \Omega$ to be a minimizer of f on Ω it is necessary that*

$$0_{n+1} \in L(x^*). \quad (30)$$

A point $X^* \in \Omega$ satisfying (30) is an inf-stationary point of f on Ω.

If $x \in \Omega$ is not inf-stationary, then

$$\rho(x) = \min_{\overline{z} \in L(x)} \|z\| = \|\overline{z}(x)\| > 0.$$

Here

$$\overline{z}(x) = [\eta(x), z(x)],$$
$$\eta(x) \in \mathbf{R}, \quad z(x) \in \mathbf{R}^n, \quad z(x) \neq 0_n.$$

The direction $g(x) = -z(x)/\|z(x)\|$ is an admissible descent direction. The vector-function $z(x)$ is continuous on Ω.

Let us describe the following method (see [5, Chap. 5, Sect. 5]):

Take any $x_0 \in \Omega$. Let $x_k \in \Omega$ have already been defined. If $\rho(x_k) = 0$ then x_k is inf-stationary. If $\rho(x_k) > 0$ then let us find

$$\min_{\substack{\alpha \geq 0, \\ (x_k - \alpha z(x_k)) \in \Omega}} f(x_k - \alpha z(x_k)) = f(x_k - \alpha_k z(x_k)).$$

Now put

$$x_{k+1} = x_k - \alpha_k z(x_k).$$

By construction $x_k \in \Omega$, $\forall k$.

If the sequence $\{x_k\}$ is finite, its last point is an inf-stationary one. If it is infinite the following result holds:

Proposition 11 *If the set*

$$\{x \in \Omega : f(x) \leq f(x_0)\}$$

is bounded, then any limit point of the sequence $\{x_k\}$ is inf-stationary.

Remark 12 The method described is 'conceptual'. For its practical implementation it is necessary to avoid the computation of df and dh (by (11) and (29)) and take some smaller sets (since the hypodifferential mapping is not uniquely defined).

Remark 13 If both functions φ and ψ are convex in x then the extremal basis method (given above) can be extended for minimizing f on Ω (see [6]).

The Kelley Method

Let us consider the problem of minimizing a function

$$f(x) = \max_{y \in G} \varphi(x, y) = \max_{y \in G} [\varphi_1(x, y) + f_1(x)]$$

on a convex compact set $\Omega \in \mathbf{R}^n$, where $\varphi_1 : \Omega \times G \to \mathbf{R}$ is convex in x on Ω for any $y \in G$ and continuous in y on G and $f_1 : \Omega \mathbf{R}$ is continuous on Ω. Then the following modification of the *Kelley cutting plane method* [8] can be used:

Choose any $x_0 \in \Omega$ and find $y_0 \in G$ such that $\varphi(x_0, y_0) = f(x_0)$. Take any $v_0 \in \partial\varphi_1(x_0, y_0)$ (where $\partial\varphi_1(x_0, y_0)$ is the subdifferential of the convex function $\varphi_1(x, y_0)$ at x_0). Put

$$B_0(x) = f_1(x) + \varphi_1(x_0, y_0) + (v_0, x - x_0).$$

Let $x_k \in \Omega$ have already been defined. Find $y_k \in G$ such that $\varphi(x_k, y_k) = f(x_k)$, take any $v_k \in \partial\varphi_1(x_k, y_k)$ and put

$$B_k(x) = f_1(x) + \varphi_1(x_k, y_k) + (v_k, x - x_k).$$

Let

$$f_k(x) = \max_{i\in 0,\dots,k} B_i(x)$$
$$= f_1(x) + \max_{i\in 0,\dots,k} [\varphi_1(x_i, y_i) + (v_i, x - x_i)].$$

Find

$$\min_{x\in\Omega} f_k(x) = f_k(x_k^*).$$

If $f_k(x_k^*) = f(x_k^*)$ then x_k^* is a minimizer of f on Ω and the process terminates. Otherwise, take $x_k^* = x_{k+1}$ and proceed as above. If the sequence $\{x_k\}$ thus constructed is finite, its last point is a minimizer. If not, the following statement holds.

Proposition 14 *Any limit point of the sequence $\{x_k\}$ is a minimizer of f on Ω.*

See also

- ▶ Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs
- ▶ Bilevel Optimization: Feasibility Test and Flexibility Index
- ▶ Dini and Hadamard Derivatives in Optimization
- ▶ Global Optimization: Envelope Representation
- ▶ Minimax: Directional Differentiability
- ▶ Minimax Theorems
- ▶ Nondifferentiable Optimization
- ▶ Nondifferentiable Optimization: Cutting Plane Methods
- ▶ Nondifferentiable Optimization: Newton Method
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Nondifferentiable Optimization: Relaxation Methods
- ▶ Nondifferentiable Optimization: Subgradient Optimization Methods
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Quasigradient Methods in Minimax Problems

References

1. Danskin JM (1967) The theory of max-min and its application to weapons allocation problems. Springer, Berlin
2. Daugavet VA, Malozemov VN (1981) Quadratic rate of convergence of one linearization method for solving descrete minimax problems. USSR J Comput Math Math Phys 21(4):835–843
3. Demyanov VF (1995) Fixed point theorem in nonsmooth analysis and its applications. Numer Funct Anal Optim 16(1–2):53–109
4. Demyanov VF, Malozemov VN (1974) Introduction to minimax. Wiley, New York
5. Demyanov VF, Rubinov AM (1995) Constructive nonsmooth analysis. P. Lang, Frankfurt am Main
6. Demyanov VF, Vasiliev LV (1986) Nondifferentiable optimization. Springer and Optim. Software, Berlin
7. Karlin S (1959) Mathematical methods and theory in games, programming and economics. Addison-Wesley, Reading
8. Kelley JF (1960) The cutting-plane method for solving convex problems. SIAM J Appl Math 8(4):703–712
9. Laurent P-J (1972) Approximation et optimisation. Hermann, Paris
10. Panin VM (1981) On some methods for solving convex programming problems. USSR J Comput Math Math Phys 21(2):315–328
11. Polak E (1987) On the mathematical foundations of nondifferentiable optimization in engineering design. SIAM Rev 29:21–89
12. Pschenichny BN (1983) The method of linearization. Nauka, Moscow (In Russian)

Nondifferentiable Optimization: Newton Method

A. M. Rubinov
School Inform. Techn. and Math. Sci. University Ballarat, Ballarat, Australia

MSC2000: 49J52, 90C30

Article Outline

Keywords

Newton method; Variational inequality problem; Nonlinear complementary problem; Smoothing; Approximation of nonsmooth mappings; Semismooth mapping; Superlinear convergence

One of the main approaches to solving equations and unconstrained optimization problems with differentiable functions involved is based on the *Newton*

method (NM). The classical version of this method (in a finite-dimensional setting) deals with an equation

$$F(x) = 0 \tag{1}$$

with a differentiable mapping F. Having an approximate solution x_k ($k = 0, 1, \dots$) we wish to find a new one $x_{k+1} = x_k + y$ which is 'better' than x_k. Since there exists the derivative ∇F, we can approximate the mapping $y \longmapsto F(x_k + y)$ by the mapping $y \longmapsto F(x_k) + \nabla F(x_k)y$ and hence approximate the equation $F(x_k + y) = 0$ by the equation

$$F(x_k) + \nabla F(x_k)y = 0\,. \tag{2}$$

It is assumed that the linear mapping $\nabla F(x_k)$ is invertible. A new approximation x_{k+1} to the solution has a form $x_{k+1} = x_k + y_k$ where $y_k = -(\nabla F(x_k))^{-1}(F(x_k))$ is a solution of the equation (2). (Sometimes it is more convenient to add a degree of freedom and consider a vector $x_{k+1} = x_k + t_k y_k$ with $t_k > 0$ as a new approximation.) Thus a classical smooth version of the NM has a form

$$x_{k+1} = x_k - (F'(x_k))^{-1}(F(x_k))\,. \tag{3}$$

There are many methods for solving an unconstrained minimization problem

$$f(x) \to \min \tag{4}$$

based on the scheme (3) and its modifications. The simplest scheme:

$$x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) \tag{5}$$

is suitable for twice-continuously differentiable (C^2) functions f. This scheme allows us to find critical points of the function f.

As it turns out many optimization and related problems even with smooth objective functions and constraints can be reduced to the solution of special kinds of *nonsmooth equations* (see, for example, [20]).

We shall consider as an example a *variational inequality problem* (VIP): find a vector $x \in D$ such that

$$(y - x)^\top F(x) \geq 0 \quad \text{for all } y \in K\,, \tag{6}$$

where K is a closed convex subset of $\mathbf{R}^n$ and F is a continuously differentiable (C^1) mapping defined on an open set $D \supset K$. Let P_K be the *metric projection* onto the set C (that is $\| x - P_K x \| = \min_{y \in K} \| x - y \|$ with the Euclidean norm $\| \cdot \|$) and

$$\begin{aligned} \widetilde{F}_K(x) &= x - P_K(x - F(x)), \\ F_K(z) &= F(P_K(z)) + (z - P_K(z)). \end{aligned} \tag{7}$$

It can be shown (see, for example, [20]) that a vector x satisfies (6) if and only if x is a solution of the equation $\widetilde{F}_K(x) = 0$ and if and only if $z = x - F(x)$ is a solution of the equation $F_K(z) = 0$. Since P_K is not necessarily a smooth mapping it follows that both mappings $\widetilde{F}_K$ and F_K are not necessarily smooth.

A special case of VIP (6) with $K = \mathbf{R}^n_+$ is a *nonlinear complementary problem* (NCP): find a vector $x \in D$ such that

$$x \geq 0, \quad F(x) \geq 0, \quad x^\top F(x) = 0\,, \tag{8}$$

where $D \subset \mathbf{R}^n$ is an open set containing the nonnegative orthant $\mathbf{R}^n_+$ and F is a continuously differentiable (C^1) function defined on D. Since $P_{\mathbb{R}^n_+}(x) = x^+$, the mappings (7) have the following form:

$$\begin{aligned} \widetilde{F}_K(x) &= \min(x, F(x)), \\ F_K(z) &= F(x^+) - x^-, \end{aligned} \tag{9}$$

where $x^+ = \max(x, 0)$ and $x^- = \min(x, 0)$; max and min stand for componentwise maximum and minimum, respectively. Thus NCP reduces to equations with max and min operators.

The following properties of the function $m(b, c) = \min(b, c)$ are important for application to NCP: m is positive homogeneous of the first degree and the set $\{x = (x_1, x_2)$: $x_1 \geq 0$, $x_2 \geq 0$, $m(x) = 0\}$ coincides with the union of two positive semi-axes. Sometimes it is more convenient to consider functions with the same properties and also being C^1 on $\mathbf{R}^2$ except the origin. One example from this large class of functions is the so-called *Fischer–Burmeister function* [8])

$$\phi(b, c) = (b^2 + c^2)^{1/2} - (b + c)\,. \tag{10}$$

If

$$H(x) = (\phi(x_1, F_1(x)), \dots, \phi(x_n, F_n(x)))^\top\,, \tag{11}$$

then x is a solution of NCP if and only if $H(x) = 0$.

Thus it became necessary to extend the NM to enable the efficient solution of nonsmooth equations.

Note that this extension is not generally possible. B. Kumer [15] has found an example of a function F defined on the real line, which enjoys very good properties (F is Lipschitz, strictly monotone, directionally differentiable everywhere and Fréchet differentiable at the solution of the equation (1)) and such that NM alternates for almost all starting points.

One of the first contributions to nonsmooth NM was given by S.M. Robinson [27] for solving feasibility problems involving inequality constraints and by M. Kojima and S. Shindo [14] for solving (1) with a piecewise smooth mapping F. Various versions and modifications of the NM for nonsmooth equations have been developed during the last decade. In particular *damped NM* and *smoothing NM* enjoy global convergence under certain monotonicity assumptions. There are finite-dimensional and infinite-dimensional settings of this problem. Here we consider only finite-dimensional versions of the nonsmooth NM. For infinite-dimensional problems see for example [7,16].

The first problem which arises in the study of nonsmooth NM is to find suitable *approximations of nonsmooth mappings*. Many authors suggested various versions of such approximations. Two types of approximations are mainly considered. One of them is an approximation by a certain set-valued mapping $x \to V(x)$ where $V(x)$ is a set of invertible matrices. In this case the equation (2) is replaced by a linear equation $F(x_k) + A_k y = 0$, where A_k is an arbitrary matrix from $V(x_k)$. Thus NM in such a setting has a form

$$x_{k+1} = x_k - A_k^{-1} F(x_k). \tag{12}$$

The second type is based on an approximation by means of a certain nonlinear mapping F'. In such a case the method can be presented in the following form: $x_{k+1} = x_k + y_k$, where y_k is a solution of the nonlinear equation

$$F(x_k) + F'(x_k)(y) = 0. \tag{13}$$

It is assumed that the auxiliary nonlinear equation (13) can be solved relatively easily.

There is a general approach [16] based on the set-valued approximation $\mathcal{F}(x, y)$ of a single-valued locally Lipschitz mapping F which includes both of the above mentioned types. This approach clarifies two conditions which lead to convergence of NM: first the uniform injectivity (invertibility) of the approximating mapping at a neighborhood Y of a solution x^*: there exists $c > 0$ such that

$$\|u\| \geq c\,\|y\| \quad \text{for all } y \in Y,\ u \in \mathcal{F}(x, y), \tag{14}$$

and secondly, that the mapping F should accomplish a relatively good approximation at this neighborhood:

$$F(x) + \mathcal{F}(x, y) \subset \mathcal{F}(x, y + (x - x^*) + o(\|x - x^*\|), \tag{15}$$

where $o(y)/\|y\| \to 0$ as $\|y\| \to 0$.

It can be shown (see [16]) that for many concrete situations (15) is equivalent to

$$F(x) + \mathcal{F}(x, x^* - x) \subset o(\|x - x^*\|)B, \tag{16}$$

where B is the unit ball. The local speed of convergence is determined by the number c and the function o in (15). Conditions (14) and (15) are necessary for convergence of NM under some additional assumptions ([16]).

We now turn to the first type of approximation. If F is a locally Lipschitz mapping then the *B-subdifferential* $\partial_B F(x)$ and *Clarke generalized Jacobian* $\partial_{\mathrm{Cl}} F(x)$ are considered as approximations. By definition

$$\partial_B F(x) = \left\{ \lim_{x_k \to x} \nabla F(x_k) : \begin{array}{c} F \text{ differentiable} \\ \text{at } x_k \end{array} \right\}$$

and $\partial_{\mathrm{Cl}} F(x)$ is the convex hull of $\partial_B F(x)$. L. Qi suggested another approximation, the *C-differential* [23], which can be applied to all continuous (not just Lipschitz) mappings. By definition the C-subdifferential T is a compact-valued upper-semicontinuous mapping such that $F(x + u) = F(x) + A(u) + o(u)$ for any $A \in T(x + u)$. (It is important that T is a mapping, not an individual set at the point x; approximation near the point x is accomplished by matrices from the sets $T(y)$ for all y sufficiently close to x.) Close construction of point-based set-valued approximation is studied in [30], where connections with constructions from [16,28] are mentioned. There exist exact calculus rules for C-differentials which allow their relatively easy computation. An interesting example of approximation is given by *approximate Jacobians* (see, for example, [10]). This construction is again applicable for all continuous mappings and allows the unification of various approaches to approximation.

Local convergence based on a matrix approximation $V(x)$ can be proved only if all matrices A belonging to $V(x)$ with x sufficiently close to a solution x^*, accomplish a sufficiently good approximation (compare with [16]). Convergence can be, in particular, proved if the mapping F is semismooth (see [25]) at the point x^* or enjoys some properties close to semismoothness (see, for example, [10]). The mapping F is called *semismooth* with respect to a matrix approximation $V(x)$ at a point x^* if the directional derivative $F'(x, \cdot)$ exists at all points x close to x^* and

$$\begin{aligned} &Au - F'(x,u) = o(\|u\|), \\ &A \in T(x+u), \quad u \to 0, \end{aligned} \tag{17}$$

that is, each matrix $A \in T(x+u)$ approximates the directional derivative at the point x in the direction u. (Semismoothness was originally introduced by R. Mifflin for real-valued functions in 1977.) In some applications strong semismoothness is required. The mapping F is called *strongly semismooth* at a point x if there exists a number K such that

$$\begin{aligned} &\|Au - F'(x,u)\| \le K\|u\|^2, \\ &A \in T(x+u), \quad u \to 0. \end{aligned} \tag{18}$$

Semismoothness and strongly semismoothness can be easily verified in many concrete situations. The simplest example of a strongly semismooth mapping is a coordinatewise maximum (or minimum) of a finite number of C^2 mappings.

It is well known that the NM produces for differentiable mappings a sequence which converges to a solution very fast. As it turns out this property holds also in nonsmooth setting under some regularity conditions. Qi and J. Sun demonstrated that for semismooth mappings the rate of convergence is *Q-superlinear* [25], that is

$$\lim_{k\to+\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0.$$

Under some additional assumptions (for example, strong semismoothness) [25] it can be shown that the rate of convergence is *Q-quadratic*:

$$\lim_{k\to+\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^2} < +\infty.$$

The so-called *Kantorovich scheme* [12] in the study of NM can also be extended for nonsmooth equations (see for example [16,25,27,29]). This scheme allows one to prove not only the convergence of the NM but also the existence of a solution in a small neighborhood of the starting point x_o if some combinations of such parameters as $\|F(x_o)\|$ and the uniform boundary of norm of inverse matrices, the value K in (18), the radius of neighborhood, are sufficiently small.

Various kinds of nonlinear approximation are used for setting the NM in the framework of the second approach.

The *B-derivative* [19,22], that is uniformly (with respect to direction) directional derivative, is often used as a nonlinear approximation. One more tool for such an approximation is the *codifferential* [5]. One of the difficult tasks arising under application of the NM based on nonlinear approximation is to solve the auxiliary subproblem (13). For B-derivative based NM, J.S. Pang [19] proposed solving this problem inexactly. A version of NM under the assumption that each matrix of $\partial_B F(x)$ is invertible (the so-called *BD*-regularity) was studied in [22]. This assumption is weaker than invertibility of B-derivative and also nonsingularity of all matrices of $\partial_{Cl} F(x)$. A different approach to nonlinear approximation was proposed by Robinson [28], who introduced the so-called *point-based approximation*. This approach is convenient in the study of equations with mappings F_K (see (7)) generated by the projection P_K on a convex set K. Q-superlinear convergence and (for strong approximations) Q-quadratic convergence can be proved for various kinds of nonlinear approximations.

An abstract version of the second approach, based on the Kantorovich scheme, was proposed in [29]. The mapping $F^{\odot}(x, v)$ is called an *approximator* for the mapping F at the point x in the direction v if $F(x+v) = F(x) + F^{\odot}(x, v) + o(v)$ where $\lim_{t\to+0} t^{-1}o(tv) = 0$. Assume that an approximator $F^{\odot}$ accomplishes a strong approximation: there exist a number K such that $\|F(x+u) - F(x) - F^{\odot}(x, u)\| \le K\|u\|^2$ for all x close to the starting point x_0, and $F^{\odot}$ enjoys the following property (which is weaker than the pseudo-Lipschitz property [1]): there exists a number L such that the equation $F^{\odot}(x, u) = y$ has a solution $\overline{u}$ with $\|\overline{u}\| \le L\|y\|$ for all sufficiently small y. Then the convergence of the NM based on $F^{\odot}$ can be proved [29] by means of the Kantorovich scheme under some typical for this scheme assumptions. The inequality $\|x^* - x_k\| \le \gamma$

2^{-k} holds for the Newton sequence (x_k) with a number $\gamma > 0$.

As a rule NM converges to a solution (even in the smooth setting) only if a starting point is sufficiently close to this solution. Thus the question arises: is it possible to find modifications of NM which provide *global convergence* for some equations, that is convergence from an arbitrary initial point? One of such modifications is the *damped Newton method*. For instance, if the NM uses a positively homogeneous approximation $F'(x_k, u)$, such as a Jacobian in the smooth case, or directional derivative [7,13,15,19], then x_{k+1} is chosen as $x_k + \lambda_k y_k$ for some $\lambda_k \in (0, 1]$ such that

$$\|F(x_{k+1})\| \cong (1 - \lambda_k)\,\|F(x_k)\| < \|F(x_k)\|\,. \qquad (19)$$

This can be generalized to nonpositively homogeneous approximations such as point based approximations [3,21] by setting up a path $p_k(\lambda)$ joining $x_k = p_k(0)$ to $x_k + y_k = p_k(1)$, where y_k is found by the NM, such that $F(p_k(\lambda)) = (1 - \lambda)\,F(x_k) + o(\lambda)$. Then it is easy to determine $\lambda_k \in (0, 1]$ such that for $x_{k+1} = p_k(\lambda_k)$, (19) holds.

There is a close connection between the damped NM and the so-called *list square merit function* of the operator F:

$$\theta(x) = \frac{1}{2}\,\|F(x)\|^2\,.$$

For some nonsmooth operators F arising from NCP and related problems the function θ is continuously differentiable. This property is very useful in the study of NM and damped NM [11].

One of modifications of the nonsmooth NM is the so-called *smoothing Newton method*, which is also called *splitting NM* or *homotopy NM*. This method based on an approximation of the operator F in (1) by a smooth function G. Usually a smoothing damped NM is studied. This method has the following form

$$x_{k+1} = x_k - t_k \nabla G_x(x_k, \varepsilon_k)^{-1} F(x_k), \qquad (20)$$

where $G(x, \varepsilon)$ is a smoothing approximation of the mapping F, that is for any $\varepsilon > 0$ the function $x \longmapsto G(x, \varepsilon)$ is continuously differentiable and $\|F(x) - G(x, \varepsilon)\| \to 0$ as $\varepsilon \to 0$.

Various types of *smoothing functions* are used in optimization for a long time (see, for example, [13]). Some of them can be constructed by means of the so-called *Chen–Harker–Kanzow–Smale function*:

$$\xi(u, \varepsilon) = \frac{1}{2}\left((u^2 + 4\varepsilon^2)^{\frac{1}{2}} + u\right), \quad (u, e) \in \mathbb{R}^2,$$

which serves for an approximation of $\max(0, u)$ (see, for example, [24]). In particular, it is possible to find smoothing operators for operators defined by (9) by means of the function ξ. An interesting approach to the smoothing methods can be found in [3]. Global convergence of the method (20) can be proved under some assumptions. If the gradient of the smoothing function G is fairly close to some approximations of the operator F then the method converges Q-superlinearly (quadratically) [4,24].

There is a close connection between smoothing NM and *interior point methods* (see, for example, [21]).

For some special classes of problems it is possible to obtain stronger results than in the general setting (see, for example, [6,11,18,26]). On the other hand there are modifications of the nonsmooth Newton method for the solution of generalized equations of the form $y \in F(x)$ with a set-valued mapping F (see, for example, [2,7,16]).

See also

- ▶ Automatic Differentiation: Calculation of Newton Steps
- ▶ Dini and Hadamard Derivatives in Optimization
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Global Optimization: Envelope Representation
- ▶ Interval Newton Methods
- ▶ Nondifferentiable Optimization
- ▶ Nondifferentiable Optimization: Cutting Plane Methods
- ▶ Nondifferentiable Optimization: Minimax Problems
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Nondifferentiable Optimization: Relaxation Methods
- ▶ Nondifferentiable Optimization: Subgradient Optimization Methods
- ▶ Unconstrained Nonlinear Optimization: Newton–Cauchy Framework

References

1. Aubin J-P, Ekeland I (1984) Applied nonlinear analysis. Wiley, New York
2. Aze D, Chou CC (1995) On a Newton type iterative methods for solving inclusions. Math Oper Res 20:790–800
3. Chen C, Mangasarian OL (1996) A class of smoothing functions for nonlinear and mixed complementarity problems. Comput Optim Appl 5:97–138
4. Chen X, Qi L, Sun D (1998) Global and superlinear convergence of the smoothing Newton method and its application to general box constrained variational inequalities. Math Comput 67:519–540
5. Demyanov VF (1995) Fixed point theorem in nonsmooth analysis and its applications. Numer Funct Anal Optim 16:53–109
6. Dirkse SP, Ferris MC (1995) The path solver: A nonmonotone stabilization scheme for mixed complementary problems. Optim Methods Softw 5:123–156
7. Dontchev AL (1996) Local convergence of the Newton method for generalized equations. CR Acad Sci Paris Ser I 322:327–331
8. Fisher A (1992) A special Newton-type optimization method. Optim 24:269–284
9. Harker PT, Xiao B (1990) Newton's method for nonlinear complimentary problem: A B-differentiable equation approach. Math Program 48:339–357
10. Jeyakumar V, Luc DT (1998) Approximate Jacobian matrices for nonsmooth continuous maps and C^1-optimization. SIAM J Control Optim 36:1815–1832
11. Jiang H, Ralph D (1998) Global and local superlinear convergence analysis of Newton-type methods for semismooth equations with smooth least squares. In: Fukushima M, Qi L (eds) Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods. Kluwer, Dordrecht, pp 181–210
12. Kantorovich L, Akilov G (1964) Functional analysis in normed spaces. MacMillan, New York
13. Kaplan AA (1981) On an approach to the solution of convex programming problems. Soviet Math Dokl 23:588–591
14. Kojima M, Shindo S (1986) Extensions of Newton and quasi-Newton methods to system of PC^1 equations. J Oper Res Soc Jpn 29:352–374
15. Kummer B (1988) Newton's method for non-differentiable functions. In: Guddat J (ed) Adv Math Optim. Akademie, Berlin, pp 114–125
16. Kummer B (1992) Newton's method based on generalized derivatives for nonsmooth functions: convergence analysis. In: Oettli W, Pallaschke D (eds) Advances in Optimization. Springer, Berlin, pp 171–194
17. Kummer B (1995) Approximation of multifunctions and superlinear convergence. In: Durier R, Michelot C (eds) Recent Developments in Optimization. Springer, Berlin, pp 243–251
18. De Luca T, Facchinei F, Kanzow C (1996) A semismooth equation approach to the solution of nonlinear complementary problems. Math Program 75:407–439
19. Pang JS (1990) Newton's method for B-differentiable equations. Math Oper Res 15:311–341
20. Pang JS, Qi L (1993) Nonsmooth equations: motivation and algorithms. SIAM J Optim 3:443–465
21. Potra E, Ye Y (1996) Interior-point methods for nonlinear complementary problems. J Optim Th Appl 88:617–642
22. Qi L (1993) Convergence analysis of some algorithms for solving nonsmooth equations. Math Oper Res 18:227–244
23. Qi L (1996) C-differential operators, C-differentiability and generalized Newton methods. Applied Math Report. Univ New South Wales, Sydney
24. Qi L, Sun D (1999) Nonsmooth equations and smoothing Newton methods. In: Eberhard A et al (eds) Progress in Optimization: Contribution from Australasia. Kluwer, Dordrecht, pp 121–146
25. Qi L, Sun J (1993) A nonsmooth version of Newton's method. Math Program 58:353–367
26. Ralph D (1994) Global convergence of damped Newton's method for nonsmooth equations, via the path search. Math Oper Res 19:352–389
27. Robinson SM (1994) Newton's method for a class of nonsmooth functions. Set-Valued Anal 2:291–305
28. Robinson SM (1972) Extension of Newton's method to nonlinear functions with values in a cone. Numerische Math 19:341–347
29. Rubinov A, Zaffaroni A (1999) Continuous approximation of nonsmooth mappings. In: Eberhard A et al (eds) Progress in Optimization: Contribution from Australasia. Kluwer, Dordrecht, pp 27–58
30. Xu H (1999) Set-valued approximations and Newton's methods. Math Program 84:401–420

Nondifferentiable Optimization: Parametric Programming

SANJO ZLOBEC
Department Math. Statist., McGill University, West Montreal, Canada

MSC2000: 90C05, 90C25, 90C29, 90C30, 90C31

Article Outline

Keywords

Parametric programming model; Sensitivity; Stability; Optimal parameter; Parameter identification; Point-to-set mapping

Mathematical descriptions of real-life problems are typically stated in terms of *decision variables* x and parameters θ. These objects may be related through a system of equations and inequalities such as

$$x \in \mathcal{F}(\theta) = \left\{ x \in \mathbf{R}^n : \begin{array}{l} g^i(x,\theta) \le 0,\ i \in I, \\ h^j(x,\ \theta) = 0,\ j \in J \end{array} \right\},$$

where g^i, h^j: $\mathbf{R}^n \times \mathbf{R}^p \to \mathbf{R}$, $i \in I$, $j \in J$, are some functions, and I and J are finite index sets. Problems of the form

$$\begin{cases} \min\limits_{(x)} & f(x,\theta) \\ \text{s.t.} & x \in \mathcal{F}(\theta), \end{cases}$$

where θ is allowed to vary over some set F in $\mathbf{R}^p$, are termed *parametric programming models* (PPM). *Parametric programming* (PP) is the study of parametric programming models. A local analysis of these models, around a fixed θ, is referred to as *sensitivity analysis* (SA). In particular, SA is concerned with changes of the minimal value subject to small perturbations of the parameter. Parametric programming is a huge area containing, or closely related to, many topics, such as path following methods (cf. ► Parametric optimization: Embeddings, path following and singularities), sensitivity in semi-infinite programming, constraint qualifications (cf. also ► First order constraint qualifications; ► Second order constraint qualifications), bilevel programming (cf. ► Bilevel programming: Introduction, history and overview), etc. We will refer to some of these topics hereby only in passing. Since every equality constraint can be replaced by two inequalities, one can assume that $J = \emptyset$. Then the model is said to be *linear* (resp. *convex*) if the functions $f(\cdot, \theta)$, $g^i(\cdot, \theta)$: $\mathbf{R}^n \to \mathbf{R}$, $i \in I$, are linear (resp. convex) for every $\theta \in \mathbf{R}^p$.

Historical Outline

Parametric programming has its roots in the study of linear programs:

$$\mathrm{LP} \begin{cases} \min & c^\top x \\ \text{s.t.} & Ax \le b, \quad x \ge 0, \end{cases}$$

where one or more coefficients of the vectors b and c, or the matrix A, are considered as parameters and allowed to vary. The study can be traced to the 1950s literature. According to [16], the right-hand side changes in a linear program were investigated by W. Orchard-Hays in his unpublished Master's thesis (1952). The term 'parametric' LP was used in [41]. The classical problems of SA and PPM dealt mainly with pivoting and the simplex method. A different approach that uses polyhedral structures rather than the simplex method was developed in [46,47,48,49]. A classical parametric problem in LP is to determine the range of perturbations for specific parameters in b and c, that preserve optimal bases. A related problem is to determine the range for which an optimal solution exists. This range is called the 'critical set'. Various approaches to solving these problems have been successfully implemented in commercial software packages and adjusted to particular situations, e. g., data envelopment analysis [45]. It is well known that difficulties may arise when the problem under consideration is degenerate (i. e., when optimal basis is not unique). In that case the commercial packages may provide essentially different results, that is to say, the information could be 'confusing and hardly allows a solid interpretation'; see, e. g., [5], where the claim is demonstrated on a transportation problem. The study of changes of the parameters in [5] is a departure from the classical approach. Instead of employing local analysis and pivoting, the authors make use of the strict complementarity condition and optimal partitioning in order to construct and study the behavior of the *optimal value function* for the right-hand side and objective-function perturbations. They do it for both LP and convex quadratic programming and obtain sharp intervals. Other approaches used to study the effect of perturbations of parameters in LP, including the 'tolerance approach' (where variations may occur simultaneously and independently), are described in [63,64]. The classical texts on sensitivity analysis and *parametric linear programming* include [9,15,37,48,49].

([15] lists 1031 references.) Almost 1000 items only on degeneracy in LP are listed in [17].

A major obstacle to using linear algebra and calculus methods in the study of linear models is that these methods are not suited to explain continuity properties of the model. Indeed, the model does not generally react 'continuously' to continuous changes of the parameters in the coefficient matrix. Let us illustrate such a situation.

Example 1 Consider the model

$$\begin{cases} \min & x \\ \text{s.t.} & \theta x = 0, \\ & -1 \le x \le 1. \end{cases}$$

When $\theta = 0$, then the feasible set $\mathcal{F}(\theta)$ is the segment $-1 \le x \le 1$, the optimal solution is $x = -1$, and the optimal value is -1. For any perturbation $\theta \neq 0$, all three objects jump to zero.

An example of a linear model where the critical set is disjoint and not closed is given in [9, pp. 114–116]. Another one with a matrix of full row rank where both the feasible set and the set of optimal solutions experience jumps under continuous perturbations of the *parameter* in the interior of the critical set is given in [42]. We extend this example to an LP in canonical form with a full row rank matrix below.

Example 2 Consider the model

$$\begin{cases} \min & -x_2, \\ \text{s.t.} & x_1 + x_2 + x_3 = 1, \\ & x_1 + \theta x_2 - x_4 = 1, \\ & x_i \ge 0, \quad i = 1, \dots, 4. \end{cases}$$

A unique optimal solution at $\theta = 1$ is $x = (x_i) = (0, 1, 0, 0)$ and the optimal value is -1. However, for any perturbation $\theta = 1 - \epsilon$, $\epsilon > 0$, the solution jumps to another unique optimal solution $(1, 0, 0, 0)$, and the optimal value becomes zero.

In order to study reactions of a model to continuous perturbations of data, the feasible set and the optimal solutions set can be viewed as images of *point-to-set mappings* with a domain in the space of parameters. Hence the study of continuity of these and related objects, such as *Lagrange multiplier sets*, requires basic tools of point-to-set topology. These tools have been used in mathematical programming sporadically and in different contexts, e. g., in an analysis of convergence of numerical algorithms in [50,65]. After papers such as [26], the point-to-set approach to the study of parametric programming has become standard. The first text on the theory of *nonlinear parametric optimization* is [4], written by several authors from the 'Berlin school of parametric optimization' initiated by F. Nožička. (Twenty nine students obtained doctorates under his supervision.) This text contains 30 pages of bibliography on PP. A classical text on methodology used in perturbation analyses in nonlinear programming is [13], and one on path following methods in PP is [25]. A unified approach to general perturbations with applications to system analysis and numerical optimization is given in [34]. Since the late 1970s there has been an outburst of research activities in PP. For instance, to date (1998) there have been 20 annual symposia on mathematical programming with data perturbations held at the George Washington Univ., and the International Conference on Parametric Optimization and Related Topics is being held bi-annually since 1985. There have been at least 15 books written on parametric programming.

The study of parametric programming can be roughly divided into three general areas: *stability*, optimality and numerical methods, and applications.

Stability

In this area one mainly studies continuity properties of the feasible set $\mathcal{F}(\theta)$, the set of all optimal solutions $\mathcal{F}^\circ(\theta) = \{x^\circ(\theta)\}$, the set of all Lagrange multipliers $U = U(\theta)$, and the optimal value $f^\circ(\theta) = f(x^\circ(\theta), \theta)$, as the parameter θ varies. Here f° is a function, while $\mathcal{F}$: $\theta \to \mathcal{F}(\theta)$, $\mathcal{F}^\circ$: $\theta \to \mathcal{F}^\circ(\theta)$, and $U \to U(\theta)$ are point-to-set mappings. If the constraints in a *parametric programming model* are continuous functions, then $\mathcal{F}$ is a closed mapping. In order to guarantee continuity of the mapping $\mathcal{F}$ one requires an extra condition, e. g., that $\mathcal{F}$ be lower-semicontinuous (or, equivalently, open). If the PPM is convex and $\mathcal{F}^\circ(\theta^*) \neq \emptyset$ and bounded, at some θ^*, then continuity of $\mathcal{F}$ locally implies both the existence of optimal solutions and continuity of the *optimal value function*, and the mapping $\mathcal{F}^\circ$ is closed. However, continuity of $\mathcal{F}$ does not imply continuity of U.

Conditions for the existence of a differentiable path of solutions $x^\circ(\theta)$ in a general model, as the parameter θ varies, are given in [14]. They are extended to a continuously differentiable path of saddle points in [13]; see also [19,20]. The results are established by applying an appropriate implicit function theorem to a necessary condition for optimality. The reference [14] also contains an explicit formula for the partial derivatives, as well as approximations based on classical penalty functions. For a state-of-the-art about *Lipschitz stability* for linear and quadratic (also nonconvex) programs up to 1987, see [30]. For a recent (1998) guided tour of sensitivity analysis that yields lower and upper estimates for the optimal value function see [6].

If the feasible set mapping $\mathcal{F}$ is continuous and the set of optimal solutions is nonempty and bounded, then such models are often termed stable. In *convex parametric programming* continuity of $\mathcal{F}$ is implied by *Slater's condition*. In the fundamental paper [51], stability for linear systems in the canonical form, subject to perturbations of all data, is characterized by the existence of a positive feasible solution $x > 0$. The result is stated for linear inequalities in partially ordered Banach spaces. It is extended in [52] to nonlinear inequalities over a closed convex set. When perturbations of specific coefficients are considered, then the existence of a positive solution is not a necessary, but it is rather a sufficient condition for stability. Also, in this case, chunks (regions) of the parameter space attached to a fixed parameter θ^*, where $\mathcal{F}$ is lower semicontinuous, are termed *regions of stability* at θ^* see [73,74]. (In the model from Example 2, a region of stability at $\theta^* = 1$ is $\theta \geq 1$.) Such regions can often be calculated globally; one of these is the set of all paths emanating from θ^* on which the constraints satisfy Slater's condition. For a list of regions of stability and a necessary condition for stability in convex PP see [69]. These regions are of independent interest in, e. g., the study of random decision systems with complete connections [66] and linear programming [7]. The radius of the largest ball centered at θ^*, with the property that the model is stable at its every interior point θ, is the *radius of stability* at θ^*, e. g., [69]. It is a measure of how much the system can be uniformly strained from θ^* before it starts breaking down. In a linear parametric programming model in canonical form with a full row rank coefficient matrix, stability is implied by the existence of a nondegenerate basic feasible solution. On the other hand, *instability* (i. e., loss of continuity of the feasible set mapping) typically occurs in situations of 'enforced optima' such as lexicographic and multilevel programming, including *von Stackelberg games* of market economy. For example, in bilevel programming, instability occurs when the optimal solutions set of the follower (lower level decision maker) loses its lower semicontinuity. The leader's (upper level decision maker's) model is then unstable, because its feasible set is the set of optimal solutions of the follower. In this situation the leader's optimal value function typically experiences a discontinuity even if the follower's model is globally stable; see [43, Chapt. 13; 16]. The notion of stability is not uniquely defined, see, e. g. [4,13,21,31,34]. *Structural stability* and continuous deformations of nonlinear programs have been studied in, e. g., [22,28,29,33]. In particular, the *'topological stability'* of the feasible set (i. e., homeomorphy with respect to all sufficiently small perturbations up to second order of the involved functions) is proved to be equivalent to the Mangasarian–Fromovitz constraint qualification being satisfied at the feasible points; see [24]. Characterizations of stability (local existence and uniqueness) of stationary points and Karush–Kuhn–Tucker points with respect to all sufficiently small perturbations up to second order are given, respectively, in [32,53]. For a study of stability with nondifferentiable data see [3,44], also [57]. Some difficulties with an abstract formulation of parametric programming are mentioned in [43, Chapt. 14]; see also [2,58,75]. Stability in optimal control is studied in, e. g., [38,39].

Optimality and Numerical Methods

A parameter θ is said to be *locally* (resp. *globally*) optimal if it locally (resp. globally) optimizes the optimal value function $f^\circ(\theta)$ over its feasible set $F = \{\theta: \mathcal{F}(\theta) \neq \emptyset\}$. Calculation and characterization of *optimal parameters* are basic problems of parametric programming. For convex PPM one can formulate necessary and sufficient conditions for local (and global) optimality of the parameter, e. g., [68,69,70,71]. These conditions are expressed in terms of *saddle-point inequalities* and local results typically require conditions such as uniqueness of the optimal solution in the x component and lower semicontinuity of the feasible set mapping. They

are simplified under *input constraint qualifications* [55]. The optimal value function is not generally known analytically, it is generally nondifferentiable, nonconvex (nonconcave), and discontinuous even for linear models. However, for the right-hand side and objective-function perturbations in linear models it can be constructed, e. g., [5].

Calculation of optimal parameters is often achieved by using the so-called *'marginal value formula'*. This formula gives the derivative of the optimal value function on a prescribed path in terms of the derivatives of the Lagrangian function relative to x and θ. At each iteration the calculation consists of two parts: first, an improvable feasible path is determined and then a step-size problem is solved on the path by a search method. Optimization of the optimal value function by stable perturbations of the parameters (also called *input optimization*) in convex PP is a challenging problem and no satisfactory theory or numerical methods presently seem to exist, e. g., [71]. In contrast, the theory of path following methods, based on nonlinear programming optimality conditions and used when data depend on one scalar parameter, is well developed, although 'not successful in every case' see, e. g., [23,25]. In order to improve the *path following approach* these authors propose jumps between connected components in the sets of local minimizers and generalized critical points.

Applications

Classical sensitivity analysis, when applied to the right-hand side perturbations in linear programming, provides interpretation of Lagrange multipliers as *shadow prices*. The results have been extended to convex programming using the marginal value formula, e. g., in [12]. Genuine applications of parametric programming extend far beyond classical sensitivity analysis. Some of them are related to the fundamental notion of a *well-posed problem* in the sense of Hadamard. These are problems in applied mathematics which have a unique solution and the solution changes continuously when the parameters (e. g., boundary conditions in differential equations) change continuously. Problems that are not well-posed are called *ill-posed*. According to Fritz John (see [62, p. ix]) 'the majority of applied problems are, and always have been, ill-posed, particularly when they require numerical answers'. There is a more general notion of well-posedness (for problems with nonunique solutions); see [10]. Many problems of mathematical physics can be formulated as optimization problems and continuous dependance of the solution on the boundary conditions can be studied using PP. In the context of mathematical programming, some of the first applications of PP included the study of convergence of numerical algorithms. The rate of convergence can be determined using continuity properties of point-to-set mappings; see, e. g., [14,50,54,65]. Some of the ambiguities that occur while solving ordinary LP can be understood and resolved by PP. For example, in some models describing real-life problems, such as the one reported in [62, pp. 212–213], significant *jumps of optimal solutions* occur when some data are perturbed, while the respective values of the objective function are comparatively close. This is a typical behavior of stable linear programming models when the optimal solutions mapping is not continuous. The authors of [62] suggest a method for stabilization of optimal solutions of such programs by 'Tikhonov regularization'.

The results on optimal parameters in parametric programming have many applications including machine scheduling [61,67], restructuring of the work force in a textile mill [71,72], ranking of efficiently administered university libraries by their robustness of data [40,72], the study of systems of differential equations under matrix perturbations in robust analysis and control [27], as well as approximation theory, especially in the problems of *best fitting to data*. These problems are formulated as follows: given a set of points, one wishes to determine a function (from a prescribed class of functions, e. g., linear) that best approximates these points. After forcing the points to satisfy the function, the problem generally reduces to an inconsistent system of equations for which one determines a best approximate solution. The solution is given in terms of *decision variables* (such as the slope of the line and its intersection with the y-axis, in the linear cases of two-dimensional data in the (x, y)-plane). If the Euclidean norm is used then the solutions are called the *least squares solutions*. However, the problem can also incorporate estimates of errors made in measuring data. The errors may fall within some known lower and upper bounds. One can consider the data vector as a pa-

rameter in which case the best approximation problem assumes a more general form. It consists of finding perturbations of data, within specified boundaries, for which one achieves the best fit. The problem is called the *generalized least squares problem* in the context of the Euclidean space. If the analytic form of the function that approximates data is known, and if it contains some 'parameters' to be determined, then the best approximation problem is called the *parameter identification problem*.

Example 3 One may wish to determine the constant of gravity g, the initial position s_0, and the initial velocity v_0 of a falling object. It is known that the object is governed by Newton's second law of motion $s = s_0 + v_0 t + gt^2/2$. Measuring $s = s(t)$ at various times t, one obtains a generally inconsistent system of linear equations in the 'parameters' s_0, v_0, and g. After minimizing a norm of the residual vector one identifies the parameters. However, one may also take into consideration relative errors that have been made in reading t and $s(t)$. The generalized parameter identification problem is to determine the best fit within the allowable errors. This approach typically gives more accurate results. In the context of parametric programming models, the optimal errors are optimal values of the parameter, while the 'parameters' to be identified, like s_0, v_0, and g, are actually decision variables.

Parameter identification problems are used in many areas. For example, in the biological sciences they are used in attempts to find kinetic constants which can quantitatively describe certain *biochemical processes*. Typically, experiments are performed with tracers (labeled with radioactive or stable isotopes), then experimental tissue radioactivity curves are fitted to a biological model to find kinetic parameters. On the basis of these kinetic parameters one can calculate regional glucose utilization in the brain [60], serotonin synthesis [8], aromatic amino acid activity [18] and receptor densities [36].

Optimal parameters are also important for *post-optimality analyses* of linear programs. Using PP one can answer basic questions like: Given an optimal solution of an LP, for what perturbations of data does the solution remain optimal? The following example exposes the problem.

Example 4 Consider the linear program in one variable with zero-value objective function:

$$\begin{cases} \min & 0 \cdot x \\ \text{s.t.} & -1 \le x \le 1. \end{cases}$$

A (global) *minimizer* is $x^* = -1$. Now suppose that this program belongs to the class of perturbed programs

$$\begin{cases} \min & \theta^2 \cdot x, \\ \text{s.t.} & -1 \le x \le 1, \end{cases}$$

when θ is fixed at $\theta^* = 0$. Then, for perturbations $\theta \neq 0$, the point (x^*, θ^*) is actually a local *maximizer*! In situations like these some optimal solutions of linear programs may be 'better' than others. Here, say, $x = 1$ is 'better' than $x^* = -1$, because its local optimality is not affected by perturbations of (x, θ).

The optimality-preserving problems, at the global optimality level, are solved for linear models using the saddle-point optimality conditions along θ-paths and after setting the terms corresponding to the components of the x variable equal to zero. Two closely related problems are:

i) given an infeasible point x (e. g., a prescribed profile of production that one wishes to achieve), find perturbations of data θ that make the point feasible; and
ii) given a feasible but nonoptimal point, find perturbations that make the point optimal.

Many results from convex parametric programming can be adjusted to work for partly convex programs (PC) and more general mathematical programs, e. g., [1,23,70]. Partly convex programs are programs which, after 'freezing' some of the coordinates in x, become convex programs in the remaining variables. The program from Example 4, with the objective function $x_2^2 \cdot x_1$ and the constraint $-1 \le x_1 \le 1$, is a PC program. (Identify $x_2 = \theta$.) Since every mathematical program with twice continuously differentiable functions can be formulated as a *partly convex program*, see [35], one can in principle study (and solve) many mathematical programs by studying PC programs and convex parametric programming models.

Numerous *applications of parametric programming* are mentioned in the classical texts [4,14,15,34]. Parametric programming has found many applications in discrete optimization, transportation problems (e. g.,

[59]), economics and finance (e. g., [11,56]), approximation, as well as in multi-objective, multilevel, stochastic and global optimization (e. g., ► Parametric global optimization: Sensitivity). Some of the recent research in parametric programming has been focused on connections between polynomial complexity and perturbation theory, stability results for nonunique solutions, control, semi-infinite programs (e. g., [28]), and nonsmooth problems.

See also

- ► Bounds and Solution Vector Estimates for Parametric NLPs
- ► Dini and Hadamard Derivatives in Optimization
- ► Global Optimization: Envelope Representation
- ► Multiparametric Linear Programming
- ► Multiparametric Mixed Integer Linear Programming
- ► Nondifferentiable Optimization
- ► Nondifferentiable Optimization: Cutting Plane Methods
- ► Nondifferentiable Optimization: Minimax Problems
- ► Nondifferentiable Optimization: Newton Method
- ► Nondifferentiable Optimization: Relaxation Methods
- ► Nondifferentiable Optimization: Subgradient Optimization Methods
- ► Parametric Global Optimization: Sensitivity
- ► Parametric Linear Programming: Cost Simplex Algorithm
- ► Parametric Mixed Integer Nonlinear Optimization
- ► Parametric Optimization: Embeddings, Path Following and Singularities
- ► Selfdual Parametric Method for Linear Programs

References

1. Antipin A (1966) Equilibrium programming problems: Prox-regularization and prox-methods. In: Gritzmann P (ed) Recent Advances in Optimization, Proc. 8th French–German Conf. Optimization, July 21–26 1996. Springer, Berlin, pp 1–18
2. Asgharian M, Zlobec S (2000) Convex parametric programming in abstract spaces. Techn Report, McGill Univ
3. Auslender A (1984) Stability in mathematical programming with nondifferentiable data. SIAM J Control Optim 22:239–254
4. Bank B, Guddat J, Klatte D, Kummer B, Tammer T (1982) Non-linear parametric optimization. Akademie, Berlin
5. Berkelaar AB, Roos K, Terlaky T (1997) The optimal set partition approach to linear and quadratic programming. In: Advances in Sensitivity Analysis and Parametric Programming. Kluwer., Dordrecht
6. Bonnans JF, Shapiro A (1998) Optimization problems with perturbations, A guided tour. SIAM Rev 40:228–264
7. Cojocaru I (1985) Régions de stabilité dans la programmation linéaire. An Univ Bucuresti Mat 34:12–21
8. Diksic M, Nagahiro S, Grdisa M (1995) The regional rate of serotonin synthesis estimated by the α-methy-tryptophan method in rat brain from a single time point. J Cerebral Blood Flow Metabolism 15:806–813
9. Dinkelbach W (1969) Sensitivitätsanalysen und parametrische Programmierung. de Gruyter, Berlin
10. Dontchev AL, Zollezzi T (1993) Well-posed optimization problems. Lecture Notes Math, vol 1543. Springer, Berlin, pp 239–254
11. Dupacova J Stability and sensitivity for a scenario-based bond portfolio management problem. In: Proc. Conf. Mathematical Methods in Economy and Industry, Liberec, June 1–5 1998
12. Eremin II, Astafiev NN (1976) Introduction to the theory of linear and convex programming. Nauka, Moscow (In Russian)
13. Fiacco AV (1983) Introduction to sensitivity and stability analysis in nonlinear programming. Acad. Press, New York
14. Fiacco AV, McCormick GP (1968) Nonlinear programming: sequential unconstrained minimization techniques. Wiley, New York
15. Gal T (1979) Postoptimal analyses, parametric programming and related topics. McGraw-Hill, New York. Updated revision of German 1973 edn; 2nd revised edition: W. de Gruyter, New York, 1995
16. Gal T (1983) Letter on historiogramme on parametric programming. J Oper Res Soc 34:162–163
17. Gal T (1994) Selected bibliography on degeneracy. Ann Oper Res 46/47:1–10
18. Garnett ES, Firnau G, Nahmias C (1983) Dopamine visualized in the basal ganglia of living man. Nature 305:137–138
19. Gauvin J, Dubeau F (1982) Differential properties of the marginal function in mathematical programming. Math Program Stud 19:101–119
20. Gauvin J, Janin R (1988) Directional behaviour of optimal solutions in nonlinear mathematical programming. Math Oper Res 13:629–649
21. Greenberg HJ, Pierskalla WP (1972) Extensions of the Evans–Gould stability theorem for mathematical programs. Oper Res 20:143–153
22. Guddat J, Jongen HTh (1987) Structural stability in nonlinear optimization. Optim 18:617–631
23. Guddat J, Jongen HTh (1988) On global optimization based on parametric optimization. In: Guddat J et al (eds) Advances in Mathematical Optimization. Akademie, Berlin, 63–79

24. Guddat J, Jongen HTh, Rückmann J-J (1986) On stability and stationary points in nonlinear optimization. J Austral Math Soc (Ser B):36–56
25. Guddat J, Vasquez FGuerra, Jongen HTh (1991) Parametric optimization: singularities, pathfollowing and jumps. Teubner and Wiley, New York
26. Hogan WW (1973) Point-to-set maps in mathematical programming. SIAM Rev 15:591–603
27. Hu T, Huang L (1996) Real stability radii and quadratic matrix inequality. Chinese Sci Bull 41:2043–2046
28. Jongen HTh, Rückmann J-J (1998) On stability and deformation in semi-infinite optimization. In: Reemtsen R, Rückmann J-J (eds) Semi-Infinite Programming. Kluwer, Dordrecht, pp 29–67
29. Jongen HTH, Jonker P, Twilt F (1983) Nonlinear optimization in R^n: Transversality, flows, parametric aspects. P. Lang, Frankfurt am Main
30. Klatte D (1987) Lipschitz continuity of infima and optimal solutions in parametric optimization: The polyhedral case. In: Guddat J et al (eds) Parametric Optimization and Related Topics. Akademie, Berlin, pp 229–248
31. Klatte D, Tammer K (1990) Strong stability of stationary solutions and Karush–Kuhn–Tucker points in nonlinear optimization. Ann Oper Res 27:285–308
32. Kojima M (1980) Strongly stable stationary solutions in nonlinear programs. In: Robinson SM (ed) Analysis and Computation of Fixed Points. Acad. Press, New York, pp 93–138
33. Kojima M, Hirabayashi R (1984) Continuous deformations of nonlinear programs. Math Program Stud 21:150–198
34. Levitin ES (1994) Perturbation theory in mathematical programming and its applications. Wiley, New York, Russian edition: 1992.
35. Liu WB, Floudas CA (1993) A remark on the GOP algorithm for global optimization. J Global Optim 3:519–521
36. Logan J, Dewey SL, Fowler AP, Brodie JS, Angrist B, Volkow ND, Gatley SJ (1991) Effects of endogenous dopamine on measures of [^{18}F] -N- methylspiroperidol binding in basal ganglia: Comparison of simulations and experimental results from PET studies in baboons. Synapse 9:195–207
37. Lommatzsch K (ed) (1979) Anwendungen der linearen parametrischen Optimierung. Akademie, Berlin
38. Malanowski K (1987) Stability and sensibility to optimal control problems for systems with control appearing linearly. Appl Math Optim 16:73–91
39. Malanowski K (1992) Second order conditions and constraint qualifications in stability and sensitivity analysis of solutions to optimization problems in Hilbert spaces. Appl Math Optim 25:51–79
40. Mann G, Zlobec S (1998) Ranking efficiency by robustness in data envelopment analysis. 20th Internat. Symp. Math. Program. with Data Perturbations, Washington, D.C., May 1998
41. Manne AS (1953) Notes on parametric linear programming. RAND Report P-468
42. Martin DH (1975) On the continuity of the maximum in parametric linear programming. J Optim Th Appl 17:205–210
43. Migdalas A, Pardalos PM, Värbrand P (eds) (1998) Multilevel optimization: Algorithms and applications. Nonconvex Optim Appl., vol 20. Kluwer, Dordrecht
44. Mordukhovich BS (1991) Sensitivity analysis in nonsmooth optimization. In: Field D, Komkov V (eds) Theoretical Aspects of Industrial Design. SIAM, Philadelphia
45. Neralić L (1997) Sensitivity in data envelopment analysis for arbitrary perturbations of data. Glasnik Mat 32:315–335
46. Nožička F (1972) Lineare parametrische Optimierung – ein Problem der Stabilität der optimalen Lösung. In: Konf Mat Met v Ekonomii, Praha, pp 45 –82
47. Nožička F (1972) Über eine Klasse von linearen einparametrischen Optimierungsproblemen. Math Operationsforsch Statist 3:159–194
48. Nožička F, Guddat J, Hollatz H (1972) Theorie der linearen Optimierung. Akademie, Berlin
49. Nožička F, Guddat J, Hollatz H, Bank B (1974) Theorie der linearen parametrischen Optimierung. Akademie, Berlin
50. Robinson SM (1974) Perturbed Kuhn–Tucker points and rates of convergence for a class of nonlinear programming algorithms. Math Program 7:1–16
51. Robinson SM (1975) Stability theory for systems of inequalities. Part I: Linear systems. SIAM J Numer Anal 12:754–772
52. Robinson SM (1976) Stability theory for systems of inequalities. Part II: Differentiable nonlinear systems. SIAM J Numer Anal 13:497–513
53. Robinson SM (1980) Strongly regular generalized equations. Math Oper Res 5:43–62
54. Robinson SM (1987) Local epi-continuity and local optimization. Math Program 37:208–222
55. Van Rooyen M, Sears M, Zlobec S (1988) Constraint qualifications in input optimization. J Austral Math Soc (Ser B) 30:326–342
56. Rupnik V (1985) Some experiences with fluid modelling of economic systems. Systems Res 2(3):211–219
57. Shapiro A (1988) Sensitivity analysis of nonlinear programs and differentiability properties of metric projections. SIAM J Control Optim 26:628–645
58. Shapiro A (1992) Perturbation analysis of optimization problems in Banach spaces. Numer Funct Anal Optim 13:97–116
59. Shubert IS, Zimmerman U (1985/6) One-parametric bottleneck transportation problems. Ann Oper Res 4:343–369
60. Sokoloff L, Reivich M, Kennedy C, DesRosiers MH, Patlak CS, Pettigrew KD, Sakurada O, Shinohara M (1977) The 14 C-deoxyglucose method for the measurement of local glucose utilization: theory, procedure, and normal values in the conscious and anesthetized albino rat. J Neurochemistry 28:897–916
61. Tharwat A, Zimmermann K (1998) On optimal choice of parameters in machine-time scheduling problems. In: Proc.

Conf. Mathematical Methods in Economy and Industry, Liberec, June 1–5 1998
62. Tikhonov AV, Arsenin VY (1977) Solutions of ill-posed problems. Wiley, New York
63. Ward J, Wendell RE (1990) Approaches to sensitivity analysis in linear programming. Ann Oper Res 27:3–38
64. Wendell RE (1997) Linear programming 3: The tolerance approach. In: Gal T, Greenberg HJ (eds) Advances in Sensitivity Analysis and Parametric Programming. Kluwer, Dordrecht
65. Zangwill WJ (1969) Nonlinear programming: A unified approach. Prentice-Hall, Englewood Cliffs
66. Zidaroiu C (1985) Regions of stability for random decision systems with complete connections. An Univ Bucuresti Mat 34:87–97
67. Zimmermann K (1989) Optimal choice of parameters in a machine scheduling problem. Ekonomicko-Matematicky Obzor 25:73–83
68. Zlobec S (1983 1986) Characterizing an optimal input in perturbed convex programming. Math Program 25:109–121, Corrigendum: Ibid 35:368–371
69. Zlobec S (1988) Characterizing optimality in mathematical programming models. Acta Applic Math 12:342–349
70. Zlobec S (1996) Lagrange duality in partly convex programming. In: Floudas CA, Pardalos PM (eds) State of the Art in Global Optimization. Kluwer, Dordrecht, pp 1–17
71. Zlobec S (1998) Stable parametric programming. Optim 45:387–416
72. Zlobec S (2000) Parametric programming: An illustrative mini-encyclopaedia. Math Comunications 5:1–39
73. Zlobec S, Ben-Israel A (1979) Perturbed convex programs: Continuity of the optimal solutions and optimal values. In: Oettli W, Steffens F (eds) Methods Oper. Res.: Proc. III Symp. Operat. Res., 31. Athenäum–Hain–Scriptor–Hanstein, pp 737–749
74. Zlobec S, Gardner R, Ben-Israel A (1982) Regions of stability for arbitrarily perturbed convex programs. In: Fiacco AV (ed) Mathematical Programming With Data Perturbations, Lecture Notes Pure and Applied Math. M. Dekker, New York, pp 69–89
75. Zowe J, Kurcyusz S (1979) Regularity and stability for the mathematical programming problem in Banach spaces. Applied Math and Optim 5:49–62

Nondifferentiable Optimization: Relaxation Methods

Ulf Bränndlund
Kungliga Tekniska Högskolan, Stockholm, Sweden

MSC2000: 49J52, 90C30

Article Outline

Keywords

Subgradient; Subdifferential; Convex function; Relaxation method; Lagrangian relaxation; Bundle methods

Relaxation methods for convex nondifferentiable optimization have their origin in relaxation methods for finding a solution to a system of linear inequalities, see [1] and [6]. In mathematical terms, one wants to find a vector $x \in \mathbf{R}^n$ such that $a_i^\top x - b_i \leq 0$ for $i = 1, \ldots, m$. In the simplest form, such a relaxation method iterates an initial guess $x_0 \in \mathbf{R}^n$ with the iteration scheme

$$x_{k+1} = x_k - \gamma^k \frac{a_{i_k}^\top x_k - b_{i_k}}{\|a_{i_k}\|_2^2} a_{i_k}, \tag{1}$$

where $i_k \in \operatorname{Argmax}_{i=1,\ldots,m} a_i^\top x_k - b_i$, and $\gamma^k \in [\delta, 2-\delta]$ and $\delta \in (0, 1)$. The iterations stop once a feasible solution is found. With $\gamma^k = 1$ the iteration formula (1) corresponds to a projection of x_k onto the most violated hyperplane at x_k.

The problem of finding a solution to an inequality system can be cast into a convex nondifferentiable optimization problem with known optimal value, namely the following

$$\min_x f(x) = \min_x \max \left\{ \max_{i=1,\ldots,m} a_i^\top x - b_i, 0 \right\},$$

which has optimal solution value equal to zero if the system is consistent.

The iteration scheme (1) can be generalized to convex nondifferentiable minimization with known optimal value in the following manner. Suppose that we want to find $x \in \mathbf{R}^n$ such that $f(x) \leq f_{\text{lev}}$, where $f_{\text{lev}} \geq f^* = \inf f(x)$ is known. Let $\mathfrak{m}_f$ denote the set of affine functions minorizing f,

$$\mathfrak{m}_f = \left\{ (a, b) \in \mathbb{R}^n \times \mathbb{R} : \begin{array}{c} a^\top y - b \leq f(y) \\ \text{for all } y \in \mathbb{R}^n \end{array} \right\}.$$

Assume as is customary in nondifferentiable optimization that we at $x \in \mathbf{R}^n$ can evaluate $f(x)$ and one arbitrarily chosen *subgradient*

$$g \in \partial f(x) = \left\{ g \in \mathbb{R}^n : \begin{array}{c} f(y) \geq f(x) + g^\top (y - x) \\ \text{for all } y \in \mathbb{R}^n \end{array} \right\} .$$

Then, $(g, g^\intercal x - f(x)) \in \mathfrak{m}_f$ and by definition of $\mathfrak{m}_f$,

$$f(x) \geq \sup_{(a,b)\in\mathfrak{m}_f} \{a^\top x - b\} \geq g^\top x - (g^\top x - f(x)) = f(x) .$$

Hence $f(x) = \sup_{(a,b)\in\mathfrak{m}_f}\{a^\top x - b\}$.

Thus, finding an x such that $f(x) \leq f_{\text{lev}}$ can be seen as solving an infinite system of inequalities, $a^\intercal x - b - f_{\text{lev}} \leq 0$ for all $(a, b) \in \mathfrak{m}_f$. The analog of (1) then becomes

$$x_{k+1} = x_k - \gamma^k \frac{f(x_k) - f_{\text{lev}}}{\|g_k\|_2^2} g_k , \tag{2}$$

where $g_k \in \partial f(x_k)$. It can be shown that if $X^* = \{x: f(x) \leq f_{\text{lev}} \neq \emptyset$ then $x_k \to x_{\text{lev}} \in X^*$.

This steplength rule has often in the literature been called the *Polyak II rule* perhaps since it appears as the second numbered equation in B.T. Polyak's classical paper [8], where its convergence properties are studied. It is shown there that, under certain mild conditions on f, the convergence rate is linear to an optimal solution, should it exist.

The step (2) corresponds to a projection onto a hyperplane. With $\gamma^k = 1$, x_{k+1} solves the quadratic programming problem

$$\begin{cases} \min & \|x - x_k\|_2^2 , \\ \text{s.t.} & f_{\text{lev}} \geq f(x_k) + g_k^T (x - x_k) . \end{cases} \tag{3}$$

One may in this framework add previously generated subgradients in order to obtain faster convergence, i. e. let x_{k+1} solve

$$\begin{cases} \min & \|x - x_k\|_2^2 , \\ \text{s.t} & f_{\text{lev}} \geq f(x_j) + g_j^T (x - x_j), \\ & j \in J^k , \end{cases} \tag{4}$$

where J^k is a subset of $\{1, \ldots, k\}$ including k.

Constraints on the Variables

The classical way to handle constraints on the variables, i. e. when x has to be in some closed and *convex set* X as well as in the set $\{x: f(x) \leq f_{\text{lev}}$, is to let

$$x_{k+1} = P_X \left(x_k - \gamma^k \frac{f(x_k) - f_{\text{lev}}}{\|g_k\|_2^2} g_k \right) , \tag{5}$$

where $P_X(y) = \text{argmin}_{x \in X} \|y - x\|_2^2$ denotes the *projection* of y onto the feasible set X.

However, it is often better to let x_{k+1} be the projection of x_k onto the set $X \cap \{x: \gamma^k (f_{\text{lev}} - f(x_k)) \geq g_k^T (x - x_k)\}$, i. e. for $\gamma^k = 1$ to add $x \in X$ to the constraint set of (3). If X consists of simple bounds or the unit-simplex then it can be shown that this latter way results in a new iteration point which is closer to the desired set $\{x \in X: f(x) \leq f_{\text{lev}}$ than does (5) (see [2]). In these two cases the computational burden of solving this one projection problem does not need to be larger, than solving the two very simple projections involved in (5).

Finding a Minimum Point

Suppose one wants to find a minimum point x^* of a convex function f. Then if the optimal value f^* is known a priori then the algorithms mentioned above can be successfully used with $f_{\text{lev}} = f^*$. But, prior knowledge of f^* is not usually the case. However, when solving the dual problem in applications of *Lagrangian relaxation* to obtain upper bounds on a primal maximization problem, it is often the case that a (good) lower bound, f_{low}, on f^* is known from a primal feasible solution. In these applications, iterations schemes as the ones above can be applied heuristically and often successfully by replacing f_{lev} in (2) by f_{low}.

To be specific, one such heuristic goes as follows. Suppose that at iteration k a lower bound f_{low}^k on f^* is known, which is used in place of f_{lev} in (2). Initially, $\gamma_0 = 2$ and at iteration k, γ_k is reduced by a constant factor $\alpha \in (0, 1)$, if there has been no decrease in terms of the function values for the last $K_{\max}$ iterations (which could indicate that too large steps are taken). Here α and $K_{\max}$ are parameters which need to be chosen appropriately for the application in question. Typical values are $\alpha = 0.5$ and $K_{\max} = 5$.

0 (Initialization)
Select a point $x_1 \in \mathbf{R}^n$ and $\delta^1 > 0$ and $\sigma_{\max} > 0$.
Set $\sigma_1 = 0$ and $f^0_{\text{rec}} = \infty$. Set $k = 1$, $l = 1$ and $k(1) = 1$.

1 (Function call)
Calculate $f(x_k)$ and $g_k \in \partial f(x_k)$.
IF $\| g_k \| = 0$
THEN terminate ($x_k \in$ Argmin f). Set $f^k_{\text{rec}} = \min\{f(x_k), f^{k-1}_{\text{rec}}\}$ and update x^k_{rec} correspondingly.

2 (Sufficient Descendent?)
If $f(x_k) \le f^{k(l)}_{\text{rec}} - \delta_l/2$, set $k(l+1) = k$, $\sigma_k = 0$, $\delta_{l+1} = \delta_l$, $l = l + 1$ and skip next step.

3 (Oscillation?)
If $\sigma_k > \sigma_{\max}$, set $k(l+1) = k$, $\sigma_k = 0$, $\delta_{l+1} = \delta_l/2$, replace x_k by x^k_{rec} and g_k correspondingly.

4 (New point)
Set $f^k_{\text{lev}} = f^{k(l)}_{\text{rec}} - \delta_l$,

$$x_{k+1} = x_k - \frac{f(x_k) - f^k_{\text{lev}}}{\| g_k \|_2^2} g_k.$$

5 (Path update)
Set $\sigma_k = \sigma_k + \| x_{k+1} - x_k \|$. Set $k = k + 1$ and return to Step 1.

A simple convergent method, [2] and [4], based on (2) for minimizing f with no assumption on prior knowledge of the optimal value is motivated as follows. It is a fact that if the set $\{x \in \mathbf{R}: f(x) \le f_{\text{lev}}\}$ is empty then the iteration scheme (2) generates a path whose length $\sum_{k=1}^{\infty} \|x_{k+1} - x_k\|$ is unbounded, and if the set is nonempty then the path length is bounded. If the former seems to be occurring then f_{lev} should be increased and if the algorithm seems to be converging to f_{lev} then a lower f_{lev} could be used. The mechanics of the algorithm should be clear from the following description. Let $f^k_{\text{rec}} = \min_{l=1,\ldots,k} f(x_l)$ denote the best function value found up to iteration k and let x^k_{rec} be the point at which this occurs. Let f^k_{lev} denote the 'level' which we are aiming for. The number $k(l)$ denotes the iteration of the lth change of f^k_{lev}. The number σ_k is the length of the path since the last update of f_{lev}.

Note that the level f^k_{lev} remains fixed in between the iterations $k(l)$ and $k(l+1) - 1$. For this algorithm it is possible to derive so called efficiency estimates. In particular, it can be shown that for 'small' $\epsilon > 0$ the algorithm produces $f^k_{\text{rec}} - f^* < \epsilon$ for $k \ge K/\epsilon^3$, where K is a positive constant.

Of course, one may in Step 4 use (4) instead of (2) to obtain a new iteration point. However, when using (4) in Step 4, more sophisticated schemes to adjust the aiming level f^k_{lev} are possible. A scheme suggested in [3] uses ideas from so called *proximal point bundle methods*, in which sufficient descent in terms of function values is enforced by means of so called null steps. Another method by C. Lemaréchal, A.S. Nemirovsky and Yu.E. Nesterov, [5], for the case when x is constrained to a compact set X, uses $f^k_{\text{lev}} = \alpha f^k_{\text{rec}} + (1-\alpha) f^k_{\text{low}}$, where f^k_{low} is the best lower bound possible, that is

$$f^k_{\text{low}} = \min_{x \in X} \max_{j=1,\ldots,k} f(x_j) + g_j^T (x - x_j).$$

This algorithm has a complexity estimate given by $f^k_{\text{rec}} - f^* < \epsilon$ for $k \ge K/\epsilon^2$, where K is a positive constant. Such a complexity estimate is optimal in the sense that it can not be improved uniformly with respect to the dimension by more than an absolute constant factor (for details, see [7]).

See also

- ▶ Dini and Hadamard Derivatives in Optimization
- ▶ Global Optimization: Envelope Representation
- ▶ Nondifferentiable Optimization
- ▶ Nondifferentiable Optimization: Cutting Plane Methods
- ▶ Nondifferentiable Optimization: Minimax Problems
- ▶ Nondifferentiable Optimization: Newton Method
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Nondifferentiable Optimization: Subgradient Optimization Methods

References

1. Agmon S (1954) The relaxation method for linear inequalities. Canad J Math 6:282–292
2. Brännlund U (1993) On relaxation methods for nonsmooth convex optimization. PhD Thesis, Kungliga Tekniska Högskolan, Stockholm
3. Brännlund U, Kiwiel KC, Lindberg PO (1995) A descent proximal level bundle method for convex nondifferentiable optimization. Oper Res Lett 17:121–126
4. Goffin J-L, Kiwiel KC (1999) Convergence of a simple subgradient level method. Math Program 85(1):207–211

5. Lemaréchal C, Nemirovskii AS, Nesterov YuE (1995) New variants of bundle methods. Math Program 69:111–147
6. Motzkin TS, Schoenberg IJ (1954) The relaxation method for linear inequalities. Canad J Math 6:393–404
7. Nemirovsky AS, Yudin DB (1983) Problem complexity and method efficiency in optimization. Ser Discret Math, Wiley, New York
8. Polyak BT (1969) Minimization of unsmooth functionals. USSR Comput Math Math Phys 9:14–29

Nondifferentiable Optimization: Subgradient Optimization Methods

Subgradient Methods

Ulf Brännlund
Kungliga Tekniska Högskolan, Stockholm, Sweden

MSC2000: 49J52, 90C30

Article Outline

Keywords

Subgradient; Subdifferential; Convex function

Subgradient methods for minimization of a *convex function* f: $\mathbf{R}^n \to \mathbf{R}$ over a closed *convex set* X have proven to be an efficient mean to solve large scale optimization problems. In particular, this is the case when X is a simple set, such as $\mathbf{R}^n$ or the positive orthant and when high accuracy of the solution is not required, e. g. in the context of Lagrangian relaxation of integer programming problems (cf. ► Integer programming: Lagrangian relaxation).

A *subgradient* at a point $x \in \mathbf{R}^n$ of a convex function f is a vector $g \in \mathbf{R}^n$ satisfying $f(y) \geq f(x) + g^\mathsf{T}(y - x)$ for all $y \in \mathbf{R}^n$. The set of subgradients at a point x is known as the *subdifferential* $\partial f(x)$. At points where the function is differentiable, the gradient is the sole member of the subdifferential.

Gradient based iterative methods, e. g. the steepest descent method designed for minimization of a smooth functions fail when one attempts to use their analogs, i. e. replacing gradients by subgradients, for minimization of convex nonsmooth functions. This is because the negative subgradient may not be a descent direction and even if it were at all points generated along the path, the sequence of iterates would not necessarily minimize f.

The basic subgradient algorithm takes the following form:

0 (Initialize)
Choose a point $x_0 \in \mathbf{R}^n$ and set $k = 0$.
1 (Function call)
Calculate $g_k \in \partial f(x_k)$.
IF $\| g_k \| = 0$ THEN terminate since $x_k \in \operatorname{Argmin} f$.
2 (New point)
Set
$$x_{k+1/2} = x_k - t_k \frac{g_k}{\|g_k\|},$$
and
$$x_{k+1} = \operatorname*{argmin}_{y \in X} \| y - x_{k+1/2} \|,$$
replace k by $k + 1$. Return to Step 1.

It can be shown, see [4], that if

$$t_k \downarrow 0 \quad \text{and} \quad \sum_{k=0}^{\infty} t_k = \infty, \tag{1}$$

then $\liminf f(x_k) = \inf_{x \in X} f(x) = f^*$ and furthermore, if also

$$\sum_{k=0}^{\infty} t_k^2 < \infty,$$

then x^k converges to a minimum point, if there is one (see [2]).

As can be expected, the rate of convergence for the above algorithm with the *divergent series rule* (1) is very slow. In fact, it can be shown that the algorithm can not have so-called geometric convergence rate. An algorithm is said to have *geometric convergence rate*, or *r-linear convergence rate*, if for any convex function and any starting point there exist M and $q \in (0, 1)$ such that $\|x_k - x^*\| \leq Mq^k$, where x^* is an optimal point. J.-L. Goffin has [3] shown that it is possible to obtain geometric convergence rate with the *geometric series rule*

$$t_k = M\rho^k \|g_k\|,$$

if M and $\rho \in (0, 1)$ are chosen large enough. In practice, though, for a particular f and starting point x_0 it is impossible to know what sufficiently large is.

Another often used steplength rule is the so-called *relaxation rule*, or *Polyak II rule*, where t^k is chosen according to

$$t_k = \gamma_k \frac{f(x_k) - f_{\text{lev}}^k}{\|g_k\|},$$

where f_{lev}^k is an estimate of the minimum value value f^* and $\gamma_k \in [\delta, 2 - \delta]$, and $\delta \in (0, 1)$. See also ► Nondifferentiable optimization: Relaxation methods.

Pure subgradient methods have a tendency to zigzag, i. e. a step in the direction $-g_k$ tends to be followed by a step which is almost parallel with g_k. Several solutions to overcome this behavior have been proposed. One such solution is the concept of *space dilation* of N.Z. Shor (see [5]). This approach is related to quasi-Newton methods for differentiable optimization. Another solution to the zig-zagging problem is the method proposed in [1], in which a step is taken in a direction which is a sum of the previous direction and the current subgradient. This approach is analogous to conjugate gradient methods for differentiable optimization.

See also

- ► Dini and Hadamard Derivatives in Optimization
- ► Global Optimization: Envelope Representation
- ► Nondifferentiable Optimization
- ► Nondifferentiable Optimization: Cutting Plane Methods
- ► Nondifferentiable Optimization: Minimax Problems
- ► Nondifferentiable Optimization: Newton Method
- ► Nondifferentiable Optimization: Parametric Programming
- ► Nondifferentiable Optimization: Relaxation Methods

References

1. Camerini PM, Fratta L, Maffioli F (1975) On improving relaxation methods by modified gradient techniques. Math Program Stud 3:79–97
2. Correa R, Lemaréchal C (1993) Convergence of some algorithms for convex minimization. Math Program 62(2):261–275
3. Goffin JL (1977) On convergence rates of subgradient optimization methods. Math Program 13:329–347
4. Polyak BT (1967) A general method for solving extremum problems. Soviet Math Dokl 8:593–597
5. Shor NZ (1970) Convergence rate of gradient descent method with dilation of the space. Cybernetics 6(2):102–108

Nonlinear Least Squares: Newton-type Methods

Chengxian Xu
Xian Jiaotong University, Xian, China

MSC2000: 49M37

Article Outline

Keywords

Newton method; Full-step Gauss–Newton method; Quasi-Newton methods; Adaptive methods; Global convergence; Local convergence rate

Nonlinear least squares problems can be phrased in terms of minimizing a real valued function that is a sum of some nonlinear functions of several variables. Efficient solution for unconstrained nonlinear least squares is important. Though some problems that arise in practical areas usually have constraints placed upon the variables and special techniques are required to handle these constraints, eventually the numerical techniques used rely upon the efficient solution of unconstrained nonlinear least squares problems.

The *unconstrained nonlinear least squares problems* have the form

$$\min f(x) = \frac{1}{2}\sum_{i=1}^{m}[r_i(x)]^2 = \frac{1}{2}r(x)^{\top}r(x),$$

where $r(x) = (r_1(x), \ldots, r_m(x))^{\top}$ and $r_i(x)$, $i = 1, \ldots, m$ are nonlinear functions of $x \in \mathbf{R}^n$. When $r(x)$, hence $f(x)$ is twice continuously differentiable, the gradient

and the Hessian matrix of $f(x)$ are given by

$$g(x) = \nabla f(x) = A(x)r(x),$$
$$G(x) = \nabla^2 f(x) = A(x)A(x)^\top + \sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x)$$

where $A(x) = [\nabla r_1(x) \cdots \nabla r_m(x)]$. The special structures of these derivatives had been exploited in developing effective solution methods for nonlinear least squares.

As a general unconstrained minimization problem, the *Newton method* plays a central role in the development of numerical methods for nonlinear least squares solution. Most commonly used nonlinear least squares methods can be viewed as variations on Newton's method. The Newton method for general optimization is derived based upon the quadratic model

$$q_k(\delta) = f_k + g_k^\top \delta + \frac{1}{2}\delta^\top G_k \delta,$$

where $f_k = f(x^{(k)})$, $g_k = g(x^{(k)})$, $G_k = G(x^{(k)})$, $\delta = x - x^{(k)}$, and $x^{(k)}$ is an approximation to a local minimizer x^* of the objective function $f(x)$. $q_k(\delta)$ is a local approximation to $f(x)$ at $x^{(k)}$ obtained from the *truncated Taylor approximation*. If this approximation is appropriate, then a presumably better approximation $x^{(k+1)} = x^{(k)} + \delta^{(k)}$ can be obtained by requiring that the step $\delta^{(k)}$ be a minimizer of $q_k(\delta)$. Thus the Newton method takes an initial approximation $x^{(1)}$ to x^* and attempts successively to improve the approximation through the iteration

- Solve the system $G_k\delta = -g_k$ for $\delta = \delta^{(k)}$.
- Set $x^{(k+1)} = x^{(k)} + \delta^{(k)}$.

If $G_k = M_k + C_k$ with $M_k = A_k A_k^\top$, $C_k = C(x^{(k)}) = \sum_{i=1}^{m} r_i(x^{(k)})\nabla^2 r_i(x^{(k)})$ is positive definite, the solution $\delta^{(k)}$ of the system is the global minimizer of $q_k(\delta)$, and if the starting point $x^{(1)}$ is sufficiently close to x^* at which $g(x^*) = 0$ and $G(x^*)$ is positive definite, the Newton method is well defined and converges at a quadratic rate.

Unfortunately, the basic Newton method as it stands is not suitable for a general purpose use since G_k may not be positive definite when $x^{(k)}$ is remote from x^* and even if G_k is positive definite, the convergence may not occur since $\{f_k\}$ may not decrease. Though both the possibilities can be eliminated by incorporating either *trust region technique* for the former case or *line search technique* for the later case, the main disadvantage of the Newton method is the demand for evaluation of second order derivatives of problem functions.

Since $r(x)$ is being minimized in the least squares sense, it may be the case that $r_i(x^*)$, $i = 1, \ldots, m$ are zero or very small. Thus when $x^{(k)}$ is close to x^*, compared with M_k, the second part C_k in G_k may be negligible. This suggests that M_k is a good approximation to G_k and gives the well known *full-step Gauss–Newton method*

- Solve the system $M_k\delta = -A_k r_k$ for $\delta = \delta^{(k)}$.
- Set $x^{(k+1)} = x^{(k)} + \delta^{(k)}$.

An important feature of the Gauss–Newton method is that the approximation to the Hessian G_k is directly obtained only from the first order derivatives of the problem functions. The approximation will be exact when the functions $r_i(x)$, $i = 1, \ldots, m$, are all either linear or zeros. Since the Gauss–Newton method is obtained from the Newton method by neglecting the part C_k of G_k, the convergence property of the method greatly depends on the size of the omitted part. If $C(x^*)$ is large relative to $M(x^*)$ in the sense $\eta_m(x^*) = \|M(x^*)^{-1} C(x^*)\| > 1$ which is regarded as *combined relative measure* of the nonlinearity and residual size of the problem (assuming $A(x^*)$ is full rank), then the method does not converge. If $\eta_m(x^*) \leq \theta < 1$ and the initial point $x^{(1)}$ is close to x^* enough, the Gauss–Newton method converges. In case convergence occurs, the rate of convergence also depends upon the size of $\eta_m(x^*)$. If $C(x^*) = 0$, the method is rather satisfactory in the sense that the method converges quadratically for zero residual problems, and if $C(x^*) \neq 0$, the convergence rate of the method is linear and the speed of convergence decreases as the relative nonlinearity or the residual size increases.

Since the matrix M_k is at worst positive semidefinite, the solution, denoted now by $s^{(k)}$, determined in the Gauss–Newton system is not uphill on $f(x)$. A possible modification to force the Gauss–Newton method to converge is to incorporate line search techniques. If the matrix A_k has full rank, then M_k is positive definite and the solution $s^{(k)}$ in the Gauss–Newton system is a descent direction of $f(x)$ at $x^{(k)}$. A line search along the direction $s^{(k)}$ determines a steplength α_k such that $f(x^{(k)}$

$+\alpha_k s^{(k)}) < f(x^{(k)})$ and a new approximation to x^* is then obtained as $x^{(k+1)} = x^{(k)} + \alpha_k s^{(k)}$. This modification was first suggested by H.O. Hartley [7] and is generally referred as *damped Gauss–Newton method*. This method does prevent divergence and usually have a larger domain of convergence than the full-step Gauss–Newton method. In fact convergence follows if the *condition number* $\mathcal{K}(A_k A_k^\top)$ is uniformly bounded above. Since A_k will usually be bounded above, this essentially requires that A_k does not loss rank in limit. Unfortunately this can happen [11] and convergence to a noncritical point can occur. Also when A_k is rank deficient, the solution $s^{(k)}$ of the Gauss–Newton system becomes numerically orthogonal to g_k at some distance from a local minimizer x^* and no further progress can be made by line searches.

A further modification to the full-step Gauss–Newton method is to incorporate the trust region technique. In this modification, the trust region subproblem

$$\begin{cases} \min & q_k(\delta) = f_k + r_k^\top A_k^\top \delta + \frac{1}{2}\delta^\top M_k \delta \\ \text{s.t.} & \|\delta\| \le \Delta_k \end{cases}$$

is solved for $\delta^{(k)}$ with properly adjusted radius Δ_k and the new point is obtained as $x^{(k+1)} = x^{(k)} + \delta^{(k)}$. The trust region radius is adjusted in such a way that the model function $q_k(\delta)$ is believed to have adequately approximated the function $f(x)$ in the region $\| \delta \| \le \Delta_k$. This modification with a different form was first suggested by K. Levenberg [8] and D.W. Marquardt [9]. J.J. Moré used the modification in the above form [10].

An aspect of the Gauss–Newton method is the efficient solution of the Gauss–Newton system. It must be emphasized here that the solution of the system can not be done by first forming the product $A_k A_k^\top$ and then performing a factorization on the product, because this will worse the conditioning of the system and lead to substantial loss in precision. A stable and efficient way is to factorize the augmented matrix $[A_k^\top \; r_k]$ using either *Householder transformation* or *Given's transformation*.

Regarded as a general optimization problem, another way to obtain approximations to the Hessian matrix G_k from first order derivative information of problem functions is to use *quasi-Newton updates* and the resulting Newton type methods are called *quasi-Newton methods*. Let B_k denote an approximation to the Hessian G_k in the quadratic model $q_k(\delta)$. Any Newton type method with line searches has the following basic framework.

- Solve the system $B_k s = -g_k$ for a descent direction $s^{(k)}$.
- Determine a steplength α_k along the direction $s^{(k)}$ by line searches.
- Set $x^{(k+1)} = x^{(k)} + \alpha_k s^{(k)}$.

In order to ensure *global convergence* and to have a rapid *local convergence rate*, quasi-Newton updates require the matrix B_k to satisfy the following conditions:

- since $G_k \delta_k \approx \gamma_k$, B_k should satisfy the so-called quasi-Newton equation

  $$B_{k+1}\delta^{(k)} = \gamma^{(k)},$$

  where

  $$\delta^{(k)} = x^{(k+1)} - x^{(k)}, \gamma^{(k)} = g_{k+1} - g_k.$$

- B_k is symmetric.
- B_{k+1} is effectively obtained from B_k using lower rank updating

  $$B_{k+1} = B_k + E_k$$

  so that the calculation of B_{k+1} is less expensive.
- B_k is positive definite so that the solution $s^{(k)}$ obtained as the minimizer of $q_k(\delta)$ is a descent direction of $f(x)$ at $x^{(k)}$.

There exist numerous updating formulas to achieve these conditions and the *Broyden family* with single parameter

$$B_{k+1}(\theta) = B_k - \frac{B_k \delta^{(k)} {\delta^{(k)}}^\top B_k}{{\delta^{(k)}}^\top B_k \delta^{(k)}} + \frac{\gamma^{(k)} {\gamma^{(k)}}^\top}{{\delta^{(k)}}^\top \gamma^{(k)}} + \theta w^{(k)} {w^{(k)}}^\top$$

is the most important, where

$$w^{(k)} = ({\delta^{(k)}}^\top B_k \delta^{(k)})^{1/2} \times \left[\frac{\gamma^{(k)}}{{\delta^{(k)}}^\top \gamma^{(k)}} - \frac{B_k \delta^{(k)}}{{\delta^{(k)}}^\top B_k \delta^{(k)}} \right]$$

and θ is a parameter [3]. If ${\delta^{(k)}}^\top \gamma^{(k)} > 0$ for all k, then $B_{k+1}(\theta)$ preserves positive definiteness for all values

$$\theta > \overline{\theta} = \frac{1}{1 - a_k b_k} (< 0),$$

where

$$a_k = \frac{{\gamma^{(k)}}^\top H_k \gamma^{(k)}}{{\delta^{(k)}}^\top \gamma^{(k)}}, \quad b_k = \frac{{\delta^{(k)}}^\top B_k \delta^{(k)}}{{\delta^{(k)}}^\top \gamma^{(k)}}$$

and $H_k = B_k^{-1}$. The $\overline{\theta}$ is the value which causes $B_{k+1}(\theta)$ to be singular. The condition ${\delta^{(k)}}^\top \gamma^{(k)} > 0$ is realistic and can always be achieved when the steplength α_k either is determined from exact line search or satisfies the *Goldstein conditions*. The Broyden family includes the famous BFGS ($\theta = 0$) formula

$$B_{k+1}^{\mathrm{BFGS}} = B_k + \frac{\gamma^{(k)} {\gamma^{(k)}}^\top}{{\delta^{(k)}}^\top \gamma^{(k)}} - \frac{B_k \delta^{(k)} {\delta^{(k)}}^\top B_k}{{\delta^{(k)}}^\top B_k \delta^{(k)}},$$

DFP ($\theta = 1$) formula

$$B_{k+1}^{\mathrm{DFP}} = B_k + \left(1 + \frac{{\delta^{(k)}}^\top B_k \delta^{(k)}}{{\delta^{(k)}}^\top \gamma^{(k)}}\right) \times \frac{\gamma^{(k)} {\gamma^{(k)}}^\top}{{\delta^{(k)}}^\top \gamma^{(k)}} - \frac{\gamma^{(k)} {\delta^{(k)}}^\top B_k + B_k \delta^{(k)} {\gamma^{(k)}}^\top}{{\delta^{(k)}}^\top \gamma^{(k)}}$$

and the *selfdual rank one formula* ($\theta = 1/(1-b_k)$)

$$B_{k+1} = B_k + \frac{y_k y_k^\top}{{\delta^{(k)}}^\top y_k}, \quad y_k = \gamma^{(k)} - B_k \delta^{(k)}.$$

Both the DFP and BFGS methods was found to work well in practice and have been widely used. These two methods have a number of important properties as follows:

1) For quadratic functions (with exact line searches)
   - terminate in at most n iterations with ($B_{n+1} = G$);
   - preserve the *hereditary property*

$$B_i \delta^{(j)} = \gamma^{(j)}, \quad j = 1, \ldots, i-1;$$

   - generate conjugate directions, and conjugate gradients (when $B_1 = I$)
2) For general functions
   - preserve positive definite B_k;
   - global convergence for strictly convex functions (with exact line searches);
   - local superlinear convergence rate.

The BFGS method is even better than the DFP method and has usually been used with low accuracy line searches. In fact, global convergence and superlinear convergence rate of the BFGS method with inexact line searches have been proved [12]. The Broyden family is important in that many of the properties of the BFGS and DFP formulas are common to whole family. L.C.W. Dixon showed that when applied to any continuously differentiable function, all Broyden methods generate the same sequence $\{x^{(k)}\}$ from the same starting point, assuming that the multiple local minima in line search are resolved consistently, degenerate values of θ are avoided and the algorithm is well-defined.

When any above quasi-Newton updating formula is used in a method, the system $B_k s = -g_k$ needs solved to get the direction $s^{(k)}$ and this needs $O(n^3)$ arithmetic operations. In early versions of quasi-Newton methods, the search direction $s^{(k)}$ is obtained as a product of a matrix and a vector from

$$s^{(k)} = -H_k g_k$$

and the matrix H_k is an inverse approximation to G_k ($H_k \approx G_k^{-1}$ obtained from an *inverse quasi-Newton updating* formula. This avoids the solution of the system and only requires $O(n^2)$ arithmetic operations. The inverse updating formulas can be similarly derived from the quasi-Newton equation

$$H_{k+1} \gamma^{(k)} = \delta^{(k)}.$$

or can be derived from above updating formulas using the *Sherman–Morrison formula*. For example, the Broyden family of inverse updatings is given by

$$H_{k+1}(\phi) = H_k - \frac{H_k \gamma^{(k)} {\gamma^{(k)}}^\top H_k}{{\gamma^{(k)}}^\top H_k \gamma^{(k)}} + \frac{\delta^{(k)} {\delta^{(k)}}^\top}{{\delta^{(k)}}^\top \gamma^{(k)}} + \phi v^{(k)} {v^{(k)}}^\top,$$

where

$$v^{(k)} = \left(\gamma^{(k)\top} H_k \gamma^{(k)}\right)^{1/2} \times \left[\frac{\delta^{(k)}}{\delta^{(k)\top}\gamma^{(k)}} - \frac{H_k\gamma^{(k)}}{\gamma^{(k)\top} H_k \gamma^{(k)}}\right]$$

and ϕ is a parameter related to θ by

$$\phi = \frac{1-\theta}{1+\theta(a_k b_k - 1)}.$$

The corresponding inverse BFGS, DFP and rank one updating formulas can be obtained by setting $\phi = 1$, $\phi = 0$ and $\phi = 1/(1 - a_k)$. It can be seen that one kind of updating formulas is obtained from another kind by simply making interchanges $B_k \leftrightarrow H_k$ and $\delta^{(k)} \leftrightarrow \gamma^{(k)}$. Although, the matrix H_k preserves the positive definiteness in theory when $\delta^{(k)\top}\gamma^{(k)} > 0$, the loss of the positive definiteness may occur in practical calculations due to round-off errors and it can not be easily recognized whether the matrix H_k is positive definite or not. Now the widely used versions of quasi-Newton methods is to represent the matrix B_k in a factorized form $L_k D_k L_k^\top$ and to update the factors in $O(n^2)$ operations to obtain the factors $L_{k+1}D_{k+1}L_{k+1}^\top$ of the matrix B_{k+1} where L is a unit lower triangular matrix and D is a diagonal matrix with positive diagonals [5]. The solution of the system $B_k s = -g_k$ is then obtained in $O(n^2)$ operations by using forward and backward substitutions. The positive definiteness of the matrix B_k is indicated by the diagonal elements of D_k and can be maintained by controlling round-off errors. Methods with quasi-Newton updating in $LDL^\intercal$ form preserve the convergence properties of quasi-Newton methods and reduce the arithmetic operations at each iteration.

When quasi-Newton updating formulas are used to generate approximations to the Hessian matrices or their inverses for nonlinear least squares problems, $A_1A_1^\top$ can be selected as the initial matrix B_1 if $A_1A_1^\top$ is positive definite. One more modification to the quasi-Newton updatings for nonlinear least squares is the definition of the vector $\gamma^{(k)}$. Let $\gamma_o^{(k)}$ denote previous defined vector $\gamma^{(k)}$. From the Taylor expansion of $g(x)$, a new definition of $\gamma^{(k)}$

$$\gamma_n^{(k)} = M_{k+1}\delta^{(k)} + (A_{k+1} - A_k)r_{k+1}$$

can be used to replace $\gamma_o^{(k)}$ in any quasi-Newton updating formula. This idea is essentially suggested by M.C. Bartholomew-Biggs [2]. However, the condition $\delta^{(k)\top}\gamma_n^{(k)} > 0$, required for maintaining the positive definite approximations, may not be guaranteed by line searches. This difficulty can be avoided by using a safeguarded value

$$\max\left\{\delta^{(k)\top}\gamma_n^{(k)}, 0.01\delta^{(k)\top}\gamma_o^{(k)}\right\}$$

in place of $\delta^{(k)\top}\gamma^{(k)}$ in updating formulas [1].

For nonlinear least squares problems, both the quasi-Newton and Gauss–Newton methods generate approximations to Hessian matrices of objective functions only from their first order derivative information. The convergence rate is superlinear when any quasi-Newton, such as BFGS method is used to any differentiable nonlinear least squares problem, but the special structure of the problem functions is not taken into account and many iterations are required to build up a satisfactory approximation to the Hessian. The Gauss–Newton method takes the advantage of the special form of nonlinear least squares and better approximations to Hessian matrices are directly obtained from the Jacobian matrix $A(x)$ of $r(x)$ for zero and small residual problems. The damped Gauss-Newton method converges at a quadratic rate for zero residual problems and at a fast linear rate for small residual problems, which in limited precision may be preferable to superlinear convergent methods. However, the method converges slowly for large residual problems, even fails on singular problems. Hence, the damped Gauss–Newton method is generally preferred for zero and small residual problems, but should be avoided for large residual and singular problems. Attempts have been made to combine the best features of both kind of methods. These include *adaptive methods* [4,13], *hybrid methods* [1,6], and *factorized quasi-Newton methods* [14,15].

In adaptive methods approximations to Hessian $G(x)$ are obtained by approximating the nonlinear term C_k in G_k by a symmetric matrix S_k and keeping the Gauss–Newton matrix M_k unchanged, that is, to define

$$B_k = M_k + S_k$$

and to develop updating formulas for the matrix S_k. This idea was first suggested by K.M. Brown and J.E. Dennis in 1973. In their method $S_k = \sum_{i=1}^m r_i(x^{(k)})$ $S_i^{(k)}$ and $S_i^{(k)}$ approximates $\nabla^2 r_i(x^{(k)})$ obtained from $S_i^{(k-1)}$ using certain quasi-Newton updating formula. In

methods of C.G. Broyden and Dennis (1973), J.T. Bett (1976), Bartholomew-Biggs (1977), P.E. Gill and W. Murray (1978), and Dennis, D.M. Gay and R.E. Welsch (1981) the formulas for updating S_k are defined by using certain quasi-Newton-like updating formulas, for example, Gill and Murray used the BFGS-like formula

$$S_{k+1} = S_k + \frac{\gamma^{(k)}\gamma^{(k)\top}}{\delta^{(k)\top}\gamma^{(k)}} - \frac{W_k\delta^{(k)}\delta^{(k)\top}W_k}{\delta^{(k)\top}W_k\delta^{(k)}},$$

where $W_k = M_{k+1} + S_k$, while Dennis, Gay and Welsch developed an updating formula for S_k:

$$S_{k+1} = S_k - \frac{\eta^{(k)}\gamma^{(k)\top} + \gamma^{(k)}\eta^{(k)\top}}{\delta^{(k)\top}\gamma^{(k)}} + \frac{\eta^{(k)\top}\delta^{(k)}}{(\delta^{(k)\top}\gamma^{(k)})^2}\gamma^{(k)}\gamma^{(k)\top},$$

where $\eta^{(k)} = S_k\delta^{(k)} - y^{(k)}$ and $y^{(k)} = (A_{k+1} - A_k)\, r_{k+1}$. These methods attempt to get better approximations to G_k from first order derivative information by using the special structure of nonlinear least squares. However, since the positive definiteness of the resulting approximation $B_k = M_k + S_k$ can not be guaranteed, these methods must be incorporated with trust region techniques. This increases the complexity of methods. Theoretical analysis and practical calculations show that the Dennis–Gay–Welsch adaptive method is superlinearly convergent and effective.

At each iteration of a hybrid method, the approximation B_k is simply chosen either the Gauss–Newton matrix M_k or a quasi-Newton matrix, for example BFGS matrix, by defining a test T_k, that is,

$$B_{k+1} = \begin{cases} \text{BFGS}(B_k, \delta^{(k)}, \gamma^{(k)}) & \text{if } T_k \text{ holds,} \\ M_{k+1} & \text{otherwise,} \end{cases}$$

where BFGS(B_k, $\delta^{(k)}$, $\gamma^{(k)}$) denotes the BFGS updating formula. Thus each step of a hybrid method is either a Gauss–Newton step or a BFGS step. In developing a hybrid method, a test must be derived to distinguish between these two steps. A reasonable test should have the capability to differentiate problems so that the method ultimately takes the damped Gauss–Newton method for zero and small residual problems or the BFGS method for large residual and singular problems. M. Al-Baali and R. Fletcher [1] proposed a test

$$\text{T}_\text{k}: \ \Delta(B_k, \delta^{(k)}, \gamma^{(k)}) \le \Delta(M_{k+1}, \delta^{(k)}, \gamma^{(k)}),$$

based on the quantity

$$\Delta(A, \delta^{(k)}, \gamma^{(k)}) = \left(a_k^2 - 2\frac{1}{b_k} + 1\right)^{1/2},$$

which is regarded as a measure of the approximation error of the matrix A to G_{k+1}, where

$$a_k = \frac{\gamma^{(k)\top}A^{-1}\gamma^{(k)}}{\delta^{(k)\top}\gamma^{(k)}}, \quad b_k = \frac{\delta^{(k)\top}A\delta^{(k)}}{\delta^{(k)\top}\gamma^{(k)}}.$$

Though the resulting hybrid method is robust and effective, the test needs extra $O(n^2)$ arithmetic operations at each iteration and the resulting method is only linearly convergent [6]. Fletcher and C.X. Xu [6] proposed a simple test

$$\text{T}_\text{k}: \ \frac{f_k - f_{k+1}}{f_k} \le \epsilon,$$

where $\epsilon \in (0, 1)$ is a preset parameter. Numerical experiences show that the method is not sensitive to the choice of the value ϵ, but the value $\epsilon = 0.2$ is recommended. The test is simple and theoretical analysis shows that the method ultimately takes the Gauss–Newton method for zero residual problems and the BFGS method for nonzero residual problems. Therefore, the method maintains the convergence properties of BFGS method and combines the best features of both the Gauss–Newton and BFGS methods. Practical calculations show that the later hybrid method is better than the Fletcher–Al-Baali hybrid method.

Factorized quasi-Newton methods for nonlinear least squares take the approximations B_k in the form

$$B_k = (A_k + L_k)(A_k + L_k)^\top$$

and develop updating formulas for the matrix L_k such that $L_kL_k^\top + A_kL_k^\top + L_kA_k^\top$ approximates the second part C_k of G_k. If the matrix $(A_k + L_k)$ is of full rank, the search direction $s^{(k)}$ obtained from the system $B_ks = -g_k$ is guaranteed to be descent. The updating formulas for L_k can similarly derived from the quasi-Newton equation

$$(A_{k+1} + L_{k+1})(A_{k+1} + L_{k+1})^\top\delta^{(k)} = \gamma^{(k)}.$$

Using the theory of *generalized inverses* of matrices, Xu, X.F. Ma and M.Y. Kong derived a class of updating for-

mulas for L_k

$$L_{k+1} = L_k + (a-d)\frac{V_k \delta^{(k)} \gamma^{(k)\top}}{\delta^{(k)\top}\gamma^{(k)}} + b\frac{V_k W_k^{-1}\gamma^{(k)}\gamma^{(k)\top}}{\delta^{(k)\top}\gamma^{(k)}} - c\frac{V_k\delta^{(k)}\delta^{(k)\top}W_k}{\delta^{(k)\top}W_k\delta^{(k)}},$$

where

$$V_k = A_{k+1} + L_k, \quad W_k = V_k V_k^\top$$

a, b, c and d are parameters satisfying the following equations:

$$\frac{\delta^{(k)\top}W_k\delta^{(k)}}{\delta^{(k)\top}\gamma^{(k)}}a^2 + 2ab + \frac{\gamma^{(k)\top}W_k^{-1}\gamma^{(k)}}{\delta^{(k)\top}\gamma^{(k)}} = 1,$$
$$\frac{ad}{\delta^{(k)\top}\gamma^{(k)}} = \frac{bc}{\delta^{(k)\top}W_k\delta^{(k)}},$$
$$c + d = 1.$$

The resulting approximation $B_{k+1} = (A_{k+1} + L_{k+1})(A_{k+1} + L_{k+1})^\top$ are a class of Broyden-like updating formulas

$$B_{k+1} = \alpha B_{k+1}^{\mathrm{BFGS}} + \beta B_{k+1}^{\mathrm{DFP}},$$

where $\alpha = c^2 + 2cd$, $\beta = d^2$ and $\alpha + \beta = (c+d)^2 = 1$, B_{k+1}^{BFGS} and B_{k+1}^{DFP} are in previous forms with B_k being replaced by W_k. Both these formulas for BFGS-like and DFP-like are first obtained by H. Yabe and T. Takahashi [15] from their proposed updating formulas for L_{k+1} which can be obtained in the above updating formulas of L_{k+1} by setting $c = 0$, $d = 1$, $b = (\delta^{(k)\top}\gamma^{(k)}/\gamma^{(k)\top}W_k^{-1}\gamma^{(k)})^{1/2}$ for BFGS-like and $c = 1$, $d = 0$, $a = (\delta^{(k)\top}\gamma^{(k)}/\delta^{(k)\top}W_k\delta^{(k)})^{1/2}$ for DFP-like, respectively.

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- ABS Algorithms for Optimization
- Automatic Differentiation: Calculation of Newton Steps
- Conjugate-Gradient Methods
- Contraction-mapping
- Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- Gauss–Newton Method: Least Squares, Relation to Newton's Method
- Generalized Total Least Squares
- Global Optimization Methods for Systems of Nonlinear Equations
- Interval Analysis: Systems of Nonlinear Equations
- Interval Newton Methods
- Large Scale Trust Region Problems
- Least Squares Orthogonal Polynomials
- Least Squares Problems
- Local Attractors for Gradient-related Descent Iterations
- Nondifferentiable Optimization: Newton Method
- Nonlinear Least Squares Problems
- Nonlinear Least Squares: Trust Region Methods
- Nonlinear Systems of Equations: Application to the Enclosure of All Azeotropes
- Unconstrained Nonlinear Optimization: Newton–Cauchy Framework

References

1. Al-Baali M, Fletcher R (1985) Variational methods for non-linear least squares. J Oper Res Soc 36:405–421
2. Bartholomew-Biggs MC (1977) The estimation of the Hessian matrix in nonlinear least squares problems with non-zero residuals. Math Program 12:67–80
3. Broyden CG (1965) A class of methods for solving nonlinear simultaneous equations. Math Comput 19:577–593
4. Dennis JE, Gay DM, Welsch RE (1981) An adaptive nonlinear least-squares algorithm. TOMS 7:348–368
5. Fletcher R, Powell MJD (1974) On the modification of LDL$^\mathsf{T}$ factorization. Math Comput 28:1067–1087
6. Fletcher R, Xu CX (1987) Hybrid methods for nonlinear least squares. IMA J Numer Anal 7:371–389
7. Hartley HO (1961) The modified Gauss–Newton method for the fitting of nonlinear regression function by least squares. Technometrics 3:269–280
8. Levenberg K (1944) A method for the solution of certain nonlinear problems in least squares. Quart Appl Math 2:164–168
9. Marquardt DW (1963) An algorithm for least squares estimation of nonlinear parameters. SIAM J 11:431–441
10. Moré JJ (1977) The Levenberg–Marquardt algorithm: implementation and theory. In: Watson GA (ed) Numerical Analysis: Dundee. Lecture Notes Math. Springer, Berlin
11. Powell MJD (1970) A hybrid method for nonlinear equations. In: Rabinowitz P (ed) Numerical Methods for Nonlinear Algebraic Equations. Gordon and Breach, New York
12. Powell MJD (1975) Some global convergence properties of a variable metric algorithm for minimization without exact line searches. AERE Harwell Report CSS15
13. Xu CX (1987) Hybrid methods for nonlinear least squares and related problems. PhD Thesis, Univ. Dundee

14. Xu CX, Ma XF, Kong MY (1996) A class of factorized quasi-Newton methods for nonlinear least squares problems. J Comput Math 14:143–158
15. Yabe H, Takahashi T (1991) Factorized quasi-Newton methods for nonlinear least squares problems. Math Program 51:75–100

Nonlinear Least Squares Problems

NLS

Chengxian Xu
Xian Jiaotong University, Xian, China

MSC2000: 90C30

Article Outline

Keywords

Nonlinear least squares; Residuals; Kuhn–Tucker optimality condition; Descent method; Line search; Trust region

Nonlinear least squares problems are among the most commonly occurring and important applications of optimization techniques. The problem is to find minima of a real valued function that has the form of a sum of some nonlinear functions of several independent variables

$$\min f(x) = \frac{1}{2}\sum_{i=1}^{m}[r_i(x)]^2 = \frac{1}{2}r(x)^\top r(x)\,,$$

where $r_i(x)$, $i = 1, \ldots, m$, are m nonlinear functions defined on $\mathbf{R}^n$, $r(x)$ is the vector representation of $r_i(x)$, $i = 1, \ldots, m$, $m \geq n$, and the real number 1/2 is generally placed for convenience.

Approaches for least squares can be traced back to more than two hundred years ago. It is well-known that C.F. Gauss proposed the method, called *Gauss–Newton method*, in 1809 to estimate motion orbits of planets from observation data. However, early in the 1750s some methods had been suggested by P.S. Laplace, L. Euler, etc. to deal with measuring data in astronomical observations. In 1805, A.M. Legendre proposed a method to determine the orbit of comets and the meridian of the earth. He called it least squares method, but did not demonstrate its optimality in theory. The advent of computer promoted the development and applications of numerical analysis including optimization methods and nonlinear least squares solutions.

Nonlinear least squares problems arise in various practical areas such as scientific computing, scientific experiments, engineering designs, survey and observations, geological prospecting, physical science, mathematics and so on. It is particularly useful in data processing and error estimations. Here are a few examples to illustrate the applications of nonlinear least squares problems.

Solution of Simultaneous Equations

It is frequently concerned with in scientific computing to find a solution of a system of nonlinear equations

$$\begin{aligned} &f_1(x_1, \ldots, x_n) = 0\,,\\ &\cdots \quad \cdots\\ &f_m(x_1, \ldots, x_n) = 0\,, \end{aligned}$$

where $x_1, \ldots, x_n$ are unknowns. The system is *underdetermined* if $m < n$, *well-determined* if $m = n$ and *overdetermined* if $m > n$. It is usually not possible to obtain an exact solution for an overdetermined system and one possibility is to seek a best least squares solution, that is, to find x^* such that the function

$$f(x) = \frac{1}{2}\sum_{i=1}^{m}[f_i(x)]^2$$

is minimized at x^*.

Curve (Data) Fitting

Perhaps the most frequently solved of all nonlinear least squares problems are data fitting problems. In scientific researches, for example chemical or physical experiments, it is often encountered that the dependence of some observable quantity y on some independent variable(s) t is predicted, based on theoretical ground, to

have the form

$$y = h(x, t)$$

and m values $y_1, \ldots, y_m$ are measured for points $t_1, \ldots, t_m$ where $x \in \mathbf{R}^n$ is an adjustable parameter vector. We are required to choose an optimal parameter vector x^* such that the function $h(x, t)$ best fits the data in the least squares sense, that is, define the function $f(x)$ by

$$f(x) = \frac{1}{2}\sum_{i=1}^{m}[h(x, t_i) - y_i]^2 = \frac{1}{2}r(x)^\top r(x)\,,$$

and x^* is found by minimizing the function $f(x)$ where $r_i(x) = h(x, t_i) - y_i$, $i = 1, \ldots, m$ are called *residuals*.

In practice, both the values $y_1, \ldots, y_m$ and $t_1, \ldots, t_m$ are usually subject to measured errors. In the above conventional approach, implicit assumptions are made that only the values $y_1, \ldots, y_m$ are subject to measured errors and that the values $t_1, \ldots, t_m$ are either exact or contain negligible errors. In many applications, however, this is an oversimplification and the use of the above nonlinear least squares problems may lead to bias in the estimated parameters and variance values. It is then necessary to take proper account of errors in all variable values. The function of the resulting problem has the form

$$\begin{aligned} f(x, \tau) &= \frac{1}{2}\sum_{i}^{m}[(h(x, \tau_i) - y_i)^2 + (\tau_i - t_i)^2] \\ &= \frac{1}{2}[r(x, \tau)^\top r(x, \tau) + e(\tau)^\top e(\tau)]\,, \end{aligned}$$

where $\tau = (\tau_1, \ldots, \tau_m)^\mathsf{T}$ and $r(x, \tau)$, $e(\tau)$ are m-vectors with components $r_i(x, \tau) = h(x, \tau_i) - y_i$ and $e_i(\tau) = \tau_i - t_i$, $i = 1, \ldots, m$, respectively. Taking errors in all variables into account increases the complexity of the problem. For example, the simplest linear problem $h(x, t) = x_1 + x_2 t$ is no longer a linear problem when errors in all variables are taken into account.

Optimal Designs

Let us consider the optimal design of a coaxial cable circuit. Let $x_1, \ldots, x_n$ be the design parameters. For a given circuit, its performance index is a function of $x_1, \ldots, x_n$ and ω

$$h(x_1, \ldots, x_n, \omega)\,,$$

where ω is the frequency of the circuit. The aim of the circuit optimal design is to determine the circuit parameters $x_1, \ldots, x_n$ such that the performance index of the circuit best approximates a given characteristic function $\psi(\omega)$ in a given interval $[\alpha, \beta]$ of the frequency ω. This can be expressed as to find $x^* \in \mathbf{R}^n$ which minimizes the function

$$\begin{aligned} f(x) &= \frac{1}{2}\sum_{i=1}^{m}[r_i(x)]^2 \\ &= \frac{1}{2}\sum_{i=1}^{m}[h(x_1, \ldots, x_n, \omega_i) - \psi(\omega_i)]^2 \end{aligned}$$

where $\alpha \le \omega_1 < \cdots < \omega_m \le \beta$. Meanwhile the circuit parameters are generally subject to some restricted factors of supplied cables such as geometric shapes, diameters, medium materials and so on. These subjects can be expressed as constrained conditions in the form

$$c_j(x_1, \ldots, x_n) \ge 0, \quad j = 1, \ldots, p\,.$$

Therefore, the mathematical model of the optimal design of a circuit generally has the form

$$\begin{cases} \min & f(x) = \frac{1}{2}\sum_{i=1}^{m}[r_i(x)]^2 \\ \text{s.t.} & c_j(x) \ge 0, \quad j = 1, \ldots, p\,. \end{cases}$$

Nonlinear least squares problems can be classified into unconstrained and constrained ones depending on whether there exist constraints on variables $x \in \mathbf{R}^n$. For example, the resulting problem in the optimal design of the coaxial cable circuit is a constrained nonlinear least squares problem while the problem formed in the solution of simultaneous equations is an unconstrained nonlinear least squares problem. In some situations, the solution of a unconstrained nonlinear least squares problem may be unacceptable. To avoid such a situation, some restrictions on parameter vector x can be imposed in the form of constraints. For example, some variables are required to be nonnegative, some may be bounded by lower and/or upper bound(s), and some dependent relationships among variables must be satisfied. This also results in constrained nonlinear least squares problem. When errors in all variables are taken into account, the resulting problem is generally called a *generalized nonlinear least squares problem*. This increases not only the problem complexity, but also the

problem dimension. An advantage of the generalized nonlinear least squares problems that can be exploited is that the problem variables are *separable*. An optimization problem is called separable if the optimization with respect to some of the variables is easier than with respect to the others. Of course, there are other separable nonlinear least squares problems such as arose in curve fitting, component analysis and orthogonal regression (see [14]).

When $r(x)$, and hence $f(x)$ is twice continuously differentiable, the gradient and the Hessian matrix of $f(x)$ are given by

$$\nabla f(x) = A(x)r(x),$$

$$\nabla^2 f(x) = A(x)A(x)^\top + \sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x)\,,$$

where $A(x) = [\nabla r_1(x), \ldots, \nabla r_m(x)]$. Let x^* be a minimizer of a NLS problem. The problem is called a *zero residual problem* if $r(x^*) = 0$ and hence $f(x^*) = 0$ and a *nonzero residual problem* if $r(x^*) \neq 0$. A nonzero residual problem is called *small residual* if $r(x^*)$ is small or the second part of $\nabla^2 f(x^*)$ is relatively small compared with the first part of $\nabla^2 f(x^*)$, otherwise it is called *large residual*.

Methods for nonlinear least squares are iterative type and are based upon trying to find a point x^* at which the so-called *Kuhn–Tucker optimality condition* is satisfied. These methods are generally of *descent* type. From a given initial point, these methods generate a sequence $\{x^{(k)}\}$ such that either one point of the sequence or any accumulation point of $\{x^{(k)}\}$ is a *Kuhn–Tucker point* (referred to as a *KT point*). The typical behavior of a method is that if the iteration is not terminated at some point $x^{(k)}$, the values $\phi(x^{(k)})$ of a *merit function* for the problem are monotonically decreased so that the iterates $x^{(k)}$ move steadily towards a neighborhood of a KT point x^*, and then converges rapidly to the point x^*. The basic structure of the kth iteration of such a method has the form

- Check if $x^{(k)}$ is a KT point.
- If $x^{(k)}$ is not a KT point, determine a $\delta^{(k)}$ such that

$$\phi(x^{(k)} + \delta^{(k)}) < \phi(x^{(k)}).$$

- $x^{(k+1)} = x^{(k)} + \delta^{(k)}$.

The information of second order derivatives of problem functions are required to determine if a KT point is a local minimizer. Since the evaluation of second order derivatives are time consuming and sometimes the second order derivatives of functions are not available, most nonlinear least squares methods do not evaluate second order derivatives and just try to locate a KT point. A KT point may not be a local minimizer. However, there exist other features of methods such as the monotonic decreasing property of the sequence $\{\phi(x^{(k)})\}$, which usually imply that a KT point is a local minimizer, except in rare cases. A descent method with this property is called globally convergent, that is, the method does not require the initial point $x^{(1)}$ close to x^*.

A globally convergent method for optimization usually defines a merit function ϕ to force convergence from poor starting points. For unconstrained nonlinear least squares problems, the choice of ϕ is simple. It is natural to choose the objective function f as ϕ. For constrained nonlinear least squares problems, the choice of ϕ is complicated by the fact that $x^{(k)} + \delta^{(k)}$ should move closer to satisfying the constraints than $x^{(k)}$ and $\delta^{(k)}$ yields a reasonable decrease in the objective function. A number of merit functions are available for constrained nonlinear least squares problems solution. These include the ℓ_1 *penalty function* proposed by S.P. Han [11] and used by R. Fletcher [9], the *augmented Lagrange functions* of M.R. Hestenes [12] and Fletcher [7], the *merit function* of G. Di Pillo and L. Grippo [6] and the *merit function* of P.T. Boggs and J.W. Tolle [2].

Strategies are available to achieve the descent property in nonlinear least squares methods. These are: *line search* strategy and *trust region* strategy. Line search strategy transfers a multivariable minimization problem into a series of sub-minimization problems with single variable, and are the most commonly used in practice. In line search methods, $\delta^{(k)}$ is obtained in the form $\delta^{(k)} = \alpha_k s^{(k)}$ where $s^{(k)}$ is a descent direction of the merit function $\phi(x)$ at $x^{(k)}$ and $\alpha_k > 0$ is a steplength along the direction $s^{(k)}$. The descent direction $s^{(k)}$ is generally obtained as a minimizer of some model problem which is a local approximation to the original problem at $x^{(k)}$ and the steplength α_k is determined by some line search method so that $x^{(k)} + \alpha_k s^{(k)}$ gives a sufficient decrease in a chosen merit function from $x^{(k)}$. Specifically, α_k is determined such that $\phi(x^{(k)} + \alpha_k s^{(k)}) < \phi(x^{(k)})$. For methods with line search, the convergence and the convergence rate depend upon the reduction on the merit

function at each iteration, which relies on the choices of the descent direction $s^{(k)}$ and the steplength α_k. Theoretically, exact line searches satisfying $\alpha_k = \text{argmin}\, \phi(x^{(k)} + \alpha s^{(k)})$ usually gives the maximal reduction on the function $\phi(x^{(k)} + \alpha s^{(k)})$. But determining α_k by exact line search is not necessary and inefficient when iterate $x^{(k)}$ is remote from x^*. Practical line search methods search for a steplength satisfying certain conditions. These conditions are readily satisfied so that the steplength can be effectively determined and guarantee the convergence of a method when the search direction $s^{(k)}$ is sufficient descent [1]. Among various conditions, *Goldstein's conditions* are widely used in practice. These conditions require the steplength α_k to satisfy

$$\phi(x^{(k)} + \alpha s^{(k)}) \le \phi(x^{(k)}) + \rho\alpha\nabla\phi(x^{(k)})^\top s^{(k)},$$
$$\phi(x^{(k)} + \alpha s^{(k)}) \ge \phi(x^{(k)}) + \sigma\alpha\nabla\phi(x^{(k)})^\top s^{(k)},$$

where $0 < \rho < \sigma < 1$. Usually, these two conditions define an interval of acceptable α-values. A disadvantage of the second condition is that the minimum of $\phi(x^{(k)} + \alpha s^{(k)})$ may be excluded to the left of the interval. A remedy to avoid this case is to use

$$\nabla\phi(x^{(k)} + \alpha s^{(k)})^\top s^{(k)} \ge \sigma\nabla\phi(x^{(k)})^\top s^{(k)}$$

to replace that condition. Numerous line search methods are available. Strategies used in line search methods generally consist of *bracketing* and *sectioning*. The simplest line search methods are pattern searches such as *golden section search* and *Fibonnaci section search* [1]. These pattern searches use only evaluated function values. When the first order derivative information are available, the simplest line search is the *backtracking* which seeks the smallest integer i satisfying

$$\phi(x^{(k)} + t^i\alpha_0 s^{(k)}) \le \phi(x^{(k)}) + \rho t^i\alpha_0\nabla\phi(x^{(k)})^\top s^{(k)}$$

and set $\alpha_k = t^j\,\alpha_0$ if j is such an integer. Effective line search methods to determine α_k satisfying the Goldstein conditions are also available [1]. These methods employ sectioning schemes and interpolations.

Trust region strategy generates $\delta^{(k)}$ by solving some model subproblem with a trust region constraint. The trust region defined by $\| \delta \| \le \Delta_k$ is a neighborhood about the current point $x^{(k)}$ and is adjusted in such a way that the subproblem model is believed to have adequately approximated the chosen merit function in that region. For a given trust region radius Δ_k, the solution $\delta^{(k)}$ of minimizing the subproblem model within the region $\| \delta \| \le \Delta_k$ is sought. If a satisfactory reduction on the merit function is obtained at $x^{(k)} + \delta^{(k)}$, $x^{(k)} + \delta^{(k)}$ is accepted as $x^{(k+1)}$. If the computed step $\delta^{(k)}$ is not acceptable, the sub-problem model is not accurate enough in that region and the radius of the trust region is reduced to improve the accuracy of the approximation and the step is recomputed. The trust region radius may be increased after an acceptable step. Methods with trust region converge globally. An obstacle preventing the trust region methods from common use is the effective solution of trust region subproblems. Repeated solution of system of linear equations with modified coefficient matrices are required to obtain a satisfactory $\delta^{(k)}$ and increase the complexity of trust region methods. Now effective approximate solution methods for the solution of trust region subproblems exist. These include positive definite *dogleg path* method [5,13], indefinite dogleg path methods [19], optimal path method in two-dimensional subspaces [3] and conjugate gradient method [15].

Subproblem models are generally local approximations to a chosen merit function at current iterate point $x^{(k)}$. These approximations are generally linear or quadratic. A linear approximation is the simplest function and a quadratic approximation is usually more accurate than linear approximation in certain neighborhood of $x^{(k)}$. The quadratic function is one of the simplest smooth functions and the minimum of a model problem with quadratic function is well-determined and is relatively easy to determine.

As an important class of optimization problems, any method for general minimization can be used to solve nonlinear least squares problems. However, special methods which take the advantage of the special structure of the objective function in nonlinear least squares are available. Most special methods are based on the well-known Gauss–Newton (G-N) method which requires only the first order derivatives of problem functions. In the G-N method, the matrix $A(x^{(k)})\,A(x^{(k)})^\top$ is used to approximate the Hessian matrix $\nabla^2 f(x^{(k)})$. Since $r(x^*)$ may be small or zero as $f(x)$ is minimized, this may be a good approximation when $x^{(k)}$ is close to x^*. It is well known that for zero residual problems, the local convergence rate of the G-N method is quadratic, but for small residual problems,

the convergence of the method is at most linear and even not converge for large residual problems. According to the information used in a method, methods for nonlinear least squares can be divided into first order derivative methods, second order derivative methods and methods without derivatives. Among them the first order derivative methods are the most commonly used. These include (damped) G-N method, quasi-Newton methods, hybrid methods [1,10], adaptive method [4], factorized quasi-Newton methods [17,18] and methods with quasi-Newton corrections to Gauss–Newton matrix. As for second order derivative methods, the Newton method is the most famous. However, the disadvantages of the Newton method such as the evaluation of second order derivatives and convergence only local make the method rarely used in practice. As for methods without using derivatives, one can make a choice among the direction set methods [8] and the hybrid Gauss–Newton–Broyden method [16]. Theoretical analysis and numerical results show that these methods have good convergence properties and are robust and trust.

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- ABS Algorithms for Optimization
- Gauss–Newton Method: Least Squares, Relation to Newton's Method
- Generalized Total Least Squares
- Least Squares Orthogonal Polynomials
- Least Squares Problems
- Nonlinear Least Squares: Newton-type Methods
- Nonlinear Least Squares: Trust Region Methods

References

1. Al-Baali M, Fletcher R (1985) Variational methods for nonlinear least-squares. J Oper Res Soc 36:405–421
2. Boggs PT, Tolle JW (1987) Merit functions and nonlinear programming. SIAM Conf. Optimization, Houston
3. Byrd RH, Schnabel RB, Shultz GA (1988) Approximate solution of the trust region problem by minimization over two-dimensional subspaces. Math Program 40:247–263
4. Dennis JE, Gay DM, Welsch RE (1981) An adaptive nonlinear least-squares algorithm. TOMS 7:348–368
5. Dennis JE, Mei HHW (1979) Two new unconstrained optimization algorithms which use function and gradient values. JOTA 28:453–482
6. Di Pillo G, Grippo L (1979) A new class of augmented Lagrangian in nonlinear programming. SIAM J Control Optim 17:618–628
7. Fletcher R (1973) An exact penalty function for nonlinear programming with inequalities. Math Program 5:129–150
8. Fletcher R (1980) Practical methods of optimization: Unconstrained optimization, vol 1. Wiley, New York
9. Fletcher R (1984) An ℓ^1 penalty method for nonlinear constraints. In: Boggs PT, Byrd RH, Schnabel RB (eds) Numerical Optimization. SIAM, Philadelphia, pp 26–40
10. Fletcher R, Xu CX (1987) Hybrid methods for nonlinear least squares. IMA J Numer Anal 7:371–389
11. Han SP (1977) A globally convergent method for nonlinear programming. JOTA 22:297–309
12. Hestenes MR (1977) Multiplier and gradient methods. JOTA 4:303–320
13. Powell MJD (1970) A hybrid method for nonlinear equations. In: Rabinowitz P (ed) Numerical Methods for Nonlinear Algebraic Equations. Gordon and Breach, New York, pp 87–114
14. Ruhe A, Wedin PA (1980) Algorithms for separable nonlinear least squares problems. SIAM Rev 22:318–337
15. Steihaug T (1983) The conjugate gradient method and trust regions in large scale optimization. SIAM J Numer Anal 20:626–637
16. Xu CX (1990) Hybrid methods for nonlinear least squares problems without calculating derivatives. JOTA 65:555–575
17. Xu CX, Ma XF, Kong MY (1996) A class of factorized quasi-Newton methods for nonlinear least squares problems. J Comput Math 14:143–158
18. Yabe H, Takahashi T (1991) Factorized quasi-Newton methods for nonlinear least squares problems. Math Program 51:75–100
19. Zhang JZ, Xu CX (1999) A class of indefinite dogleg methods for unconstrained minimization. SIAM J Optim 9:646–667

Nonlinear Least Squares: Trust Region Methods

CHENGXIAN XU
Xian Jiaotong University, Xian, China

MSC2000: 49M37

Article Outline

Keywords

Trust region method; Solution of a system; Optimal path; Dogleg path; Negative curvature

Descent methods for nonlinear least squares problems generate a sequence $\{x^{(k)}\}$ such that $f(x^{(k+1)}) < f(x^{(k)})$ and $x^{(k)}$ converges to a local minimizer of the objective function $f(x)$. Methods with *line searches* generate the sequence using the iteration

$$x^{(k+1)} = x^{(k)} + \alpha_k s^{(k)} ,$$

where $s^{(k)}$ is a descent direction of $f(x)$ at $x^{(k)}$ and α_k is a steplength along the direction $s^{(k)}$ determined by line searches. The search direction $s^{(k)}$ is obtained as the solution of the system

$$B_k s = -g_k ,$$

where g_k is the gradient of $f(x)$ at $x^{(k)}$ and B_k is either the Hessian G_k of $f(x)$ at $x^{(k)}$ or its approximation. The matrix B_k is required to be positive definite, so that the solution $s^{(k)}$ which is the unique minimizer of the quadratic model

$$q_k(\delta) = f_k + g_k^\top \delta + \frac{1}{2}\delta^\top B_k \delta$$

is guaranteed to be a descent direction, where $q_k(\delta)$ is a local approximation to $f(x)$ at $x^{(k)}$, $f_k = f(x^{(k)})$ and $\delta = x - x^{(k)}$.

When the matrix B_k is not positive definite that are often encountered in practical calculation, the quadratic model does not have a unique minimizer and methods with line searches may not be defined. A more realistic approach is to take $x^{(k+1)} = x^{(k)} + \delta^{(k)}$. The $\delta^{(k)}$ minimizes the quadratic model within a neighborhood $N(x^{(k)}, \Delta_k)$ of the point $x^{(k)}$ in which the quadratic function is believed adequately to approximate the function $f(x)$. The neighborhood $N(x^{(k)}, \Delta_k)$ is generally called a *trust region* and methods having this framework are called *trust region methods*. These methods can retain the rapid rate of convergence of Newton type methods, but are also generally applicable and *globally convergent*.

The development of trust region methods can be traced back to the work of K. Levenberg [8] and D.W. Marquardt [9] on unconstrained *nonlinear least squares* problems

$$\min f(x) = \frac{1}{2}\sum_{i=1}^{m}[r_i(x)]^2 = \frac{1}{2} r(x)^\top r(x) ,$$

where $r(x) = (r_1(x), \ldots, r_m(x))^\mathsf{T}$ and $r_i(x)$, $i = 1, \ldots, m$, are nonlinear functions of $x \in \mathbf{R}^n$. Assuming that $r_i(x)$, $i = 1 \cdots, m$, hence $f(x)$ is twice continuously differentiable, the gradient and the Hessian matrix of $f(x)$ are defined by

$$\begin{aligned} g(x) &= \nabla f(x) = A(x) r(x), \\ G(x) &= \nabla^2 f(x) \\ &= A(x)A(x)^\top + \sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x) . \end{aligned}$$

The *Levenberg–Marquardt method* is a modification of the well-known *Gauss–Newton method* and is based upon the idea that when the full Gauss–Newton step fails, a proper bias towards the steepest descent direction may generate a satisfactory reduction on the function $f(x)$. Thus the step $\delta^{(k)}$ between iterates of the Levenberg–Marquardt method is a solution of the system

$$(A_k A_k^\top + \mu_k I)\delta = -g_k$$

for some properly chosen $\mu_k \geq 0$. The application of the Levenberg–Marquardt method to general unconstrained minimization is given by S.M. Goldfeld, R.E. Quandt and H.F. Trotter [6], in which the matrix $A_k A_k^\top$ in the above system is just replaced by the matrix B_k, that is,

$$(B_k + \mu_k I)\delta = -g_k .$$

Let $\delta^{(k)}$ be the solution of the system. Then $\delta^{(k)}$ solves the trust region subproblem

$$\begin{cases} \min & q_k(\delta) = f_k + g_k^\top \delta + \frac{1}{2}\delta^\top B_k \delta \\ \text{s.t.} & \|\delta\| \leq \Delta_k \end{cases}$$

with $\Delta_k = \|\delta^{(k)}\|$. Also for any given Δ_k, the solution $\delta^{(k)}$ of the trust region subproblem usually satisfies the system

$$(B_k + \mu_k I)\delta = -g_k, \quad \left\|\delta^{(k)}\right\| = \Delta_k ,$$

where $\mu_k \geq 0$ such that $B_k + \mu_k I$ is at least positive semidefinite, except for the case when B_k is positive definite and $\|B_k^{-1} g_k\| \leq \Delta_k$, then the solution is $\delta^{(k)} = -$

$B_k^{-1} g_k$. It can be seen that the solution $\delta^{(k)}$ can be obtained by controlling either the value μ_k or the radius Δ_k.

Early versions of the Levenberg–Marquardt and Goldfeld–Quandt–Trotter methods determined $\delta^{(k)}$ from the system by controlling μ_k and recently used trust region methods determined $\delta^{(k)}$ by controlling the radius Δ_k. Direct control of μ_k has a number of disadvantages. One is that there does not seem to have a reasonable and automatic initial choice for the value μ_1, while a reasonable value for initial Δ_1 can be a small fraction of the size $\|x^{(1)}\|$ of the starting point $x^{(1)}$. One more problem occurs when $x^{(k)} + \delta^{(k)}$ leads to an increase in the objective function $f(x)$. In this case, the function and derivative information evaluated at $x^{(k)}$ and $x^{(k)} + \delta^{(k)}$ can be used to estimate a required decrease in radius Δ_k, but it is not clear how these information are used to estimate a reasonable increase for μ_k.

In implementing recent forms of trust region methods, there are two main problems. One problem is how the trust region radius Δ_k shall be chosen. To prevent undue restriction of the step $\delta^{(k)}$, the trust region radius Δ_k should be as large as possible under the condition that $q_k(\delta)$ adequately approximates $f(x)$ in that region. Let $\delta^{(k)}$ be the solution of the trust region subproblem for given Δ_k, the agreement between $q_k(\delta)$ and $f(x)$ in the neighborhood $N(x^{(k)}, \Delta_k)$ can be measured by the comparison between the actual reduction in $f(x)$

$$\text{ared}(\delta^{(k)}) = f(x^{(k)}) - f(x^{(k)} + \delta^{(k)})$$

and the reduction predicted by the quadratic model

$$\begin{aligned}\text{pred}(\delta^{(k)}) &= f(x^{(k)}) - q_k(\delta^{(k)}) \\ &= -g_k^{\top}\delta^{(k)} - \frac{1}{2}\delta^{(k)\top} B_k \delta^{(k)} ,\end{aligned}$$

where B_k is either the Hessian matrix G_k of $f(x)$ or its approximation, for example $A_k A_k^{\top}$. If $\text{ared}(\delta^{(k)})$ is satisfactory compared with the pred $(\delta^{(k)})$, which implies a good reduction in f, the trust region is referred to as proper. Then $x^{(k)} + \delta^{(k)}$ is accepted as a new iterate point $x^{(k+1)}$ and the trust region radius may either keep unchanged or be increased in case the reduction in f is sufficient and the trust region constraint is active. If the computed step $\delta^{(k)}$ is not acceptable, which occurs when $q_k(\delta)$ is not accurate enough in $N(x^{(k)}, \Delta_k)$, then the radius Δ_k is reduced to improve accuracy of the approximation and the step is recomputed from the trust region subproblem with reduced Δ_k. A model trust region method with direct control of Δ_k can now be described as follows:

1) Give parameters $0 < \gamma_1 < 1 < \gamma_2$, $0 < \eta_1 < \eta_2 < 1$ and $\Delta_{\max} > 0$, initial point $x^{(1)}$ and $\Delta_1 (\leq \Delta_{\max})$ and set $k = 1$.
2) Calculate f_k, g_k. If $g_k = 0$ then terminate, else form B_k.
3) Solve the trust region subproblem for $\delta^{(k)}$.
4) Compute $\theta_k = \text{ared}(\delta^{(k)})/\text{pred}(\delta^{(k)})$.
5) If $\theta_k < \eta_1$, then $\Delta_k = \gamma_1 \Delta_k$ and go to step 3).
6) $x^{(k+1)} = x^{(k)} + \delta^{(k)}$,

$$\Delta_{k+1} = \begin{cases} \overline{\Delta} & \text{if } \theta_k \geq \eta_2 \\ & \text{and } \left\|\delta^{(k)}\right\| = \Delta_k, \\ \Delta_k & \text{otherwise,} \end{cases}$$

set $k = k + 1$, where $\overline{\Delta} = \min\{\gamma_2 \Delta_k, \Delta_{\max}\}$

The most important issue to implement a trust region method is the efficient solution of the trust region subproblem. There are three possible cases to determine the solution of the trust region subproblem:

a) *Newton step case*: B_k is positive definite and $\|B_k^{-1} g_k\| \leq \Delta_k$. The solution is $\delta^{(k)} = -B_k^{-1} g_k$.
b) *General case*: B_k is positive definite and $\|B_k^{-1} g_k\| > \Delta_k$ or B_k is indefinite and $\|(B_k - \mu_1^{(k)} I)^{+} g_k\| \geq \Delta_k$ where $(A)^{+}$ denotes the generalized inverse of the matrix A and $\mu_1^{(k)}$ is the smallest eigenvalue of B_k. The solution is $\delta^{(k)} = -(B_k + \mu_k I)^{-1} g_k$ with $\mu_k > \mu_\ell = \max\{0, -\mu_1^{(k)}\}$, $\|\delta^{(k)}\| = \Delta_k$.
c) *Hard case* [11]: B_k is indefinite and $\|(B_k - \mu_1^{(k)} I)^{+} g_k\| < \Delta_k$. The solution is $\delta^{(k)} = -(B_k - \mu_1^{(k)} I)^{+} g_k + t_k u_1^{(k)}$ where $u_1^{(k)}$ is the eigenvector of B_k corresponding to $\mu_1^{(k)}$ and t_k is determined from the equation $\|(B_k - \mu_1^{(k)} I)^{+} g_k + t u_1^{(k)}\| = \Delta_k$.

Since the hard case rarely occurs, the solution of trust region subproblem is closely related to the solution of the equation

$$\begin{aligned}\phi_k(\mu) &= \|\delta_k(\mu)\| - \Delta_k \\ &= \left\|(B_k + \mu I)^{-1} g_k\right\| - \Delta_k = 0 .\end{aligned}$$

This is a nonlinear equation with respect to μ and there is generally no finite method to find its exact solution. Since $\phi_k(\mu_\ell) > 0$ and $\phi_k(\infty) = -\Delta_k$, it is clear that the

solution can be found in the interval (μ_ℓ, ∞). Since $\phi_k(\mu)$ is a convex and continuous monotonic decreasing function in the interval (μ_ℓ, ∞), the solution of the equation is unique. M.D. Hebden [7] used the practical structure of the function $\phi_k(\mu)$ to propose a method that generates a sequence $\{\mu^{(j)}\}$ such that $\mu^{(j)} \to \mu_k$. Let $\mu^{(j)}$ be a current estimation to μ_k, a fractional function

$$\psi_k(\mu) = \frac{\alpha}{\mu + \beta} - \Delta_k$$

is used to approximate the function $\phi_k(\mu)$ such that

$$\psi_k(\mu^{(j)}) = \phi_k(\mu^{(j)}), \quad \psi_k'(\mu^{(j)}) = \phi_k'(\mu^{(j)})$$

where

$$\alpha = -\frac{(\phi_k(\mu^{(j)}) + \Delta_k)^2}{\phi_k'(\mu^{(j)})},$$

$$\beta = -\frac{\phi_k(\mu^{(j)}) + \Delta_k}{\phi_k'(\mu^{(j)})} - \mu^{(j)}.$$

Setting $\psi_k(\mu) = 0$ and substituting α and β give the iteration

$$\mu^{(j+1)} = \mu^{(j)} + \xi(\mu^{(j)}) \left[\frac{\|\delta_k(\mu^{(j)})\|}{\Delta_k} - 1 \right],$$

$$\xi(\mu^{(j)}) = \frac{\|\delta_k(\mu^{(j)})\|^2}{\delta_k(\mu^{(j)})^\top (B_k + \mu^{(j)} I)^{-1} \delta_k(\mu^{(j)})}.$$

J.J. Moré and D.C. Sorensen [11] also derived the iteration by applying the Newton method to the equation

$$h_k(\mu) = \frac{1}{\Delta_k} - \frac{1}{\|\delta_k(\mu)\|}.$$

The iteration can be calculated in the following way.

i) Factorize the matrix $(B_k + \mu^{(j)} I) = R^\top R$ with R an upper triangular matrix.

ii) Solve $R^\top R\delta = -g_k$ for $\delta_k(\mu^{(j)})$ using forward and backward substitutions.

iii) Solve $R^\top z = \delta_k(\mu^{(j)})$ for z_j using forward substitution.

iv)
$$\mu^{(j+1)} = \mu^{(j)} + \frac{\|\delta_k(\mu^{(j)})\|^2}{\|z_j\|^2} \left[\frac{\|\delta_k(\mu^{(j)})\|}{\Delta_k} - 1 \right].$$

To start the iteration, an initial value for $\mu^{(1)}$ is required. A natural choice is the value $\mu^{(1)} = \mu_\ell$. This choice needs the calculation of the smallest eigenvalue of the matrix B_k and will cause numerical difficulties when B_k is not positive definite. Moré [10] used the choice

$$\mu^{(1)} = \mu_{k-1} \frac{\Delta_{k-1}}{\Delta_k}$$

as the initial value of μ_k at the kth iteration where μ_{k-1} is the accepted value of the equation $\phi_{k-1}(\mu) = 0$ at the $(k-1)$th iteration. J.E. Dennis and R.B. Schnabel [4] proposed the choice

$$\mu^{(1)} = \mu_{k-1} + \xi(\mu_{k-1}) \left[\frac{\|\delta_{k-1}(\mu_{k-1})\|}{\Delta_k} - 1 \right],$$

$$\xi(\mu_{k-1}) = \frac{\|\delta_{k-1}(\mu_{k-1})\|^2}{\delta_{k-1}(\mu_{k-1})^\top (B_{k-1} + \mu_{k-1} I)^{-1} \delta_{k-1}(\mu_{k-1})},$$

which is an analog to the iteration for $\mu^{(j+1)}$. Of course, a safeguard strategy is imposed to force convergence, that is, lower and upper bounds for $\mu^{(j)}$ are provided and updated in iteration process. The iteration finds an approximate solution $\delta^{(k)}$ satisfying the *Hebden conditions* [7]

$$\text{pred}(\delta^{(k)}) \geq \tau(f_k - q_k^*),$$

$$\left| \|\delta^{(k)}\| - \Delta_k \right| \leq \rho \Delta_k,$$

where τ and ρ are positive constants and q_k^* is the optimal value of the trust region subproblem for given Δ_k. Then strong convergence result can be obtained (see [5]).

In the general case, the repeated *solution of the system* $(B_k + \mu^{(j)} I)\delta = -g_k$ for different values of $\mu^{(j)}$ to determine a satisfactory approximate solution $\delta^{(k)}$ of the trust region subproblem for given value Δ_k and the recomputation of $\delta^{(k)}$ for reduced values Δ_k may be required at each iteration. It is this complication that prevent trust region methods from wide use in past two decades. Most practical trust region methods attempt to find an approximate solution of the trust region subproblem in a reasonable amount of computational effort. G.A. Shultz, Schnabel and R.H. Byrd [13] proposed general conditions on the approximate solution $\delta^{(k)}$ to ensure a satisfactory reduction on $q_k(\delta)$ so that resulting trust region methods have strong convergence properties. These conditions are as follows:

A) There exist constants $\beta_1 > 0$ and $\sigma_1 > 0$ such that for all Δ_k and B_k

$$\text{pred}(\delta^{(k)}) \geq \beta_1 \|g_k\| \min\left\{\Delta_k, \sigma_1 \frac{\|g_k\|}{\|B_k\|}\right\}.$$

B) There exists a $\beta_2 > 0$ such that for all Δ_k and B_k

$$\text{pred}(\delta^{(k)}) \geq -\beta_2 \mu_1^{(k)} \Delta_k^2 .$$

C) If the matrix B_k is positive definite and $\|B_k^{-1} g_k\| \leq \Delta_k$, then

$$\delta^{(k)} = -B_k^{-1} g_k .$$

Condition A) ensures the global convergence of a trust region method to a point satisfying the first order necessary condition and a trust region method satisfying condition B) will converge to a point at which the second order necessary conditions are satisfied. When condition C) is satisfied, a convergent trust region method converges at a quadratic rate if $G(x^*)$ is positive definite at the limit point x^*.

Let Δ_k vary in the interval $(0, \infty)$, then the solution $\delta_k(\mu)$ of the trust region subproblem forms a continuous *optimal path* $\Gamma^{(k)}(\tau)$ in the space $\mathbf{R}^n$. The optimal path can be expressed as

$$\Gamma^{(k)}(\tau) = \Gamma_1^{(k)}(t(\tau)) + \Gamma_2^{(k)}(\theta(\tau))$$

where

$$\Gamma_1^{(k)}(t(\tau)) = -\sum_{i \in I} \frac{t(\tau)}{\mu_i^{(k)} t(\tau) + 1} v_i^{(k)} - t(\tau) \sum_{i \in N} v_i^{(k)},$$

$$\Gamma_2^{(k)}(\theta(\tau)) = \theta(\tau) u_1^{(k)},$$

$$t(\tau) = \begin{cases} \tau & \text{if } \tau < \frac{1}{\mu_\ell}, \\ \frac{1}{\mu_\ell} & \text{if } \tau \geq \frac{1}{\mu_\ell}, \end{cases}$$

$$\theta(\tau) = \begin{cases} 0 & \text{if } \mu_\ell = 0, \\ \max\left\{\tau - \frac{1}{\mu_\ell}, 0\right\} & \text{if } \mu_\ell > 0, \end{cases}$$

$$I = \left\{i:\ \mu_i^{(k)} \neq 0\right\}, \quad N = \left\{i:\ \mu_i^{(k)} = 0\right\}$$

$$v_i^{(k)} = {u_i^{(k)}}^\top g_k u_i^{(k)}, \quad i = 1, \ldots, n ,$$

and $\mu_1^{(k)} \leq \cdots \leq \mu_n^{(k)}$ are eigenvalues of B_k and $u_i^{(k)}$, $i = 1, \ldots, n$, are corresponding orthonormal eigenvectors. The optimal path has two properties. As a point x proceeds from $x^{(k)}$ along the path: i) the distance to $x^{(k)}$ is monotonically increasing, and ii) the value of $q_k(\delta)$ is monotonically decreasing. These properties guarantee that for any given Δ_k, the solution of the trust region subproblem is $\delta^{(k)} = \Gamma^{(k)}(\tau_k)$ where τ_k is uniquely determined from the equation $\| \Gamma^{(k)}(\tau) \| = \Delta_k$. The formulation of the path $\Gamma^{(k)}(\tau)$ needs the calculation of all eigenvalues and eigenvectors of the matrix B_k. This is time consuming and is unrealistic. J.P. Bulteau and J.-Ph. Vial [1] restrict the solution of the trust region subproblem in a two-dimensional subspace $S = \text{span}[a_1 a_2]$ and choose the solution of the problem

$$\begin{cases} \min & q_k(\delta) = f_k + g_k^\top \delta + \frac{1}{2} \delta^\top B_k \delta , \\ \text{s.t.} & \|\delta\| \leq \Delta_k,\ \delta \in S , \end{cases}$$

as an approximate solution of the subproblem. The vectors a_1 and a_2 are chosen in such a way that $A = [a_1 a_2]$, $A^\top A = I$. Then any $\delta \in S$ can be expressed as $\delta = Az$ for any $z \in \mathbf{R}^2$ and the restricted subproblem can be simplified as

$$\begin{cases} \min & q_k(z) = f_k + g_k^\top A z + \frac{1}{2} z^\top A^\top B_k A z , \\ \text{s.t.} & \|z\| \leq \Delta_k . \end{cases}$$

This is a trust region subproblem in the space $\mathbf{R}^2$ and can be easily solved using the optimal path method, since we only need calculate the whole eigensystem of a (2×2)-matrix $A^\top B_k A$ to form the optimal path $\Gamma_S^{(k)}(\tau)$. The subspace S is generally chosen in the following way:

$$S = \begin{cases} \text{span}[-g_k, -B_k^{-1} g_k] & \text{if } B_k \text{ P.D.}, \\ \text{span}[-g_k, u_1^{(k)}] & \text{if } B_k \text{ I.D.}, \end{cases}$$

where P.D. and I.D. denote positive definite and indefinite, respectively. In fact the path $\Gamma_S^{(k)}(\tau)$ can be regarded as a projection of the optimal path $\Gamma^{(k)}(\tau)$ in the subspace S and is an approximation of the path $\Gamma^{(k)}(\tau)$. Based on this idea, the recent efficient solution methods for trust region subproblems first form an approximate path $\Gamma_a^{(k)}(\tau)$ and then choose $\delta^{(k)} = \Gamma_a^{(k)}(\tau_k)$ with $\| \Gamma_a^{(k)}(\tau_k) \| = \Delta_k$. This greatly reduce the complication for the solution of the trust region subproblem, since for any given Δ_k the solution is obtained from the solution of the equation $\| \Gamma_a^{(k)}(\tau) \| = \Delta_k$ and the reformulation of the path for any reduced Δ_k is not required. When formulating an approximate path, it is required

to satisfy the two properties i) and ii) that the optimal path has.

Dogleg path methods are most effective and are first suggested by M.J.D. Powell and modified by Dennis and H.H.W. Mei [3] for positive definite matrices and extended to indefinite matrices by J.Z. Zhang and C.X. Xu [15]. Powell's method uses a single dogleg path in the form

$$\Gamma_{ps}^{(k)} = [0, \delta_{cp}^{(k)}, \delta_{np}^{(k)}]$$

to approximate the optimal path, where

$$\delta_{cp}^{(k)} = -\frac{g_k^\top g_k}{g_k^\top B_k g_k} g_k$$

is the minimizer of $q_k(\delta)$ in the steepest descent direction and

$$\delta_{np}^{(k)} = -B_k^{-1} g_k$$

is the global minimizer of $q_k(\delta)$. Then the solution $\delta^{(k)}$ of the problem

$$\min \left\{ q_k(\delta) \colon \ \|\delta\| \le \Delta_k,\ \delta \in \Gamma_{ps}^{(k)} \right\}$$

is taken as an approximate solution of the trust region subproblem. The solution $\delta^{(k)}$ can be obtained in the following way

$$\delta^{(k)} = \begin{cases} -B_k^{-1} g_k & \text{if } \left\| \delta_{np}^{(k)} \right\| \le \Delta_k \,, \\ -\frac{\Delta_k}{\|g_k\|} g_k & \text{if } \left\| \delta_{cp}^{(k)} \right\| \ge \Delta_k \,, \\ \delta_{cp}^{(k)} + t_k (\delta_{np}^{(k)} - \delta_{cp}^{(k)}) & \text{otherwise} \,, \end{cases}$$

where $t_k \in (0, 1)$ is determined from the equation $\| \delta_{cp}^{(k)} + t(\delta_{np}^{(k)} - \delta_{cp}^{(k)}) \| = \Delta_k$. Dennis and Mei modified the single dogleg path to form a double dogleg path

$$\Gamma_{dm}^{(k)} = [0, \delta_{cp}^{(k)}, \delta_{\eta}^{(k)}, \delta_{np}^{(k)}] \,,$$

where

$$\delta_{\eta}^{(k)} = \eta \delta_{np}^{(k)} \,, \quad \frac{\|g_k\|^4}{g_k^\top B_k g_k \cdot g_k^\top B_k^{-1} g_k} < \eta < 1 \,.$$

The solution $\delta^{(k)}$ in both the methods satisfies the general conditions A) and C). Hence both the methods are global convergent and convergence rate is quadratic if $G(x^*)$ is positive definite at limit point x^*.

Both the single and double dogleg path methods do work well when all the matrices B_k are positive definite. But they are unable to deal with the nonpositive definite case which occurs very often in practice. Zhang and Xu proposed indefinite dogleg paths for indefinite matrices B_k. One of these indefinite dogleg paths has the form

$$\Gamma_{ip}^{(k)} = [0, \delta_{\mu p}^{(k)}, \delta_{\mu}^{(k)}, d) \,,$$

where

$$\delta_{\mu p}^{(k)} = -\frac{g_k^\top g_k}{g_k^\top (B_k + \mu_k I) g_k} g_k \,,$$
$$\delta_{\mu}^{(k)} = -(B_k + \mu_k I)^{-1} g_k \,,$$

d is a *negative curvature* direction of the matrix B_k and μ_k is chosen such that $(B_k + \mu_k I)$ is positive definite. Since the optimal path $\Gamma^{(k)}(\tau)$ is infinite when B_k is indefinite, this is an infinite dogleg path. The solution $\delta^{(k)}$ obtained in the path has the following form

$$\delta^{(k)} = \begin{cases} -\frac{\Delta_k}{\|g_k\|} g_k & \text{if } \left\| \delta_{\mu p}^{(k)} \right\| \ge \Delta_k \,, \\ \delta_{\mu}^{(k)} + t_k d, \, t_k > 0 & \text{if } \left\| \delta_{\mu}^{(k)} \right\| \le \Delta_k \,, \\ \delta_{\mu p}^{(k)} + t_k (\delta_{\mu}^{(k)} - \delta_{\mu p}^{(k)}), & \\ t_k \in (0, 1) & \text{otherwise} \,, \end{cases}$$

where both t_k are determined from the equation $\|\delta^{(k)}(t)\| = \Delta_k$. When this indefinite dogleg path is combined with either the single or the double dogleg path, the resulting trust region method generates the solution $\delta^{(k)}$ satisfying all the three general conditions A)–C).

The negative curvature direction d can be effectively obtained from the Bunch–Parlett factorization

$$P B_k P^\top = L D L^\top$$

for symmetric matrices [2], where P is a permutation matrix, L a unit lower triangular matrix and D a block diagonal matrix with 1×1 and 2×2 diagonal blocks. Since matrices B_k and D have the same *inertia*, the negative curvature directions of B_k can be directly obtained from the eigenvectors of D corresponding to its negative eigenvalues. Let v_1 be the eigenvector corresponding to the smallest eigenvalue of D and $v = (v_1^\top L^\intercal P g_k) v_1$. Then

$$d = -\operatorname{sgn}(g_k^\top P^\top L^{-1} v) P^\top L^{-1} v$$

is a satisfactory negative curvature direction of B_k [15].

For large scale problems, methods with matrix factorizations generally are not suitable. T. Steihaug [14] proposed to use a *conjugate gradient method* for the system $B_k\delta = -g_k$ to find an approximate solution of the trust region subproblem. After δ_1, h_1 and d_1 are given, the iteration of the conjugate gradient method has the form:

$$\begin{aligned}
\delta_{j+1} &= \delta_j + \alpha_j d_j\,,\\
\alpha_j &= \frac{h_j^\top h_j}{d_j^\top B_k d_j}\,,\\
h_{j+1} &= h_j - \alpha_j B_k d_j\,,\\
d_{j+1} &= h_{j+1} + \beta_j d_j\,,\\
\beta_j &= \frac{h_{j+1}^\top h_{j+1}}{h_j^\top h_j}\,,\\
j &= 1,\ldots,n\,.
\end{aligned}$$

When this iteration is used to generate an approximate solution to the trust region subproblem, the solution is obtained in the following way. Assume that $d_j^\top B_k d_j > 0$ for $j = 1,\ldots,i-1$ and $\|\delta_i\| < \Delta_k$ and that h_i and d_i have been calculated. In case either $d_i^\top B_k d_i < 0$ or $d_i^\top B_k d_i > 0$ but $\|\delta_i + \alpha_i d_i\| \geq \Delta_k$, $\delta_i + t_i d_i$ is chosen as $\delta^{(k)}$ where the value t_i is determined from the equation $\|\delta_i + td_i\| = \Delta_k$. If $d_j^\top B_k d_j > 0$ for all $j = 1,\ldots,n$ and $\|\delta_{n+1}\| \leq \Delta_k$, then it follows from the properties of the conjugate gradient method that the matrix B_k is positive definite, $h_{n+1} = 0$ and $\delta_{n+1} = -B_k^{-1}g_k$ is the exact solution of the system $B_k\delta = -g_k$. Therefore, δ_{n+1} is the desired solution of trust region subproblem. The conjugate gradient method for the solution of the trust region subproblem has the form.

1) Give $\delta_1 = 0$, $h_1 = -g_k$, $d_1 = h_1$, $\eta_k \in (0,1)$, $j = 1$.
2) If $d_j^\top B_k d_j < 0$ then go to 6) below, else calculate $\alpha_j = h_j^\top h_j / d_j^\top B_k d_j$ and $\delta_{j+1} = \delta_j + \alpha_j d_j$.
3) If $\|\delta_{j+1}\| \geq \Delta_k$, go to 6) below.
4) Calculate $h_{j+1} = h_j - \alpha_j B_k d_j$. If $\|h_{j+1}\| \leq \eta_k\|g_k\|$, choose $\delta^{(k)} = \delta_{j+1}$.
5) $\beta_j = h_{j+1}^\top h_{j+1}/h_j^\top h_j$, $d_{j+1} = h_{j+1} + \beta_j d_j$, $j = j+1$, then go to 2).
6) Determine t_j such that $\|\delta_j + t_j d_j\| = \Delta_k$ and choose $\delta^{(k)} = \delta_j + t_j d_j$.

In fact, the conjugate gradient method for the approximate solution of the trust region subproblem is essentially a *multiple dogleg path* method. When the matrix B_k is positive definite, the multiple dogleg path is

$$\Gamma_{mp}^{(k)} = [0, \delta_1, \ldots, \delta_{n+1}]$$

and when the matrix B_k is indefinite, the multiple dogleg path is

$$\Gamma_{mi}^{(k)} = [0, \delta_1, \ldots, \delta_i, d_i)$$

where d_i is the first calculated negative curvature direction in the iteration process. Then the solution $\delta^{(k)}$ is obtained either in the path $\Gamma_{mp}^{(k)}$ or in the path $\Gamma_{mi}^{(k)}$.

See also

- ABS Algorithms for Linear Equations and Linear Least Squares
- ABS Algorithms for Optimization
- Conjugate-gradient Methods
- Gauss–Newton Method: Least Squares, Relation to Newton's Method
- Generalized Total Least Squares
- Large Scale Trust Region Problems
- Least Squares Orthogonal Polynomials
- Least Squares Problems
- Local Attractors for Gradient-related Descent Iterations
- Nonlinear Least Squares: Newton-type Methods
- Nonlinear Least Squares Problems

References

1. Bulteau JP, Vial J-Ph (1985) A restricted trust region algorithm for unconstrained optimization. JOTA 47:413–434
2. Bunch JB, Parlett BN (1971) Direct methods for solving symmetric indefinite system of linear equations. SIAM J Numer Anal 8:639–655
3. Dennis JE Jr, Mei HHW (1979) Two new unconstrained optimization algorithms which use function and gradient values. JOTA 28:453–482
4. Dennis JE Jr, Schnabel RB (1983) Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, Englewood Cliffs
5. Gay DM (1981) Computing optimal locally constrained step. SIAM J Sci Statist Comput 2:186–197
6. Goldfeld SM, Quandt RE, Trotter HF (1966) Maximisation by quadratic hill-climbing. Econometrica 34:541–551
7. Hebden MD (1973) An algorithm for minimization using exact second derivatives. Atomic Energy Res Establishment Harwell, Report TP 515

8. Levenberg K (1944) A method for the solution of certain nonlinear problems in least squares. Quart Appl Math 2:164–168
9. Marquardt DW (1963) An algorithm for least squares estimation of nonlinear parameters. SIAM J 11:111–115
10. Moré JJ (1978) The Levenberg–Marquardt algorithm: implementation and theory. In: Watson GA (ed) Numerical Analysis: Dundee. Lecture Notes Math. Springer, Berlin, pp 105–116
11. Moré JJ, Sorensen DC (1981) Computing a trust region step. SIAM J Sci Statist Comput 4:553–572
12. Powell MJD (1970) A hybrid method for nonlinear equations. In: Rabinowitz P (ed) Numerical methods for nonlinear algebraic equations. Gordon and Breach, New York, pp 87–114
13. Shultz GA, Schnabel RB, Byrd RH (1985) A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties. SIAM J Numer Anal 22:47–67
14. Steihaug T (1983) The conjugate gradient method and trust region in large scale optimization. SIAM J Numer Anal 20:626–637
15. Zhang JZ, Xu CX (1999) A class of indefinite dogleg methods for unconstrained minimization. SIAM J Optim 9:646–667

Nonlinear Systems of Equations: Application to the Enclosure of All Azeotropes

S. T. Harding, Christodoulos A. Floudas
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90C30

Article Outline

Keywords

Thermodynamics; Heterogeneous; Homogeneous; Azeotrope

An *azeotrope* occurs when a liquid mixture boils at a constant temperature, producing a vapor with the same composition as the liquid. This is a significant phenomenon in *distillation* processes. An azeotrope represents a barrier to distillation, beyond which further separation of the components in the mixture is not possible. Therefore, the problem of locating all azeotropes in a chemical mixture is more than simply a thermodynamic curiosity, it is important for a number of reasons. First, the task of locating *all* azeotropes in a chemical mixture is essential for the *synthesis of separation processes*. A separation process that is designed without complete knowledge about the azeotropes in the system is likely to be infeasible.

In addition, many thermodynamic models have been proposed which can predict the phase behavior of nonideal mixtures. Unfortunately, the accuracy of these models is not uniform over a wide range of mixtures. One useful way of testing the accuracy of a model for a given mixture is to compare the compositions of the azeotropes predicted by the model with those determined by experiment.

Despite the considerable interest in the area of predicting phase equilibria for chemical mixtures, relatively few methods for enumerating the azeotropes for a given system have been reported. The thermodynamic conditions for azeotropy constitute a nonlinear system of equations. This problem presents the additional challenge of finding all of the solutions to the nonlinear system of equations where the number of solutions is not known.

Nonlinear System of Equations

The most common type of azeotropes studied to date are called homogeneous azeotropes and occur when a single liquid phase is in equilibrium with the vapor phase. The thermodynamic conditions for homogeneous azeotropy are therefore gives,

1) equilibrium between the vapor phase and a single liquid phase, and
2) composition of the vapor phase is identical to the composition of the liquid phase.

Two other classifications of azeotropes, heterogeneous and reactive, will be discussed later in this article.

Condition 1: Phase Equilibrium

The equilibrium condition requires that the chemical potential of each component must be the same in the liquid and the vapor phases. If we consider a mixture of N components,

$$\mu_i^V = \mu_i^L , \quad \forall i \in N ,$$

where μ_i^V and μ_i^L represent the *chemical potential* of component i in the vapor and liquid phases. From the definition of the fugacity of component i in a mixture, $\hat{f}_i$,

$$\hat{f}_i^V = \hat{f}_i^L , \quad \forall i \in N ,$$

hence,

$$y_i \hat{\phi}_i^V P = x_i \gamma_i^L f_i^L , \quad \forall i \in N . \tag{1}$$

The symbol $\hat{\phi}_i^V$ represents the mixture *fugacity coefficient* of component i in the vapor phase. For the liquid phase, γ_i^L is the *activity coefficient*, and f_i^L is the fugacity of component i in the liquid phase. Rearranging (1) gives,

$$\frac{y_i}{x_i} = \frac{\gamma_i^L f_i^L}{\hat{\phi}_i^V P} , \quad \forall i \in N .$$

A common simplification is to assume that at low pressure the vapor phase can be modeled as an ideal gas, for which $\hat{\phi}_i^V = 1$. For the liquid phase the fugacity is equal to $f_i^L = \phi_i^{sat} P_i^{sat} (PF)_i$. But, for an ideal gas, $\phi_i^{sat} = 1$, and $(PF)_i = 1$. Therefore,

$$\frac{y_i}{x_i} = \frac{\gamma_i^L P_i^{\text{sat}}}{P} . \tag{2}$$

Condition 2: Equality of Phase Compositions

The azeotropy condition requires that the composition of the vapor phase is identical to the composition of the liquid phase.

$$y_i = x_i , \quad \forall i \in N . \tag{3}$$

Equations must also be added to require that the mole fractions in each phase sum to unity and have values between 0 and 1.

$$\begin{cases} \sum_{i \in N} y_i = \sum_{i \in N} x_i = 1 , \\ 0 \le y_i, x_i \le 1 , \qquad \forall i \in N . \end{cases} \tag{4}$$

Equations (2), (3), (4) constitute the nonlinear system of equations that are satisfied by a homogeneous azeotrope. Typical nonlinear equation solvers cannot be used robustly to find all of the solutions to this system of equations, since it is a nonlinear, constrained problem with multiple solutions. Many systems contain multiple azeotropes, each of which is a solution to the system. In addition, each pure component is a solution, giving at least N solutions.

Solution Methodologies

Most of the previous work reported in the literature has been limited to calculating homogeneous azeotropes using local nonlinear equation solvers. This means that the ability of these methods to predict azeotropes is dependent upon choosing good starting points for the solution technique. These methods cannot guarantee that all of the azeotropes have been located in a particular system. [1] calculated ternary homogeneous azeotropes using the Wilson model under isothermal conditions. [10] calculated homogeneous azeotropes of binary mixtures using an equation of state as the thermodynamic model. Their approach was to fix temperature and vary composition and volume until thermodynamic equilibrium conditions were satisfied. [12] also used an equation of state to calculate homogeneous azeotropes for binary mixtures.

[2] presented a search method for finding homogeneous and heterogeneous azeotropes which uses a *Levenberg–Marquardt algorithm* to find homogeneous azeotropes and then checks the stability of each solution with the tangent plane criterion described by [8]. A solution which is found to be unstable is then used as the starting point for a new search for an heterogeneous azeotrope. Again, this technique relies on local solution techniques and cannot guarantee that all azeotropes are located.

[4] proposed a method for locating all homogeneous azeotropes in multicomponent systems. This method uses a homotopy continuation technique for tracking branches of solutions to the nonlinear system of equations. They demonstrated that the technique performed robustly for systems containing up to five components using the Wilson activity coefficient equation. Recently, [3] have extended this method for heterogeneous azeotropes, and [9] have used it to predict

reactive azeotropes. [7] have formulated the problem of locating all homogeneous azeotropes as a global optimization problem in which each global minimum solution corresponds to an azeotrope. They applied this approach robustly using the Wilson, NRTL, UNIQUAC and UNIFAC activity coefficient equations for systems containing up to five components.

An excellent review on nonideal distillation, including a discussion on the computation of azeotropes has recently been published in [13].

Global Optimization Approach

In order to find all azeotropes, one must find all solutions to the system of nonlinear equations constituted by (2), (3), and (4). The method proposed in [6] reformulates the problem of enclosing all solutions of nonlinear systems of constrained equations into a global optimization problem in which the task is to enclose all global minimum solutions. In this approach, each nonlinear equality is replaced by two inequalities and a single slack variable is introduced. For the location of all homogeneous azeotropes, this corresponds to employing equations (2), (3), and (4) and reformulating them as the following global optimization problem:

$$\begin{cases} \min\limits_{\mathbf{x},T,s} & s \\ \text{s.t.} & \ln P - \ln P_i^{\text{sat}} - \ln \gamma_i - s \le 0, \quad \forall i, \\ & -\ln P + \ln P_i^{\text{sat}} + \ln \gamma_i - s \le 0, \quad \forall i, \\ & \sum\limits_{i \in N} x_i = 1, \\ & 0 < \mathbf{x} < 1, \\ & s \ge 0. \end{cases} \tag{5}$$

Problem (5) may have multiple global minima. Each global minimum of Problem (5) (where the solution $s^* = 0$) corresponds to an homogeneous azeotrope since when $s = 0$ the constraints (2), (3), and (4) are satisfied. Note that the first two sets of constraints of (5) correspond to the nonlinear equations (2) of the equilibrium constraint written as two inequalities. In addition, note that the nonlinear term $P_i^{sat}\gamma_i x_i$ appears as both a positive and a negative term. Thus, this term must be nonconvex in at least one of the two constraints. This means that if a local optimization approach is used to solve Problem (5), some or all of the global solutions may be missed.

For azeotropes in which less than N of the components participate (a k-ary azeotrope where $k \le N$), the case where $x_i = 0$ for one or more component must be accounted for. This can be done by multiplying the equilibrium constraints used in (5) by x_i. The general search for all k-ary homogeneous azeotropes is formulated as:

$$\begin{cases} \min\limits_{\mathbf{x},T,s} & s \\ \text{s.t.} & x_i(\ln P - \ln P_i^{\text{sat}} - \ln \gamma_i) - s \le 0, \\ & \quad \forall i \in N, \\ & x_i(-\ln P + \ln P_i^{sat} + \ln \gamma_i) - s \le 0, \\ & \quad \forall i \in N, \\ & \sum\limits_{i \in N} x_i = 1, \\ & 0 \le \mathbf{x} \le 1, \\ & s \ge 0. \end{cases} \tag{6}$$

0	Start with an initial guess for a solution (this does not affect the convergence of the algorithm).
1	Determine if the current solution satisfies the original nonlinear system of equation. IF it does, THEN store the solution, it corresponds to an azeotrope.
2	Is the current region smaller than the minimum size tolerance? IF it is, THEN fathom the region.
3	Partition the region into two subdomains.
4	Solve a lower bounding problem in each subdomain. IF the solution is greater than zero, THEN fathom the region. ELSE the solution is stored in the list of lower bounding solutions.
5	Choose among the active regions the one with the minimum lower bounding solution and update the current solution point. Return to Step 1. IF there are no regions remaining, THEN terminate.

Global optimization procedure

In [7], the authors have applied a global optimization approach to enclose all of the solutions to Problems (5) and (6). In this approach, each nonconvex term

is identified, and replaced by a convex underestimator. The solution of the convexified problem is then a lower bound on the solution to the original nonconvex problem. A *branch and bound* procedure is used to improve the lower bound. Regions that cannot contain an azeotrope have a positive lower bound and can be fathomed. Regions whose lower bound is less than or equal to zero are refined further until all of the azeotropes have been located. [7] have analyzed the Wilson, NRTL, UNIQUAC, and UNIFAC equations and have developed tight convex underestimators for all of the nonconvex terms for these thermodynamic functions.

Example 1 In this quaternary system (methanol, benzene, i-propanol, n-propanol) three binary azeotropes have been reported in the literature, as shown in Table 1. No experimental data was found for the ternary and quaternary systems.

Using the global optimization approach of [7], both the Wilson and NRTL equations predicted only the three reported azeotropes. The results for the Wilson equation are very close to the reported compositions and temperatures of the azeotropes. The results for the NRTL equation are also close to the reported values, with the exception of the Methanol-Benzene azeotrope.

Nonlinear Systems of Equations: Application to the Enclosure of All Azeotropes, Table 1
Solution for methanol (1) – benzene (2) – i-propanol (3) – n-propanol (4)

Azeo	x_1	x_2	x_3	x_4	T(deg C)
experimental data from [5]					
$1-2$	0.605	0.395	–	–	58.08
$1-3$	no azeotrope				
$1-4$	no azeotrope				
$2-3$	–	0.600	0.400	–	71.80
$2-4$	–	0.791	–	0.209	77.10
$3-4$	no azeotrope				
Wilson equation					
$1-2$	0.624	0.376	–	–	58.129
$2-3$	–	0.586	0.414	–	71.951
$2-4$	–	0.780	–	0.220	76.946
NRTL equation					
$1-2$	0.063	0.937	–	–	80.166
$2-3$	–	0.588	0.412	–	71.832
$2-4$	–	0.776	–	0.224	77.131

Homotopy Continuation Approach

The method proposed in [4] is based on tracking branches of solutions to nonlinear systems of equations that are perturbed from the original system. The key idea is to start with an equilibrium surface, on which all of the solutions are known a priori. The postulation is that every solution is connected to one of the branches of the initial surface. This surface is then gradually deformed, and the solution branches are tracked, ending with the actual nonideal equilibrium surface.

If the original equilibrium condition, (2), is rearranged so that the vapor mole fraction, y_i, is represented as a function of x_i, we obtain an equation of the form

$$y_i = \frac{\gamma_i^L P_i^{\text{sat}}}{P} x_i \,.$$

The ideal system can be represented by *Raoult's law*:

$$y_i^{\text{id}} = \frac{P_i^{\text{sat}}}{P} x_i \,.$$

One of the most convenient ways to represent the gradual deformation of the equilibrium surface is to use the linear convex combination of the ideal and nonideal equilibrium surfaces, where the homotopy parameter, t, determines the degree of deformation.

$$\widetilde{y}_i = \left[(1-t) + t\gamma_i^L\right] \frac{P_i^{\text{sat}}}{P} x_i \,.$$

The problem now is to find the roots of

$$\mathbf{h}(\mathbf{x}, t) = 0 \,, \tag{7}$$

where

$$\mathbf{h}(\mathbf{x}, t) = \widetilde{\mathbf{y}} - \mathbf{x} \,.$$

In [4], the authors use a *homotopy continuation* method to find the roots of (7). At $t = 0$, the pure components are used at the initial solutions to the ideal equilibrium equation. The homotopy parameter is then increased by a small amount and the solution branches are tracked. At each step, the determinant of $\mathbf{h}_\mathbf{x}$ is calculated, where

$$\mathbf{h}_\mathbf{x} = \left[\frac{\partial h_i}{\partial x_j}\right]$$
$$i = 1, \ldots, N-1, \quad j = 1, \ldots, N-1 \,.$$

Nonlinear Systems of Equations: Application to the Enclosure of All Azeotropes, Table 2
Solution for acetone (1) – chloroform (2) – methanol (3)

Azeo	x_1	x_2	x_3	T (deg C)
experimental data from [5]				
$1-2$	0.360	0.640	—	64.50
$1-3$	0.800	—	0.200	55.70
$2-3$	—	0.653	0.347	53.35
$1-2-3$	0.318	0.241	0.441	57.67
Wilson equation				
$1-2$	0.3437	0.6563	—	65.47
$1-3$	0.7944	—	0.2056	55.34
$2-3$	—	0.6481	0.3519	53.91
$1-2-3$	0.3234	0.2236	0.4530	57.58

If $\det[\mathbf{h}_\mathbf{x}] = 0$, then a bifurcation may occur, and an additional solution branch is started, which corresponds to an azeotrope.

[4] applied this method to a range of well-understood systems using the Wilson activity coefficient equation. Even though the homotopy continuation approach does not offer theoretical guarantee of locating all azeotropes, [4] observed that the method predicted all of the azeotropes that have been verified experimentally for these systems.

Example 2 [4] examined the ternary system of acetone, chloroform, and methanol They found that their homotopy continuation approach predicted all three binary azeotropes that are reported in the literature, using the Wilson equation. Their results are shown in Table 2.

Heterogeneous Azeotropes

Azeotropy is not limited to systems with a single liquid phase. In the more general case, where multiple liquid phases exist in equilibrium, a liquid mixture that boils at a constant temperature is called a heterogeneous azeotrope. Heterogeneous azeotropes can be of two different types. Type I heterogeneous azeotropes occur when the overall liquid composition is identical to the vapor composition. Conversely, Type II heterogeneous azeotropes occur when the overall liquid composition is not equal to the vapor composition. Type II heterogeneous azeotropes are possible theoretically, but have never been verified experimentally.

The phase equilibrium condition for a heterogeneous azeotrope in a system with M liquid phases is written:

$$\mu_i^V = \mu_i^{L_j}, \quad \forall i \in N, \ \forall j \in M .$$

When the definitions of the chemical potentials are applied and simplifications are made, the expression becomes,

$$\frac{y_i}{x_i^j} = \frac{\gamma_i^{L_j} P_i^{\text{sat}}}{P} .$$

In Type I heterogeneous azeotropes, the overall composition of the liquid is equal to the composition of the vapor,

$$y_i = \sum_{j=1}^{M} \psi^j x_i^j , \quad \forall i \in N ,$$

where ψ^j is the mole fraction of liquid phase j in the liquid mixture. The additional constraints are:

$$\sum_{i=1}^{N} y_i = 1 ,$$
$$\sum_{i=1}^{N} x_i^j = 1, \quad \forall j \in M ,$$
$$\sum_{j=1}^{M} \psi^j = 1 ,$$

An extension of the homotopy continuation method for homogeneous azeotropes of [4] was developed by [3] in order to examine the problem of finding all Type I heterogeneous azeotropes.

Reactive Azeotropes

Azeotropic behavior can also occur in reacting mixtures, [11]. The authors derived necessary and sufficient conditions for reactive azeotropes. These are:

$$Y_i - X_i = 0 , \quad i = 1, \ldots, N-1 ,$$

where

$$X_i = \frac{\nu_k x_i - \nu_i x_k}{\nu_k - \nu_T x_k} ,$$
$$Y_i = \frac{\nu_k y_i - \nu_i y_k}{\nu_k - \nu_T y_k} ,$$

where ν_i is the vector of reaction stoichiometric coefficients for component i, the index k refers to the reference component, and ν_T is the vector of the sum of the stoichiometric coefficients for each reaction.

In addition to this requirement, the system must also be in phase equilibrium, and in chemical equilibrium. The chemical equilibrium expression for each reaction r is written,

$$\left(\frac{K_{\text{eq}}^r}{K_{\text{eq}}^r+1}\right)\prod_{i\in\text{react}}(x_i\gamma_i)^{|\nu_{\text{react}}^r|} - \left(\frac{1}{K_{\text{eq}}^r+1}\right)\prod_{i\in\text{prod}}(x_i\gamma_i)^{|\nu_{\text{prod}}^r|} = 0\,,$$

where K_{eq}^r is the equilibrium coefficient for reaction r.

[9] have applied an arc-length continuation approach to search for all reactive azeotropes at fixed K_{eq} for systems with a single chemical reaction.

See also

- ► Contraction-Mapping
- ► Global Optimization Methods for Systems of Nonlinear Equations
- ► Interval Analysis: Systems of Nonlinear Equations
- ► Nonlinear Least Squares: Newton-type Methods

References

1. Aristovich VY, Stepanova EI (1970) Determination of the existence and composition of multicomponent azeotropes by calculation from data for binary systems. Zh Prikl Khim (Leningrad) 43:2192–2200
2. Chapman RG, Goodwin SP (1993) A general algorithm for the calculation of azeotropes in fluid mixtures. Fluid Phase Equilib 85:55–69
3. Eckert E, Kubicek M (1997) Computing heterogeneous azeotropes in multicomponent mixtures. Comput Chem Eng 21(3):347–350
4. Fidkowski ZT, Malone MF, Doherty MF (1993) Computing azeotropes in multicomponent mixtures. Comput Chem Eng 17(12):1141–1155
5. Gmehling J, Menke J, Krafczyk J, Fischer K (1994) Azeotropic data. Wiley-VCH, Weinheim
6. Harding ST, Floudas CA (1997) Global optimization in multiproduct and multipurpose batch design under uncertainty. Industr Eng Chem Res 36:1644–1664
7. Harding ST, Maranas CD, McDonald CM, Floudas CA (1997) Locating all azeotropes in homogeneous azeotropic systems. Industr Eng Chem Res 36:160–178
8. Michelsen ML (1981) The isothermal flash problem: Part I. Stability. Fluid Phase Equilib 9:1–19
9. Okasinski MJ, Doherty MF (1997) Thermodynamic behavior of reactive azeotropes. AIChE J 43(9):2227–2238
10. Teja AS, Rowlinson JS (1973) The prediction of the thermodynamic properties of fluids and fluid mixtures - IV. Critical and azeotropic states. Chem Eng Sci 28:529–538
11. Ung S, Doherty MF (1995) Necessary and sufficient conditions for reactive azeotropes in multireaction mixtures. AIChE J 41(11):2383–2392
12. Wang S, Whiting WB (1986) New algorithm for calculation of azeotropes from equations of state. Industr Eng Chem Process Des Developm 25:547–551
13. Widagdo S, Seider WD (1996) Azeotropic distillation. AIChE J 42:96–130

Nonlocal Sensitivity Analysis with Automatic Differentiation

LEIGH TESFATSION
Iowa State University, Ames, USA

MSC2000: 34-XX, 49-XX, 65-XX, 68-XX, 90-XX

Article Outline

Keywords

Nonlocal sensitivity analysis; Nasa program; Automatic differentiation; Feed algorithm; Adaptive homotopy; Adaptive computational method

Basic Problem Formulation

Sensitivity analysis problems typically reduce to determining the response of a vector $x^* = (x_1^*, \ldots, x_n^*)$ to changes in a scalar α^*, where x^* and α^* are required to satisfy an n-dimensional system of nonlinear equations of the form

$$0 = \psi(x,\alpha) \equiv (\psi^1(x,\alpha),\ldots,\psi^n(x,\alpha))^\top\,. \qquad (1)$$

This problem formulation, with a scalar parameter α, is more general than it might first appear. For example, suppose an analyst wishes to investigate the surface of function values $x = f(z)$ taken on by some function f: $\mathbf{R}^m \to \mathbf{R}^n$ as z ranges over a specified region Z in $\mathbf{R}^m$. One approach is to consider a suitably smooth curve s: $[0, 1] \to Z$ which roughly fills this region, of the form $z = s(\alpha)$, and to define a new function of the form $\psi(x, \alpha) \equiv x - f(s(\alpha))$. Solving the system of equations $\psi(x, \alpha) = 0$ for x as a function of α as α ranges from 0 to 1 then yields a curve of points $x(\alpha)$ on the function surface which gives some idea of the shape of this surface over the region Z.

Assuming ψ: $\mathbf{R}^{n+1} \to \mathbf{R}^n$ is twice continuously differentiable and has a nonsingular Jacobian matrix $\psi_x(x^*, \alpha^*)$, the implicit function theorem guarantees the existence of a continuously differentiable function $x(\alpha)$ taking some neighborhood $N(\alpha^*)$ of α^* into $\mathbf{R}^n$ such that

$$0 = \psi(x(\alpha), \alpha)\,, \quad \alpha \in N(\alpha^*)\,, \tag{2}$$

with $x(\alpha^*) = x^*$. From (2) one obtains the fundamental equation for sensitivity analysis,

$$\frac{dx(\alpha)}{d\alpha} = -\psi_x(x(\alpha), \alpha)^{-1}\psi_\alpha(x(\alpha), \alpha)\,, \quad \alpha \in N(\alpha^*)\,. \tag{3}$$

As it stands, (3) is an analytically incomplete system of ordinary differential equations. That is, a closed form representation for the Jacobian inverse $J(\alpha)^{-1} \equiv \psi_x(x(\alpha), \alpha)^{-1}$ as a function of α is often not obtainable for $n \geq 3$. Thus, the integration of (3) from initial conditions would typically require the supplementary algebraic determination of the Jacobian inverse $J(\alpha)^{-1}$ at each step in the integration process.

Why not simply incorporate a linear equation solver to accomplish the needed matrix inversions? Two reasons can be given. First, the Jacobian matrix might have one or more eigenvalues which are small in absolute value. Consequently, as can be seen using a singular value decomposition, the inverse matrix can be highly ill-conditioned in the sense that its elements have large absolute values and take on both positive and negative values. In this case, small roundoff and truncation errors can cause large errors in the resulting numerically determined component values of the sensitivity vector $dx(\alpha)/d\alpha$. Second, there exists an alternative approach [14] that has proven its reliability and efficiency in numerous contexts over the past twenty years: replace the algebraic operation of matrix inversion by an initial value problem highly suited for modern digital computers.

The latter approach is taken in [10]. The differential system (3) is extended by the incorporation of ordinary differential equations for the Jacobian inverse. More precisely, letting $A(\alpha)$ and $\delta(\alpha)$ denote the adjoint and the determinant of the Jacobian matrix $J(\alpha)$, and recalling that the inverse of any nonsingular matrix can be represented as the ratio of its adjoint to its determinant, the following differential system is validated for $x(\alpha)$, $A(\alpha)$, and $\delta(\alpha)$:

$$\frac{\mathrm{d}x(\alpha)}{\mathrm{d}\alpha} = -A(\alpha)\frac{\psi_\alpha(x(\alpha), \alpha)}{\delta(\alpha)}\,; \tag{4}$$

$$\frac{\mathrm{d}A(\alpha)}{\mathrm{d}\alpha} = \frac{A(\alpha)\,\mathrm{Trace}(A(\alpha)B(\alpha)) - A(\alpha)B(\alpha)A(\alpha)}{\delta(\alpha)}\,; \tag{5}$$

$$\frac{\mathrm{d}\delta(\alpha)}{\mathrm{d}\alpha} = \mathrm{Trace}(A(\alpha)B(\alpha))\,. \tag{6}$$

The ijth component of the matrix $B(\alpha) \equiv dJ(\alpha)/d\alpha$ appearing in equations (5) and (6) is

$$\sum_{k=1}^{n}\left(\psi^i_{jk}(x(\alpha), \alpha)\frac{\mathrm{d}x_k(\alpha)}{\mathrm{d}\alpha}\right) + \psi^i_{j,n+1}(x(\alpha), \alpha)\,, \tag{7}$$

where ψ^i_{jk} denotes the second partial of ψ^i with respect to x_j and x_k, and $\psi^i_{j,n+1}$ denotes the second partial of ψ^i with respect to x_j and α. Given (4), note that each of the components (7) is expressible as a known function of $x(\alpha)$, $A(\alpha)$, $\delta(\alpha)$, and α. Initial conditions for equations (4)–(6) must be provided at a parameter point α^* by specifying values for $x(\alpha^*)$, $A(\alpha^*)$, and $\delta(\alpha^*)$ satisfying $0 = \psi(x(\alpha^*), \alpha^*)$, $A(\alpha^*) = \mathrm{Adj}(J(\alpha^*))$, and $\delta(\alpha^*) = \mathrm{Det}(J(\alpha^*)) \neq 0$.

In summary, the system of equations (4)–(6) provides an analytically complete system of ordinary differential equations for the *nonlocal sensitivity analysis* of the original system of interest, $0 = \psi(x, \alpha)$. That is, it permits the *tracking* of the solution vector $x(\alpha)$ and the sensitivity vector $dx(\alpha)/d\alpha$, together with the adjoint $A(\alpha)$ and the determinant $\delta(\alpha)$ of the Jacobian matrix $J(\alpha)$, over any α-interval $[\alpha^*, \alpha^{**}]$ where the determinant remains nonzero.

Fully Automated Implementation

The complete differential system (4)–(6) was initially implemented in [10] by means of a Fortran program incorporating a fourth-order Adams–Moulton integration method with a Runge–Kutta start and hand-coded partial derivatives. High numerical accuracy was obtained in illustrative applications, even near critical points α where the determinant $\delta(\alpha)$ became zero. Nevertheless, hand-coding of partial derivatives was clearly an undesirable feature of the program. The partial derivative expressions in (7) involve the second order partial derivatives of $\psi(\cdot)$; and $\psi(\cdot)$ in turn could involve the partial derivatives of some still more basic function, such as the criterion function for an optimization problem. This is indeed the typical case for economic problems (e. g., the profit maximization problem handled in [10]), since such problems invariably incorporate the decision-making processes of various types of economic agents.

In consequence, a more fully *automated Fortran program for nonlocal sensitivity analysis* was eventually developed in [11]. This program is referred to as *Nasa* (an acronym for Nonlocal Automated Sensitivity Analysis); it is available for downloading as freeware from the Web site http://www.econ.iastate.edu/tesfatsi/. Nasa incorporates a fairly substantial library for the forward-mode automatic evaluation of partial derivatives through order three [13] as well as an adaptive homotopy method [12] for automatically obtaining all required initial conditions. The following sections briefly describe these features. A recent example of how Nasa has been applied to an applied general equilibrium problem in economics is detailed in [2].

Incorporation of Automatic Differentiation

Four basic approaches (see Jerrell [8] for an interesting comparative discussion of these four alternative approaches) can be used to obtain computer-generated numerical values for derivatives: hand-coding; numerical differentiation; symbolic differentiation; and automatic derivative evaluation, or *automatic differentiation* for short. Recently, *computational differentiation* has come to be the preferred term for automatic differentiation; see [1]. To avoid confusion, the more traditional term is used here. Numerical differentiation methods substitute discrete approximate forms for derivative expressions. For example, finite difference methods involve the approximation of derivatives by ratios of discrete increments; e. g., $f'(t) \approx [f(t + h) - f(t)]/h$ for some suitably small h. Symbolic differentiation methods generate exact symbolic expressions for derivatives that can be manipulated algebraically as well as evaluated numerically. In contrast, automatic differentiation methods do not generate explicit derivative expressions, either approximate or symbolic. Rather, these methods focus on the generation of derivative evaluations by breaking down the evaluation of a derivative at a given point into a sequence of simpler evaluations for functions of at most one or two variables. These evaluations are exact up to roundoff and truncation error.

For the nonlocal sensitivity analysis problem outlined above, the primary requirement is for partial derivative evaluations through order three to be obtained in a reliable and efficient manner. The use of numerical differentiation methods such as finite difference introduces systematic approximation errors into applications that can be reduced but not eliminated entirely due to the risk of catastrophic floating point error. Symbolic differentiation software packages such as Macsyma, Mathematica, and Maple produce analytical expressions for derivatives but are notorious for 'expression swell', that is, for the great many lines of code they produce for the derivative expressions of even relatively simple functional forms despite repeated use of reduction routines; see [5] for explicit examples (note that automatic differentiation has now been introduced into Maple, see [6]). Thus, an automatic derivative evaluation routine would seem to be the preferred alternative for the application at hand.

Automatic differentiation appears to have been independently developed by R.E. Moore [16] and R. Wengert [20]. The key idea of Moore and Wengert was to decompose the evaluation of complicated functions of many variables into a sequence of simpler evaluations of special functions of one or two variables, referred to below as a 'function list'. Total differentials of the special functions could be automatically evaluated along with the special function values, and partial derivatives could then be recovered from the total differentials by solving certain associated sets of linear algebraic equations.

Nonlocal Sensitivity Analysis with Automatic Differentiation, Table 1
Illustrative application of the Feed algorithm

Function List	$\partial/\partial x$	$\partial/\partial y$	$\partial^2/\partial x^2$	$\partial^3/\partial x^3$
$a = x$	1	0	0	0
$b = y$	0	1	0	0
$c = ab$	$a_x b + a b_x$	$a_y b + a b_y$	$a_{xx} b + 2 a_x b_x + a b_{xx}$	$a_{xxx} b + 3 a_{xx} b_x + 3 a_x b_{xx} + a b_{xxx}$
$d = \log(c)$	$c^{-1} c_x$	$c^{-1} c_y$	$-c^{-2} c_x^2 + c^{-1} c_{xx}$	$2 c^{-3} c_x^3 - 3 c^{-2} c_x c_{xx} + c^{-1} c_{xxx}$
$z = a + d$	$a_x + d_x$	$a_y + d_y$	$a_{xx} + d_{xx}$	$a_{xxx} + d_{xxx}$

As detailed in [1] and [4], great strides have been made over the past thirty years in developing fast and reliable automatic differentiation algorithms. The Nasa program incorporates one such algorithm, originally developed in [13], that is now referred to as *Feed* (an acronym for Fast Efficient Evaluation of Derivatives). A detailed discussion of the use of this automatic differentiation algorithm for both optimization and sensitivity analysis can be found in [9]. Total differentials are replaced by derivative arrays in order to avoid repeated function evaluations and the need to recover partial derivatives from total differentials for each successively higher-order level of differentiation.

As a simple illustration of Feed, consider the function F: $\mathrm{R}^2_{++} \to \mathrm{R}$ defined by

$$z = F(x, y) \equiv x + \log(xy)\,. \tag{8}$$

Suppose one wishes to evaluate the function value z and the partial derivatives z_x, z_y, z_{xx} and z_{xxx} at a given domain point (x, y). Consider Table 1.

The first column of Table 1 constitutes the function list for the function (8); it sequentially evaluates the function value $z = x + \log(xy)$ at the given domain point (x, y). The remaining entries in each row give the indicated derivative evaluations of the first entry in the row, using only algebraic operations. The first two rows initialize the algorithm, one row being required for each independent variable. The only input required for the first two rows is the domain point (x, y). Each subsequent row outputs a one-dimensional array of the form $(p, p_x, p_y, p_{xx}, p_{xxx})$, using the arrays obtained from previous row calculations as inputs. The final row yields the desired evaluations $(z, z_x, z_y, z_{xx}, z_{xxx})$. Note that the limitation to this collection of partial derivative evaluations is for expositional simplicity only. The evaluation of any additional desired partial derivative of z, say z_{xyy} or z_{xxxy}, can be obtained in a similar manner by suitably augmenting Table 1 with an additional column of algebraic operations.

The elements in each of the rows in Table 1 can be numerically evaluated by means of sequential calls to Feed calculus subroutines. These evaluations are exact up to roundoff and truncation error. For expositional simplicity, Table 1 only depicts evaluations for partial derivatives through order three. However, Feed calculus subroutines can in principle be constructed to evaluate the function value and the distinct partial derivatives through order k of any real-valued multivariable function that can be sequentially evaluated in a finite number of steps by means of the two-variable functions

$$\begin{aligned} &w = u + v, \quad w = u - v, \quad w = uv\,, \\ &w = \frac{u}{v}, \quad w = u^v \end{aligned} \tag{9}$$

and arbitrary nonlinear one-variable kth-order differentiable functions such as

$$\begin{aligned} &\cos(u), \quad \sin(u), \quad \exp(u), \quad c^u, \\ &\log(u), \quad au^b + c, \end{aligned} \tag{10}$$

for arbitrary constants a, b, and c. Systematic rules for constructing general kth-order calculus subroutines for special functions such as (10) are derived in [13]. References to other work focusing on recurrence relations for the derivatives of special functions such as (10) can be found, for example, in [15]. A detailed discussion of the library of Feed calculus subroutines currently incorporated into Nasa is given in [11].

The Feed algorithm thus envisions the successive transformation of arrays of partial derivatives through any specified order k into similarly-configured arrays as one forward sweep is taken through the function list for a specified kth-order differentiable function. A similar approach is proposed in [15] and [18, p. 280]. In

contrast, the partial derivative evaluation methods proposed in [17, Chapter VI, pp. 91–111] and [21] have a tree structure; that is, gradient operations are used to generate evaluations for each successively higher-order collection of partial derivatives using the results of previous gradient operations as inputs. Another approach that has attracted a great deal of interest is reverse-mode differentiation; see [3] and [7].

Automatic Initialization via Adaptive Homotopy Continuation

The initial conditions needed to integrate the complete differential system (4)–(6) from a given initial parameter point α^* consist of a solution vector $x(\alpha^*)$ together with evaluations for the adjoint $A(\alpha^*)$ and determinant $\delta(\alpha^*)$ of the Jacobian matrix $\psi_x(x(\alpha^*), \alpha^*)$. For many nonlinear problems, finding an initial solution vector is a difficult matter in and of itself.

Nasa incorporates an adaptive homotopy method [12] for automating these needed initializations. A standard (linear) homotopy method applied to the problem of finding a solution x^* for a system of equations $0 = F(x)$ proceeds by introducing a homotopy of the form

$$0 = tF(x) + [1 - t][x - c] \tag{11}$$

and solving for x as a function of t as t varies from 0 to 1 along the real line, where c represents any initial guess for the solution vector x^*. In contrast, an *adaptive homotopy* is a homotopy for which the usual continuation parameter t varying from 0 to 1 on the real line is replaced by an adaptive continuation 'agent' that makes its way by trial and error from $0 + 0i$ to $1 + 0i$ in the complex plane in accordance with certain stated objectives.

Specifically, the continuation agent designed in [12] adaptively selects a path of β values from $0 + 0i$ to $1 + 0i$ in the complex plane for the homotopy

$$0 = [F(x) - F(c)] + \beta F(c)\,, \tag{12}$$

where c again represents any initial guess for the solution vector x^*. The path for β is selected in accordance with the following multiple objective optimization problem: Reach the point $1 + 0i$ starting from the point $0 + 0i$ by taking as few steps as possible along a spider-web (spoke/hub) grid centered at $1 + 0i$ in the complex plane, but do so in a way that avoids regions where the Jacobian matrix becomes ill-conditioned.

The adaptive homotopy method introduced in [12] and incorporated into Nasa is thus an example of what might more generally be called an *adaptive computational method*, i. e., a computational method that embodies the following principle important for applied researchers: Let the computational algorithm adapt to the physical problem at hand instead of requiring users to reformulate their physical problems to conform to algorithmic requirements. For sufficiently smooth functions $F(\cdot)$, a properly constructed homotopy (e. g., a probability one homotopy as formulated in [19]) is theoretically guaranteed to have no singular points along the real continuation path from 0 to 1 for almost all initial starting points c. However, successful implementation of such homotopy methods can require a mathematically sophisticated reformulation of the user's original problem.

The homotopy (12) is solved for x as a function of β as β varies from $0 + 0i$ to $1 + 0i$ in the complex plane by making use of a complete system of ordinary differential equations analogous to the system set out in the basic problem formulation above. At each β point one obtains a solution vector $x^*(\beta)$ together with evaluations $A^*(\beta)$ and $\delta^*(\beta)$ for the adjoint and determinant of the homotopy Jacobian matrix $J^*(\beta) = F_x(x^*(\beta))$. Note that the homotopy Jacobian matrix coincides with the Jacobian matrix for the original function of interest $F(\cdot)$, implying that singularities are not artificially induced into the problem by the homotopy method per se. In principle, the solution vector $x^*(1 + 0i)$ obtained for (12) at $\beta = 1 + 0i$ yields a solution vector for the original system of interest, $0 = F(x)$. In particular, letting $F(x) \equiv \psi(x, \alpha^*)$, one obtains complete initial conditions for the original problem of interest, the nonlocal sensitivity analysis of the system $0 = \psi(x, \alpha)$ over an interval of α values starting at α^*.

See also

- ► Automatic Differentiation: Calculation of the Hessian
- ► Automatic Differentiation: Calculation of Newton Steps
- ► Automatic Differentiation: Geometry of Satellites and Tracking Stations

- ▶ Automatic Differentiation: Introduction, History and Rounding Error Estimation
- ▶ Automatic Differentiation: Parallel Computation
- ▶ Automatic Differentiation: Point and Interval
- ▶ Automatic Differentiation: Point and Interval Taylor Operators
- ▶ Automatic Differentiation: Root Problem and Branch Problem
- ▶ Parametric Global Optimization: Sensitivity
- ▶ Sensitivity Analysis of Complementarity Problems
- ▶ Sensitivity Analysis of Variational Inequality Problems
- ▶ Sensitivity and Stability in NLP
- ▶ Sensitivity and Stability in NLP: Approximation
- ▶ Sensitivity and Stability in NLP: Continuity and Differential Stability

References

1. Berz M, Bischof Ch, Corliss G, Griewank G (1996) Computational differentiation techniques, applications, and tools. SIAM, Philadelphia
2. Dakhlia S (1999) Testing for a unique equilibrium in applied general equilibrium models. J Econom Dynam Control 23:1281–1297
3. Griewank A (1989) On automatic differentiation. In: Iri M, Tanabe K (eds), Mathematical Programming: Recent Developments and Applications. Kluwer, Dordrecht, pp 83–108
4. Griewank A, Corliss G (1991) Automatic differentiation of algorithms: Theory, implementation, and application. SIAM, Philadelphia
5. Hayes K, Hirschberg J, Slottje D (1987) Computer algebra: Symbolic and algebraic computation in economic/econometric applications. In: Adv Econometrics, vol 6. JAI Press, pp 51–89
6. Heck A (1993) Introduction to Maple. Springer, Berlin
7. van Iwaarden R (1993) Automatic differentiation applied to unconstrained nonlinear optimization with result verification. Interval Comput 4:30–41
8. Jerrell ME (1997) Automatic differentiation and interval arithmetic for estimation of disequilibrium models. Comput Economics 10:295–316
9. Kagiwada H, Kalaba R, Rasakhoo N, Spingarn K (1985) Numerical derivatives and nonlinear analysis. Plenum, New York
10. Kalaba R, Tesfatsion L (1981) Complete comparative static differential equations. Nonlinear Anal Th Methods Appl 5:821–833
11. Kalaba R, Tesfatsion L (1990) Nonlocal automated sensitivity analysis. Comput Math Appl 20:53–65
12. Kalaba R, Tesfatsion L (1991) Solving nonlinear equations by adaptive homotopy continuation. Appl Math Comput 41:99–115
13. Kalaba R, Tesfatsion L, Wang J-L (1983) A finite algorithm for the exact evaluation of higher-order partial derivatives of functions of many variables. J Math Anal Appl 92:552–563
14. Kalaba R, Zagustin E, Holbrow W, Huss R (1977) A modification of Davidenko's method for nonlinear systems. Comput Math Appl 3:315–319
15. Kedem G (1980) Automatic differentiation of computer programs. ACM Trans Math Softw 6:150–165
16. Moore RE (1962) Interval arithmetic and automatic error analysis in digital computing. PhD Thesis, Dept. Computer Sci., Stanford Univ
17. Rall L (1981) Automatic differentiation: Techniques and applications. Springer, Berlin
18. Rall L (1986) The arithmetic of differentiation. Math Magazine 59:275–282
19. Watson LT, Sosonkina M, Melville RC, Morgan AP, Walker HF (1997) HOMPACK90: A suite of FORTRAN 90 codes for globally convergent homotopy algorithms. ACM Trans Math Softw 23:514–549
20. Wengert R (1964) A simple automatic derivative evaluation program. Comm ACM 7:463–464
21. Wexler A (1988) An algorithm for exact evaluation of multivariate functions and their derivatives to any order. Comput Statist Data Anal 6:1–6

Nonoriented Multicommodity Flow Problems

PIERRE CHARDAIRE[1], ABDEL LISSER[2]

[1] University East Anglia, Norwich, UK

[2] Recherche et Developpement, France Telecom, Issy les Moulineaux, France

MSC2000: 90B10, 90C05, 90C06, 90C35

Article Outline

Keywords

Network; Multicommodity flow; Decomposition algorithms

The routing of traffic requirements is one of the important problems that have to be solved when designing a telecommunication network. Another important

problem is the computation of standby capacities that must be set up to enable the rerouting of the traffic requirements affected by any network failure of some given types. Both problems can be formulated as nonoriented multicommodity network flow models sometimes also called *undirected multicommodity network flow models*. The routing problem captures the main aspects of such models and is therefore considered in more details hereafter. The reader interested in the computation of standby capacities is referred to [11,12].

A transmission network can be viewed as an undirected graph. An edge (i. e. a link of the network) is characterized by a pair of nodes, a transmission cost per circuit and a transmission capacity in the number of circuits. The *routing problem* is the determination of the most economical way of using the available transmission capacities to route a traffic matrix (a number of circuits between various pairs of nodes) through the network. This problem can be expressed as follows:

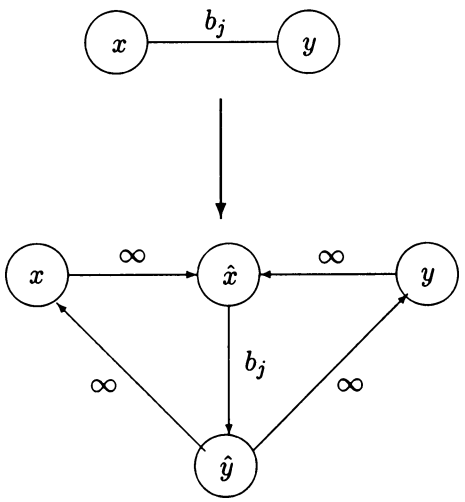
$$\begin{cases} \min & \sum_{k,j} c_j^k \left| x_j^k \right| \\ \text{s.t.} & Ax^k = r^k, \quad k \in K, \\ & \sum_k \left| x_j^k \right| \le b_j, \quad j \in J, \end{cases}$$

where:

- A is the node-arc incidence matrix corresponding to an arbitrary orientation of the network graph (i. e. $A_{ij} = +1$ if arc j is directed away from node i, $A_{ij} = -1$ if arc j is directed towards node i, $A_{ij} = 0$ otherwise);
- $J = \{1, \ldots, \overline{J}\}$ is the set of arcs of the network;
- k is a commodity characterized by a number of circuits, d^k, to be routed through the network between a given pair of nodes, s^k and t^k;
- $K = \{1, \ldots, \overline{K}\}$ is the set of commodities to be routed;
- r^k is the requirement vector for commodity k (i. e. $r^k_{s^k} = d^k$, $r^k_{t^k} = -d^k$ and $r^k_l = 0$ when $l \notin \{s^k, t^k\}$);
- x^k is the flow vector for a given commodity k (i. e. x_j^k is the flow on arc $j = (s, t)$ with $x_j^k > 0$ if the flow goes from s to t and $x_j^k < 0$ if the flow goes from t to s);
- b_j is the capacity of link j;
- c_j^k is the cost per unit of flow on link j for commodity k.

The first group of constraints contains one block of constraints per commodity. These so-called *flow constraints* specify that x is a routing of the traffic requirements. The second group of constraints ties together the various commodities. These *coupling constraints* specify that only a limited number of circuits can be routed through any given edge of the network. The above formulation is called a *node-arc formulation of the problem*.

If we assume that cost coefficient c_j^k are non negative, this nonlinear problem can be replaced with an equivalent standard (linear) multicommodity flow problem by using the transformation of the network graph depicted in the following figure where the figures over edges and arcs denote capacities.



The structure of the resulting model is similar to that of the original model except that absolute values are removed and nonnegativity restrictions on the flow variables are introduced. The transformation is justified by the fact that the support of the flow associated with any given commodity does not contain any directed cycle in the optimal solution of the transformed problem. The transformed model could be solved by any code designed for the solution of directed multicommodity flow problems. However, as this model is much larger than the original nonlinear model a better approach has to be devised when a computationally efficient solution method is sought for. Assuming that $c_j^k > 0$, one can introduce nonnegative variables, x_j^{k+} and x_j^{k-}, such that $x_j^k = x_j^{k+} - x_j^{k-}$ and $|x_j^k| = x_j^{k+} + x_j^{k-}$ to obtain an equiv-

alent linear formulation [11]:

$$\begin{cases} \min & \sum_{k,j} c_j^k \left(x_j^{k+} + x_j^{k-} \right) \\ \text{s.t.} & A\left(x^{k+} - x^{k-}\right) = r^k, \quad \forall k, \\ & \sum_{k} \left(x_j^{k+} + x_j^{k-} \right) \le b_j, \quad \forall j \in J, \\ & x_j^{k+} \ge 0, \quad \forall k \in K, \quad \forall j \in J, \\ & x_j^{k-} \ge 0, \quad \forall k \in K, \quad \forall j \in J. \end{cases}$$

For real networks such as the Paris district transmission network, the number of nodes, $\bar{I}$, can be larger than 100 and the number of links larger than 300. If one assumes that there can be a traffic requirement between any pair of nodes, the size of the problem can be larger than approximately $M \approx 5 \times 10^5$ constraints and $N \approx 3 \times 10^6$ variables, since $\bar{K} = \bar{I}(\bar{I}-1)/2$ in this case. However, the formulation can be often simplified and an equivalent smaller problem solved. In fact, when the cost per unit of flow on a given link does not depend on the commodity (i. e. $c_j^k = c_j^1$ for any k) all the commodities which have a common endpoint can be 'merged'. More precisely, consider a particular node, i, and the set, K_i, of commodities, k, such that $s_k = i$ or $t_k = i$. As the orientation of the commodities is arbitrary the commodities in K_i can be replaced with a single commodity characterized by a requirement vector, r, of components, $r_i = \sum_{k \in K_i} d_k$, $r_j = -d^k$ for all $j \in \cup_{k \in K_i} \{s_k, t_k\} - \{i\}$ and $r_j = 0$ for all $j \notin \cup_{k \in K_i} \{s_k, t_k\}$. A merging of commodities that minimizes the number of merged commodities can be found by solving a minimum node cover problem in a nondirected graph (cf. ► Network location: Covering problems) where the vertices correspond to the endpoints of the commodities and the edges correspond to the commodities. In practice a satisfactory solution to this problem can be obtained by a greedy algorithm where at each iteration the node that covers the larger number of edges not yet covered is selected. In any case, the problem reduces to a number of commodities not larger than $\bar{I}$. Assuming that $\bar{K} = \bar{I}$, one obtains $M \approx 10{,}000$ constraints and $N \approx 60{,}000$ variables. This size of problem is within the reach of modern general purpose linear programming codes. However, if extra constraints were to be included in the initial formulation the merging of the commodities might not be any longer possible. For example, this would be the case if quality of service constraints were introduced to impose that no more than p percent of any traffic requirement is routed on any link. Constraints of the form $x_j^{k+} + x_j^{k-} \le p d_k/100$ would have to be added to the above formulation. The size of this new model would be huge.

Fortunately, specialized algorithms that exploit both the block structure of the problem and the structure of each block of flow constraints can be designed to provide efficient solution methods. A primal partitioning simplex algorithm specialization is presented in [3]. Specializations based on price-directive decomposition (cf. ► Branch and price: Integer programming with column generation), resource-directive decomposition and partitioning of linear systems (simplex and dual affine scaling algorithm) are reviewed in [5]. Technical details on specializations as well as comparative experiments are presented in [4].

It is worth giving more details on one of the most attractive specialization method obtained by applying the Dantzig–Wolfe decomposition principle to the above formulation. The advantage of this specialization is threefold. First, it leads to a reformulation of the problem that appeals to network planners (as the concept of routing path is more directly part of the model) and is easily understood without any references to the Dantzig–Wolfe decomposition principle. Secondly, it can be easily implemented using modern linear programming libraries. Thirdly, and more importantly it is computationally efficient for real-life instances that are usually weakly constrained (no more than 10% edges saturated at optimality) (see [4]).

By applying the Dantzig–Wolfe decomposition principle it can be shown that the routing problem can be reformulated as follows [5]:

$$\begin{cases} \min & \sum_{k,l} [c^k]^\top v_l^k \alpha_l^k \\ \text{s.t.} & \sum_{l} \alpha_l^k = d^k, \quad k \in K, \\ & \sum_{k,l} \alpha_l^k v_l^k \le (b_1, \ldots, b_{\bar{J}})^\top, \\ & \alpha^k \ge 0, \quad k \in K. \end{cases}$$

v_l^k is the characteristic vector of a path, p_l^k, that connects s_k to t_k in the undirected network graph, i. e. the components j of v_l^k is 1 if link j belongs to p_l^k and is 0 otherwise. $[c^k]^\top v_l^k$ is the cost of p_l^k per unit of flow routed on that path (i. e. the sum of the costs c_j^k of its links) and

α_l^k is the amount of traffic routed on path p_l^k. The first set of constraints expresses that d_k units of traffic have to be routed between s_k and t_k whereas the second set of constraints expresses the limitation on the capacity available on the links to route the traffic. The formulation above is known as the *node-path formulation* of the multicommodity flow problem.

This formulation has an exponential number of columns but it is not necessary to express all the columns of this linear program. Indeed, the reduced cost of variable α_l^k is $w_l^k = [c^k + \lambda]^\intercal \nu_l^k - \pi_k$, where $-\lambda$ is the vector of simplex multipliers associated with the capacity constraints and π_k is the vector of simplex multipliers associated with the constraint corresponding to commodity k in the first group of constraints. Therefore $\min_l w_l^k$ can be obtained by computing a shortest path between s_k and t_k in the undirected network graph where the edge lengths are given by the vector $c^k + \lambda$.

In the decomposition algorithm the formulation given above is called the (*full*) *master program*. The $\overline{K}$ shortest path problems which are solved to generate columns of the master program are called the *satellite problems*. At each stage of the algorithm a *reduced master program*, i. e. a program that contains a subset of the columns of the master program, is solved. The first reduced master program is initialized with a set of columns corresponding to a basic matrix and the columns corresponding to the slack variables. If no such initial set of columns is available, artificial variables are added to the formulation and a first phase or a big-M method have to be used to drive the program to a feasible basis. After solving a reduced master program to optimality, the $\overline{K}$ satellite problems are solved to check whether the solution obtained is optimal for the full master program. If the reduced costs corresponding to the solution of the satellite problems are nonnegative the solution to the reduced master program is an optimal solution to the full master program, otherwise the master program is increased with the new columns candidate for basis entry and the simplex algorithm goes on. After the solution of a reduced master program the reduced cost of all variables in that program are nonnegative. Hence, as λ is the reduced cost vector associated with the slack variables of the coupling constraints we deduce that $\lambda \geq 0$ and that in turn the edge lengths in the satellite problems are nonnegative. The satellite problems can therefore be solved using the Dijkstra algorithm (see [9] and [13] for details of efficient implementations).

The solution of each master program can be made more efficient by using a refinement of the GUB specialization of the simplex algorithm (see [4] for details). Some variations of the algorithm may be also applied. For example, some columns generated may be discarded from the reduced master program at later stages, the generation of new columns may be carried out before optimality is reached provided that the simplex multipliers λ are nonnegative.

Finally it is worth mentioning that a new cutting plane technique based on interior point method and called the *analytic center cutting plane method* (ACCPM) was recently proposed to solve large scale convex optimization problems [8]. Its application to the dual of the above formulation gives performances that are roughly similar to those of the Dantzig–Wolfe decomposition algorithm [4] but ACCPM is much more efficient for highly nonlinear problems [7].

Further details on network programming and modeling can be found in [1,2,6,10].

See also

- ► Maximum Flow Problem
- ► Minimum Cost Flow Problem
- ► Multicommodity Flow Problems
- ► Nonconvex Network Flow Problems
- ► Piecewise Linear Network Flow Problems

References

1. Ahuja R, Magnanti T, Orlin J (1992) Network flows. Prentice-Hall, Englewood Cliffs
2. Bazaraa MS, Jarvis JJ (1977) Linear programming and network flows. Wiley, New York
3. Chardaire P, Lisser A (1996) A primal simplex specialization for solving non-oriented multicommodity flow problems. Invest Oper 5(2–3):117–152
4. Chardaire P, Lisser A (1999) Simplex and interior point specialized algorithms for solving non-oriented multicommodity flow problems. Oper Res
5. Chardaire P, Lisser A (2000) Minimum cost multicommodity flow. In: Pardalos PM, Resende M (eds) Handbook Applied Optim. Oxford Univ. Press, Oxford
6. Glover F, Klingman D, Phillips NV (1992) Network models in optimization and their applications in practice. Wiley, New York

7. Goffin J-L, Gondzio J, Sarkissian R, Vial J-P (1996) Solving non linear multicommodity flow problems by the analytic center cutting plane method. Math Program 76:131–154
8. Goffin J-L, Haurie A, Vial J-P (1992) Decomposition and nondifferentiable optimization with the projective algorithm. Manag Sci 38
9. Johnson E (1972) On shortest path and sorting. Proc. Annual Conf. ACM, Boston, p 510
10. Kennington JL, Helgason RV (1980) Algorithms for network programming. Wiley, New York
11. Minoux M (1986) Mathematical programming: Theory and algorithms. Wiley, New York
12. Minoux M, Serrault JY (1980) Subgradient optimization and large scale programming application to multicommodity network synthesis with security constraints. RAIRO 15(2):185–203
13. Williams JWJ (1964) Algorithm 232, heapsort. Comm ACM 7(6):347–348

Nonsmooth Analysis: Fréchet Subdifferentials

Alexander Y. Kruger
Centre for Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Ballarat, Australia

MSC2000: 49K27, 90C48, 58C20, 58E30

Article Outline

Keywords

Nonsmooth analysis; Variational analysis; Subdifferential; Normal cone; Optimality conditions; Extremal principle; Multifunction; Asplund space

Introduction

Being natural generalizations of the classical Fréchet derivative and the subdifferential in the sense of convex analysis, *Fréchet subdifferentials* have been known for more than 30 years. They were probably first introduced in finite dimensions in [1] (under the name "lower semidifferentials"). Some of their properties in the infinite-dimensional setting were investigated in [18,21]. During the first decade Fréchet subdifferentials were not widely used because of rather poor (direct) calculus. They mostly served as building blocks for more sophisticated *limiting (Fréchet) subdifferentials* [19,23,30,31].

The discovery of the "fuzzy rules" in the 1980s [11,12,15,29] revitalized interest in Fréchet subdifferentials. It was shown that calculus results and optimality conditions can be formulated in terms of Fréchet and other "simple" subdifferentials computed not at the given point, but at some points arbitrarily close to it, thus incorporating "differential" properties of the function at nearby points. Such results are actually at the core of the corresponding statements for limiting subdifferentials.

Being the smallest among all "simple" subdifferentials with reasonable properties, the Fréchet subdifferentials have proved to be convenient tools for the analysis of nondifferentiable functions on *Asplund* spaces, a very important subclass of general Banach spaces. Furthermore, the main fuzzy results in terms of Fréchet subdifferentials present characterizations of Asplund spaces themselves [6,12,13,34,35,38,45].

The article contains no proofs. A more detailed survey of Fréchet subdifferentials can be found in [25].

Mostly standard notations are used throughout the article. X and Y denote normed linear spaces and X^* and Y^* denote their topological duals. $\langle \cdot, \cdot \rangle$ is a bilinear form defining a canonical paring between a space and its dual. $B_\rho(x)$ stands for a closed ball with center x and radius ρ. We write B_ρ instead of $B_\rho(0)$ and just B if $\rho = 1$ (unit ball).

Definitions

Fréchet Subdifferentials

Let $f : X \to \mathbb{R}_\infty = \mathbb{R} \cup \{+\infty\}$, $f(x) < \infty$. The set

$$\partial f(x) = \left\{ x^* \in X^* : \liminf_{u \to x} \frac{f(u) - f(x) - \langle x^*, u - x \rangle}{\|u - x\|} \geq 0 \right\} \quad (1)$$

is called the *Fréchet subdifferential* of f at x. This set is (norm) closed and convex. The next proposition shows that it generalizes the notions of the Fréchet derivative and the subdifferential in the sense of convex analysis.

Proposition 1

1. *If f is Fréchet-differentiable at x with the derivative $\nabla f(x)$ then $\partial f(x) = \{\nabla f(x)\}$.*
2. *If f is convex then*

$$\partial f(x) = \left\{ x^* \in X^* : f(u) - f(x) \geq \langle x^*, u - x \rangle, \forall u \in X \right\}.$$

Note that the Fréchet subdifferential does not change if another equivalent norm on X is used in (1).

Example 1 The set (1) can be empty. Take $f : \mathbb{R} \to \mathbb{R} : f(u) = -|u|$, $u \in \mathbb{R}$.

One can also consider the *Fréchet superdifferential*

$$\partial^+ f(x) = \left\{ x^* \in X^* : \limsup_{u \to x} \frac{f(u) - f(x) - \langle x^*, u - x \rangle}{\|u - x\|} \leq 0 \right\}. \quad (2)$$

While the set (1) consists of linear continuous functionals "supporting" f from below, the functionals from (2) "support" f from above. Unlike the classical case, the existence of two different derivative-like objects is quite natural for nonsmooth analysis: "differential" properties of a function "from below" and "from above" could be essentially different.

Subdifferentials and superdifferentials are related by the equality

$$\partial(-f)(x) = -\partial^+ f(x). \quad (3)$$

Surely, in the nondifferentiable case at least one of the sets (1) and (2) must be empty.

Proposition 2 *$\partial f(x) \neq \emptyset$ and $\partial^+ f(x) \neq \emptyset$ if and only if f is Fréchet-differentiable at x. In this case one has $\partial f(x) = \partial^+ f(x) = \{\nabla f(x)\}$.*

Example 2 Both sets (1) and (2) can be empty simultaneously. Take $f : \mathbb{R} \to \mathbb{R} : f(u) = u \sin(1/u)$ if $u \neq 0$, and $f(0) = 0$.

Example 3 The fact that the set (1) is a singleton does not imply differentiability. Take $f : \mathbb{R} \to \mathbb{R} : f(u) = \max(u \sin(1/u), 0)$ if $u \neq 0$, and $f(0) = 0$. Then f is nondifferentiable at 0, although one evidently has $\partial f(0) = \{0\}$.

Example 4 Fréchet differentiability is essential in Proposition 1 Gâteaux differentiable functions can be nonsubdifferentiable in the Fréchet sense. Take $f : \mathbb{R}^2 \to \mathbb{R} : f(u_1, u_2) = -\sqrt{|u_1|^2 + |u_2|^2}$ if $u_2 = u_1^2$, $f(u_1, u_2) = 0$ otherwise. The Gâteaux derivative of f is 0, while $\partial f(0) = \emptyset$.

Remark 1 One can define the *Gâteaux subdifferential* on the basis of the notion of the Gâteaux differentiability. For this subdifferential, the analogs of Propositions 1 and 2 and some other results hold true. Considering Gâteaux (and other types of) "simple" subdifferentials can be useful in some applications. The Gâteaux subdifferential always contains the Fréchet subdifferential.

If $\dim X < \infty$ the Fréchet subdifferential can be expressed equivalently in terms of certain generalized directional derivatives [1,16,18,39,42,43].

Fréchet Normal Cone

Now consider a set $\Omega \subset X$ and let $x \in \Omega$. Similarly to definition (1) of the Fréchet subdifferential one can define the *Fréchet normal cone*

$$N(x|\Omega) = \left\{ x^* \in X^* : \limsup_{u \overset{\Omega}{\to} x} \frac{\langle x^*, u - x \rangle}{\|u - x\|} \leq 0 \right\} \quad (4)$$

to Ω at x. Here $u \overset{\Omega}{\to} x$ means that $u \to x$ with $u \in \Omega$.

It is a norm closed and convex cone closely related to the subdifferential defined above. It is actually the Fréchet subdifferential $\partial \delta_\Omega(x)$ of the indicator function

δ_Ω of Ω ($\delta_\Omega(u) = 0$ if $u \in \Omega$ and $\delta_\Omega(u) = \infty$ otherwise).

This fact allows one to deduce some properties of normal cones from the corresponding statements about subdifferentials. Thus, it follows from Proposition 1, that the normal cone (4) generalizes the corresponding notion of convex analysis.

In finite dimensions the Fréchet normal cone coincides with the polar of the *tangent* (*contingent*, *Bouligand tangent*) cone to Ω at x. If X is reflexive it coincides with the polar of the *weak tangent* cone [1,2,9,14,43].

The relationship between Fréchet subdifferentials and normal cones is bilateral. For a function $f : X \to \mathbb{R}_\infty$ one can consider its *epigraph* $\mathrm{epi} f = \{(u, \mu) \in X \times \mathbb{R} : f(u) \le \mu\}$. If the norm on $X \times \mathbb{R}$ is compatible with that on X (that is, $\|(x, 0)\| = \|x\|$), then the following equivalent definition of the Fréchet subdifferential holds true:

$$\partial f(x) = \left\{x^* \in X^* : (x^*, -1) \in N(x, f(x)|\mathrm{epi} f)\right\} . \tag{5}$$

"Horizontal" normals to the epigraph can also be of interest. They define the *singular Fréchet subdifferential* of f at x:

$$\partial^\infty f(x) = \left\{x^* \in X^* : (x^*, 0) \in N(x, f(x)|\mathrm{epi} f)\right\} .$$

Of course, if f is *calm* at x [7,42], that is, $\|f(u)-f(x)\| \le l\|u-x\|$ for some $l > 0$ and for all u in a neighborhood of x, then the latter set is empty.

Strict Fréchet δ-Subdifferentials

As mentioned in "Introduction," Fréchet subdifferentials have poor calculus and their direct application has been rather limited. There exists a way of enriching the properties of the subdifferentials. It consists in considering differential properties of a function at nearby points.

Consider a new derivative-like object based on the Fréchet subdifferential:

$$\hat\partial_\delta f(x) = \bigcup_{\substack{u \in B_\delta(x) \\ |\mathrm{cl} f(u) - f(x)| \le \delta}} \partial(\mathrm{cl} f)(u). \tag{6}$$

It depends on some positive δ. cl f denotes here the lower semicontinuous envelope of f (its epigraph is the closure of the epigraph of f in $X \times \mathbb{R}$). Unlike (1) the set (6) can be nonconvex. It is called the *strict Fréchet δ-subdifferential* of f at x [24].

The *strict Fréchet δ-superdifferential* $\hat\partial^+_\delta f(x)$ of f at x can be defined in a similar way. The equality $\hat\partial^+_\delta f(x) = -\hat\partial_\delta(-f)(x)$ holds true. The strict subdifferentials and superdifferentials can be nonempty simultaneously and can be essentially different. The set $\hat\partial^0_\delta \varphi(x) = \hat\partial_\delta f(x) \cup \hat\partial^+_\delta f(x)$ can be useful in some situations. It is called the *strict Fréchet δ-differential* of f at x.

The *strict Fréchet δ-normal cone* to a set Ω at $x \in \Omega$ is defined similarly:

$$\hat N_\delta(x|\Omega) = \bigcup_{u \in \mathrm{cl}\Omega \cap B_\delta(x)} N(u|\mathrm{cl}\Omega) .$$

The goal of introducing strict Fréchet δ-subdifferentials is mainly notational. They are convenient for formulating "fuzzy" results, but such results can certainly be formulated in terms of ordinary Fréchet subdifferentials.

Limiting Subdifferentials

The limiting Fréchet subdifferentials are defined as limits of "simple" ones [23,30,31,34]. To simplify the definitions we assume in this subsection that $f : X \to \mathbb{R}_\infty$ is lower semicontinuous in a neighborhood of x.

The *limiting Fréchet subdifferential* of f at x is defined as

$$\bar\partial f(x) = \{x^* \in X^* : \exists \quad \text{sequences} \\ \{x_k\} \subset X, \{x^*_k\} \subset X^* \text{such that} \\ x_k \xrightarrow{f} x,\ x^*_k \xrightarrow{w^*} x^* \text{ and } x^*_k \in \partial f(x_k),\ k = 1, 2, \dots\} . \tag{7}$$

The notations $x_k \xrightarrow{f} x$ and $x^*_k \xrightarrow{w^*} x^*$ here mean, respectively, that $x_k \to x$ with $f(x_k) \to f(x)$ (*f-attentive convergence* [42]), and x^*_k converges to x^* in the *weak* * topology of X^*.

$\bar\partial f(x)$ is a weakly* sequentially closed set in X^*. In general it is nonconvex. If f is strictly differentiable at x the set (7) reduces to the derivative.

Using strict δ-subdifferentials, one can rewrite (7) in the following way:

$$\bar{\partial} f(x) = \bigcap_{\delta>0} \mathrm{cl}^* \hat{\partial}_\delta f(x) ,$$

where cl* denotes the weak* sequential closure.

Other limiting objects (the *limiting superdifferential*, the *limiting differential*, the *limiting normal cone*, the *singular limiting subdifferential*, and the *limiting coderivative*) can be defined in a similar way.

Thus, the limiting normal cone to a closed set Ω is defined by the equality

$$\bar{N}(x|\Omega) = \bigcap_{\delta>0} \mathrm{cl}^* \hat{N}_\delta(x|\Omega) .$$

It coincides with the limiting subdifferential of the indicator function of Ω. The analog of (5) is also valid:

$$\bar{\partial} f(x) = \{x^* \in X^* : (x^*, -1) \in \bar{N}(x, f(x)|\mathrm{epi} f)\} .$$

The limiting subdifferentials and normal cones have been well investigated. They possess good calculus (which is the consequence of the fuzzy calculus of Fréchet subdifferentials; see [19,23,30,31,32,34,36,42] for the properties of these objects and some examples). They have proved to be very efficient for formulating optimality conditions in nonsmooth optimization [20,22,29,30,31,32,34,36], especially in finite dimensions. When applying limiting subdifferentials in infinite dimensional spaces, one must be careful about nontriviality of the limits in the weak* topology. Additional regularity conditions are needed (*compact epi-Lipschitzness*, *sequential normal compactness*, *partial sequential normal compactness* [4,34,36], etc.)

Fréchet ε-Subdifferentials and ε-Normals

In some cases it can be convenient to use ε-extensions of the Fréchet subdifferentials and normal cones [21,23,29,44]. For instance, the *Fréchet ε-subdifferential* and the *Fréchet ε-superdifferential* of f at x are defined as

$$partial_\varepsilon f(x) = \left\{ x^* \in X^* : \liminf_{u \to x} \frac{f(u) - f(x) - \langle x^*, u - x\rangle}{\|u - x\|} \geq -\varepsilon \right\} ,$$

$$\partial_\varepsilon^+ f(x) = \left\{ x^* \in X^* : \limsup_{u \to x} \frac{f(u) - f(x) - \langle x^*, u - x\rangle}{\|u - x\|} \leq \varepsilon \right\} .$$

Unlike (1) and (2), these sets depend on the specific norm on X (when $\varepsilon > 0$).

The next two propositions extend Propositions 1 and 2 respectively.

Proposition 3 *If f is convex then* $\partial_\varepsilon f(x) = \partial f(x) + \varepsilon B^* = \{x^* \in X^* : f(u) - f(x) \geq \langle x^*, u - x\rangle - \varepsilon\|u - x\|, \ \forall u \in X\}$.

Remark 2 Note that the above ε-subdifferential differs from the corresponding notion of convex analysis, which is usually defined [41] as the set of all $x^* \in X^*$, such that $f(u) - f(x) \geq \langle x^*, u - x\rangle - \varepsilon$ for all $u \in X$.

Proposition 4 *If $x_1^* \in \partial_{\varepsilon_1} f(x)$, $x_2^* \in \partial_{\varepsilon_2}^+ f(x)$, $\varepsilon_1 \geq 0$, $\varepsilon_2 \geq 0$ then $\|x_1^* - x_2^*\| \leq \varepsilon_1 + \varepsilon_2$.*

Formulation

Direct Calculus

The propositions below present some simple calculus results for Fréchet subdifferentials. Most of them follow directly from the definitions. More advanced statements of fuzzy calculus are presented in the next subsections.

Proposition 5 *If f attains a local minimum at x then* $0 \in \partial f(x)$.

Proposition 6 $\partial(\lambda f)(x) = \lambda \partial f(x)$ *for any* $\lambda > 0$.

Proposition 7 *Let $f_1, f_2 : X \to \mathbb{R}_\infty$ be finite at x. Then*

$$\partial(f_1 + f_2)(x) \supset \partial f_1(x) + \partial f_2(x) . \tag{8}$$

The above proposition presents an example of a *Sum Rule*. Usually the sum rule is the central result of any subdifferential calculus. Unfortunately, the inclusion (8) is almost useless: it does not allow one to decompose elements of the subdifferential of the sum of functions in terms of elements of subdifferentials of the original functions. Simple examples show that inclusion (8) can be strict even in the convex case. The next proposition gives two important cases when the equality holds true in (8).

Proposition 8 *Let $f_1, f_2 : X \to \mathbb{R}_\infty$ be finite at x.*

1. *If f_1 and f_2 are convex and one of them is continuous at x then*

$$\partial(f_1 + f_2)(x) = \partial f_1(x) + \partial f_2(x).$$

2. *If f_1 is Fréchet-differentiable at x then*

$$\partial(f_1 + f_2)(x) = \nabla f_1(x) + \partial f_2(x)\ . \tag{9}$$

Part 1 of Proposition 8 is known as the Moreau–Rockafellar theorem [41]. Part 2 is a simple corollary of Proposition 7. It is an interesting example, when an inclusion implies an equality. Indeed, applying Proposition 7 to the sum of the functions $f_1 + f_2$ and $-f_2$ and making use of (3), one gets

$$\partial f_2(x) \supset \partial(f_1 + f_2)(x) - \partial^+ f_1(x)\ . \tag{10}$$

Taking into account Proposition 2, the inclusions (8) and (10) imply (9).

Proposition 8 yields a simple necessary optimality condition generalizing Proposition 5.

Proposition 9 *Let $f_1 : X \to \mathbb{R}$ be Fréchet-differentiable at x where $f_2 : X \to \mathbb{R}_\infty$ is finite. If $f_1 + f_2$ attains a local minimum at x then $-\nabla f_1(x) \subset \partial f_2(x)$.*

The next two assertions are corollaries of Propositions 7 and 9.

Proposition 10 *Let $x \in \Omega_1 \cap \Omega_2$.*
Then $N(x|\Omega_1 \cap \Omega_2) \supset N(x|\Omega_1) + N(x|\Omega_2)$.

Proposition 11 *Let f be Fréchet-differentiable at x. If f attains at x a local minimum on Ω then $-\nabla f(x) \in N(x|\Omega)$.*

For a set $\Omega \subset X$ and $u \in X$ consider the distance $d_\Omega(u) = \inf_{\omega\in\Omega} \|u - \omega\|$.

Proposition 12 ([18]) *For any $x \in \Omega$ one has $\partial d_\Omega(x) = \{x^* \in N(x|\Omega) : \|x^*\| \le 1\}$.*

Strict Differentiability

Recall that f is called *strictly differentiable* [34,42] at x (with the strict derivative $\nabla f(x)$) if

$$\lim_{\substack{u \to x,\, u' \to x \\ u \ne u'}} \frac{f(u') - f(u) - \langle \nabla f(x), u' - u\rangle}{\|u' - u\|} = 0\ .$$

In the case of a strictly differentiable function the Fréchet subdifferentials and superdifferentials at nearby points cannot differ much from the strict derivative.

Proposition 13 ([25]) *If f is strictly differentiable at x with the derivative $\nabla f(x)$ then for any $\varepsilon > 0$ there exists a $\delta > 0$ such that:*

1. *$\hat\partial^0_\delta f(x) \subset \nabla f(x) + \varepsilon B^*$.*
2. *$\nabla f(x) \in \partial_\varepsilon f(u) \cap \partial^+_\varepsilon f(u)$ for all $u \in B_\delta(x)$.*

The rest of the section is devoted to "fuzzy" results in terms of Fréchet subdifferentials and strict Fréchet δ-differentials.

Variational Principles

The *variational principles* by Ekeland [10], Borwein and Preiss [3] as well as their subsequent followers [6,8,34,40,42] are very powerful tools of modern variational analysis. They make it possible to substitute an "almost minimal" point (up to ε) by another point, arbitrarily close to the initial one, which is the local minimizer for a slightly perturbed (usually by adding a small term) function. Thus, such principles can be viewed as fuzzy results.

The next assertion is valid for an arbitrary Asplund space. It is known as the *Subdifferential Variational Principle*. Let us recall that a Banach space is called *Asplund* [6,8,34,40] if any continuous convex function on it is Fréchet-differentiable on a dense G_δ set of points. Asplund spaces form a rather broad subclass of Banach spaces. It includes, for instance, all spaces which admit Fréchet-differentiable bump functions (in particular, Fréchet smooth spaces). Reflexive spaces are examples of Fréchet smooth spaces.

Asplund spaces provide a very convenient framework for investigating "differential" properties of nonsmooth functions. Actually the Asplund property of a Banach space is not only sufficient but also a necessary condition for the fulfillment of some basic results in nonsmooth analysis involving Fréchet normals and subdifferentials (see [6,13,34,35,38] and the statements below).

Theorem 1 (Mordukhovich and Wang [38]) *Let X be Asplund, $f : X \to \mathbb{R}_\infty$ be lower semicontinuous and bounded below, $\varepsilon > 0$, $\lambda > 0$. Suppose that $f(x) < \inf f + \varepsilon$. Then there exists a $u \in B_\lambda(x)$ and an $x^* \in \partial f(u)$, such that $f(u) \le f(x)$ and $\|x^*\|_* < \varepsilon/\lambda$.*

The following theorem states that the class of Fréchet *subdifferentiability spaces* [15] coincides with Asplund spaces.

Theorem 2 *The following assertions are equivalent:*
1. *X is an Asplund space.*
2. *For any lower semicontinuous function $f : X \to \mathbb{R}_\infty$ the set $\{u \in X : \partial f(u) \neq \emptyset\}$ is dense in $\mathrm{dom} f$.*
3. *For any lower semicontinuous function $f : X \to \mathbb{R}_\infty$ there exists an $x \in \mathrm{dom} f$ such that $\partial f(x) \neq \emptyset$.*

Sum Rules

After the sum rule was first established in the limiting form in [19] (see [23,31]) the fuzzy versions were derived in [12,15] (see also [6,17,37]). Now two main versions of the fuzzy sum rule are known. For simplicity they are formulated below in terms of strict δ-subdifferentials.

Rule 1 Weak Fuzzy Sum Rule *Let $f_1, f_2 : X \to \mathbb{R}_\infty$ be finite at x and lower semicontinuous near x. Then $\partial(f_1 + f_2)(x) \subset \hat{\partial}_\delta f_1(x) + \hat{\partial}_\delta f_2(x) + U^*$ for any $\delta > 0$ and any weak* neighborhood U^* of 0 in X^*.*

Rule 2 Strong Fuzzy Sum Rule *Let $f_1 : X \to \mathbb{R}_\infty$ be finite at x and lower semicontinuous near x and $f_2 : X \to \mathbb{R}$ be Lipschitz continuous near x. Then $\partial(f_1 + f_2)(x) \subset \hat{\partial}_\delta f_1(x) + \hat{\partial}_\delta f_2(x) + \delta B^*$ for any $\delta > 0$.*

A Banach space is called a *trustworthy space* [15] (for some kind of a subdifferential) if Rule 1 is valid in it. The following theorem proved by Fabian [12] states that for the Fréchet subdifferential the class of trustworthy spaces coincides with Asplund spaces.

Theorem 3 *The following assertions are equivalent:*
1. *X is an Asplund space.*
2. *The Weak Fuzzy Sum Rule is valid in X.*
3. *The Strong Fuzzy Sum Rule is valid in X.*

The Weak Fuzzy Sum Rule yields the following representation of Fréchet normals to the intersection of closed sets.

Proposition 14 *Let Ω_1, Ω_2 be closed subsets in an Asplund space X and $x \in \Omega_1 \cap \Omega_2$. Then $N(x, \Omega_1 \cap \Omega_2) \subset \hat{N}_\delta(x, \Omega_1) + \hat{N}_\delta(x, \Omega_2) + U^*$ for any $\delta > 0$ and any weak* neighborhood U^* of 0 in X^*.*

Other fuzzy calculus results (chain rules, formulas for maximum-type functions, mean value theorems, etc.) for functions and multifunctions can be deduced from (some form of) the sum rule [5,16,24,37].

Extremal Principle

The *Extremal Principle* continues the line of variational principles discussed above and is in a sense equivalent to them as well as to the sum rules.

Let Ω_1, Ω_2 be closed subsets in X. They are called *locally extremal* [29,30] near $x \in \Omega_1 \cap \Omega_2$ if there exists a neighborhood U of x and sequences $\{a_{ik}\} \in X$, $i = 1, 2$, $k = 1, 2, \ldots$, such that $a_{ik} \to 0$ when $k \to \infty$ and $(\Omega_1 - a_{1k}) \cap (\Omega_2 - a_{2k}) \cap U = \emptyset$, $k = 1, 2, \ldots$

This means that by an arbitrarily small shift the sets can be made unintersecting in a neighborhood of x. The definition represents a rather general notion of extremality: some locally extremal system corresponds to a local solution of any optimization problem (see various examples in [22,29,31,33]).

The *Extremal Principle*, first established in [29] (see also [22,30,31]) for the case of a Fréchet smooth space (and in terms of ε-normals) and in [35] (see [34]) in the Asplund space setting, provides a dual space characterization of locally extremal systems in terms of Fréchet normals. It can be viewed as a fuzzy form of the *separation* property.

Extremal Principle *If a system of sets Ω_1, Ω_2 is locally extremal near $x \in \Omega_1 \cap \Omega_2$ then for any $\delta > 0$ there exist elements $x_1^* \in \hat{N}_\delta(x|\Omega_1)$, $x_2^* \in \hat{N}_\delta(x|\Omega_2)$ such that $\|x_1^* + x_2^*\| < \delta$, $\|x_1^*\| + \|x_2^*\| = 1$.*

The following theorem proved in [35] shows that the Extremal Principle provides an extremal characterization of Asplund spaces.

Theorem 4 *The following assertions are equivalent:*
1. *X is an Asplund space.*
2. *The Extremal Principle is valid in X.*

Due to Theorems 3 and 4 the Extremal Principle is equivalent to the Sum Rules. It is also equivalent to some other basic results of nonsmooth analysis [6,34,45].

The Extremal Principle can be viewed as a certain extension of the classical separation theorem for convex sets. It was used in [22,29,30,31,36] and in many other papers as a main tool for deducing calculus formulas and necessary optimality conditions.

The local extremality assumption in the Extremal Principle can be replaced by a weaker *stationarity* condition [26,27,28]. The resulting *Extended Extremal Principle* is formulated as a necessary and sufficient condition and is also equivalent to the asplundity of the space.

As noticed in [35], considering the extremal system provided by the pair $\{x\}, \Omega$, where x is a boundary point of a closed set Ω makes it possible to deduce from Theorem 4 the following nonconvex generalization of the well-known *Bishop–Phelps theorem* [40].

Corollary 1 *Let Ω be a closed subset in an Asplund space X and let $x \in bd\Omega$. Then for any $\delta > 0$ there exists $x^* \in \hat{N}_\delta(x|\Omega)$ such that $\|x^*\| = 1$.*

See also

► Nonsmooth Analysis: Weak Stationarity

References

1. Bazaraa MS, Goode JJ, Nashed ZZ (1974) On the cones of tangents with applications to mathematical programming. J Optim Theory Appl 13:389–426
2. Borwein JM (1978) Weak tangent cones and optimization in a Banach space. SIAM J Contr Optim 16:512–522
3. Borwein JM, Preiss D (1987) A smooth variational principle with applications to subdifferentiability and differentiability of convex functions. Trans Am Math Soc 303:517–527
4. Borwein JM, Strojwas HM (1985) Tangential approximations. Nonlinear Anal 9:1347–1366
5. Borwein JM, Zhu QJ (1999) A survey of subdifferential calculus with applications. Nonlinear Anal 38:687–773
6. Borwein JM, Zhu QJ (2005) Techniques of Variational Analysis. In: CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC, 20. Springer, New York
7. Clarke FH (1983) Optimization and Nonsmooth Analysis. Wiley, New York
8. Deville R, Godefroy G, Zizler V (1993) Smoothness and Renorming in Banach spaces. In: Pitman Monographs and Surveys in Pure and Applied Math., vol 64. Longman, Harlow UK
9. Dubovitskii AY, Milyutin AA (1965) Extremum problems in the presence of restrictions. USSR Comp Math Math Phys 5:1–80
10. Ekeland I (1974) On the variational principle. J Math Anal Appl 47:324–353
11. Fabian M (1986) Subdifferentials, local ε-supports and Asplund spaces. J London Math Soc 34:568–576
12. Fabian M (1989) Subdifferentiability and trustworthiness in the light of a new variational principle of Borwein and Preiss. Acta Univ Carolinae 30:51–56
13. Fabian M, Mordukhovich BS (1998) Nonsmooth characterizations of Asplund spaces and smooth variational principles. Set-Valued Anal 6:381–406
14. Gould FJ, Tolle JW (1975) Optimality conditions and constraint qualifications in Banach space. J Optim Theory Appl 15:667–684
15. Ioffe AD (1983) On subdifferentiability spaces. Ann NY Acad Sci 410:107–119
16. Ioffe AD (1984) Calculus of Dini subdifferentials of functions and contingent coderivatives of set-valued maps. Nonlinear Anal 8:517–539
17. Ioffe AD (1990) Proximal analysis and approximate subdifferentials. J London Math Soc 41:175–192
18. Kruger AY (1977) Subdifferentials of nonconvex functions and generalized directional derivatives. Deposited in VINITI no. 2661–77. Minsk (in Russian)
19. Kruger AY (1981) Generalized differentials of nonsmooth functions. Deposited in VINITI no. 1332–81. Minsk (in Russian)
20. Kruger AY (1981) Necessary conditions of an extremum in problems of nonsmooth optimization. Deposited in VINITI no. 1333–81. Minsk (in Russian)
21. Kruger AY (1981) ε-semidifferentials and ε-normal elements. Deposited in VINITI no. 1331–81. Minsk (in Russian)
22. Kruger AY (1985) Generalized differentials of nonsmooth functions and necessary conditions for an extremum. Siberian Math J 26:370–379
23. Kruger AY (1985) Properties of generalized differentials. Siberian Math J 26:822–832
24. Kruger AY (1996) On calculus of strict ε-semidifferentials. Dokl Akad Nauk Belarusi 40(4):34–39 (in Russian)
25. Kruger AY (2003) On Fréchet subdifferentials. J Math Sci (NY) 116(3):3325–3358 Optimization and related topics 3
26. Kruger AY (2004) Weak stationarity: eliminating the gap between necessary and sufficient conditions. Optimization 53(2):147–164
27. Kruger AY (2005) Stationarity and regularity of set systems. Pac J Optim 1(1):101–126
28. Kruger AY (2006) About regularity of collections of sets. Set-Valued Anal 14(2):187–206
29. Kruger AY, Mordukhovich BS (1980) Extremal points and the Euler equation in nonsmooth optimization. Dokl Akad Nauk BSSR 24(8):684–687 (in Russian)
30. Kruger AY, Mordukhovich BS (1980) Generalized normals and derivatives and necessary conditions for an extremum in problems of nondifferentiable programming. Deposited in VINITI, I – no. 408–80, II – no. 494–80. Minsk (in Russian)
31. Mordukhovich BS (1988) Approximation Methods in Problems of Optimization and Control. Nauka, Moscow (in Russian)
32. Mordukhovich BS (1994) Generalized differential calculus for nonsmooth and set-valued mappings. J Math Anal Appl 183(1):250–288
33. Mordukhovich BS (2001) The extremal principle and its applications to optimization and economics. In: Rubinov A,

Glover B (eds) Optimization and Related Topics, Applied Optimization, vol 47. Kluwer, Dordrecht, pp 343–369
34. Mordukhovich BS (2006) Variational Analysis and Generalized Differentiation. I Basic theory, vol 330 of Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]. Springer, Berlin
35. Mordukhovich BS, Shao Y (1996) Extremal characterizations of Asplund spaces. Proc Am Math Soc 124:197–205
36. Mordukhovich BS, Shao Y (1996) Nonsmooth sequential analysis in Asplund spaces. Trans Am Math Soc 348:1235–1280
37. Mordukhovich BS, Shao Y (1997) Fuzzy calculus for coderivatives of multifunctions. Nonlinear Anal 29:605–626
38. Mordukhovich BS, Wang B (2002) Necessary suboptimality and optimality conditions via variational principles. SIAM J Control Optim 41(2):623–640
39. Penot J-P (1978) Calcul sous-differentiel et optimisation. J Funct Anal 27:248–276
40. Phelps RR (1993) Convex Functions, Monotone Operators and Differentiability, 2nd edn. Lecture Notes in Mathematics, vol 1364. Springer, New York
41. Rockafellar RT (1970) Convex Analysis. Princeton Univ Press, Princeton
42. Rockafellar RT, Wets RJ-B (1998) Variational Analysis. Springer, New York
43. Sachs E (1978) Differentiability in optimization theory. Math Operationsforsch Statist, Ser Optim 9:497–513
44. Treiman JS (1986) Clarke's gradients and epsilon-subgradients in Banach spaces. Trans Am Math Soc 294(1):65–78
45. Zhu QJ (1998) The equivalence of several basic theorems for subdifferentials. Set-Valued Anal 81:171–185

Nonsmooth Analysis: Weak Stationarity

ALEXANDER Y. KRUGER
Centre for Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Ballarat, Australia

MSC2000: 90C46, 90C48, 58C20, 58E30

Article Outline

Keywords

Variational analysis; Subdifferential; Normal cone; Optimality; Stationarity; Regularity; Slope; Multifunction; Asplund space

Introduction

The article considers different stationarity and regularity concepts for extended real-valued functions on metric spaces.

All the properties can be characterized in terms of certain local constants. A function is said to be stationary at a point (in some sense) if the corresponding constant is zero (a critical point). Otherwise the function is said to be regular at this point (in the same sense), and the constant provides a quantitative estimate of regularity.

Traditionally the stationary behavior of a function at a point (*stationary point*) means that it is arbitrarily close to a constant near this point, that is the rate of change of the function is infinitely small compared to the increment of the variable. Of course, this is equivalent to the derivative being equal to zero (dual characterization of stationarity). The classical examples of the stationary behavior of a function, found in the textbooks, are given by the functions $y = x^2$, $y = -x^2$, and $y = x^3$. These three examples characterize the three possible types of stationary behavior in the differentiable case.

Stationarity arises naturally in optimization theory: a point of minimum or maximum is necessarily a stationary point. At the same time it is easy to see that weaker stationarity concepts are applicable to optimization problems. For instance, when dealing with minimizing a real-valued function, only its decrement must be infinitely small at a stationary point, and the function itself does not need to be arbitrarily close to a constant or even differentiable at the point. Of course, if

it is differentiable, one has the same classical stationarity concept. In the nondifferentiable case more types of the stationary behavior are possible. The examples are: $y = |x|$ and $y = \max(x, -x^2)$ (both functions are considered near the point $x = 0$). One can speak about *inf-stationarity* (the term suggested by Vladimir F. Demianov). From the point of view of maximization a concept of *sup-stationarity* can be considered in a similar way. Thus in the nondifferentiable case the stationarity splits into two "semi-stationarity" concepts.

Another pair of stationarity concepts can be of interest in optimization theory: the point itself may not be stationary (inf or sup), but in any of its neighborhood there exists another point in which the behavior of the function is arbitrarily close to stationary (a "fuzzy" condition). We will speak about *weak stationarity* (inf or sup). The exact definitions will be given below.

An example of this type of stationary behavior is given by the function $y = x\sin(1/x)$ if $x \neq 0$, and $y = 0$ if $x = 0$. One can easily see that this function is differentiable on $\mathbb{R}\backslash\{0\}$, and there exists a sequence $\{x_k\}$ such that $x_k \to 0$ and x_k is a point of local minimum, $k = 1, 2, \ldots$ Another example: $y = x + x^2\sin(1/x)$ if $x \neq 0$, and $y = 0$ if $x = 0$. This function is everywhere differentiable, $y'(0) = 1$, and there exists a sequence $\{x_k\}$ such that $x_k \to 0$ and $y'(x_k) = 0$, $k = 1, 2, \ldots$

Stationarity concepts can also be defined in terms of dual space elements (subdifferentials). The relations between primal and dual definitions provide dual characterizations of (primal space concepts of) stationarity.

This article contains no proofs. A more detailed description of the stationarity and regularity concepts for real-valued functions can be found in [11].

Mostly standard notations are used throughout this article. X denotes a metric space with distance d. $B_\rho(x)$ stands for a closed ball with center x and radius ρ.

Definitions

Inf-θ-Stationarity and Inf-θ-Regularity

Let f be a function on a metric space X with values in the extended real line $\mathbb{R}_\infty = \mathbb{R} \cup \{+\infty\}$. It is assumed to be finite at some point $x^\circ \in X$.

For $\rho > 0$ define the constant

$$\theta_\rho[f](x^\circ) = \inf_{x \in B_\rho(x^\circ)} f(x) - f(x^\circ). \tag{1}$$

Note that $\theta_\rho[f](x^\circ) \le 0$ and the equality $\theta_\rho[f](x^\circ) = 0$ for some $\rho > 0$ (for all $\rho > 0$) means that x° is a point of local (global) minimum of f.

Of course, the infimum in (1) can be limited to the set $\{x \in B_\rho(x^\circ)\colon f(x) \le f(x^\circ)\}$, or even to the set $\{x \in B_\rho(x^\circ)\colon f(x) < f(x^\circ)\}$ under the additional agreement that the infimum over the empty subset of $\mathbb{R}_-$ is 0.

The function $\rho \to \theta_\rho[f](x^\circ)$ is nonincreasing on $\mathbb{R}_+$ and $\lim_{\rho\to+0} \theta_\rho[f](x^\circ) \le 0$. The equality $\lim_{\rho\to+0} \theta_\rho[f](x^\circ) = 0$ means that f is lower semicontinuous at x°. In the latter case it can be important to know how quickly $\theta_\rho[f](x^\circ)$ approaches 0 compared to ρ.

Define two more "derivativelike" constants based on (1):

$$\theta[f](x^\circ) = \limsup_{\rho\to+0} \frac{\theta_\rho[f](x^\circ)}{\rho}, \tag{2}$$

$$\hat\theta[f](x^\circ) = \limsup_{x \xrightarrow{f} x^\circ, \rho\to+0} \frac{\theta_\rho[f](x)}{\rho}, \tag{3}$$

where $x \xrightarrow{f} x^\circ$ means that $x \to x^\circ$ with $f(x) \to f(x^\circ)$. Due to the variations of x in (3), $\hat\theta[f](x^\circ)$ gains some properties of the strict derivative.

The constants (2) and (3) are nonpositive too, and "zero cases" correspond to certain kinds of stationary behavior of f near x°. If a constant is strictly negative, this can be considered as a kind of regularity.

Definition 1 f is

(i) *inf-θ-stationary* at x° if $\theta[f](x^\circ) = 0$;
(ii) *weakly inf-θ-stationary* at x° if $\hat\theta[f](x^\circ) = 0$;
(iii) *inf-θ-regular* at x° if $\theta[f](x^\circ) < 0$;
(iv) *strongly inf-θ-regular* at x° if $\hat\theta[f](x^\circ) < 0$.

The purpose of the "inf" prefix in this definition is to emphasize that minimization problems are addressed here. Unlike the classical case, stationarity-regularity properties of nondifferentiable functions "from below" and "from above" can be essentially different.

Inf-τ-Stationarity and Inf-τ-Regularity

Another way of defining stationarity-regularity is based on using slightly modified versions of (2) and (3):

$$\tau[f](x^\circ) = \liminf_{x\to x^\circ} \frac{[f(x) - f(x^\circ)]_-}{d(x, x^\circ)}, \tag{4}$$

$$\hat{\tau}[f](x^\circ) = \limsup_{\substack{f \\ x \to x^\circ,\ \rho \to +0}} \inf_{u \in B_\rho(x) \setminus \{x\}} \frac{[f(u) - f(x)]_-}{d(u,x)} . \tag{5}$$

The notation $[\alpha]_- = \min(\alpha, 0)$ is used here. Again, only the points $x \in B_\rho(x^\circ)$ with $f(x) < f(x^\circ)$ and $u \in B_\rho(x)$ with $f(u) < f(x)$ are of interest in (4) and (5), respectively. The role of the notation is to handle the case where the set of such points is empty. Similarly to (2) and (3), these constants are nonpositive.

Remark 1 $\tau[f](x^\circ)$ coincides up to a sign with the *strong slope* $|\nabla f|(x^\circ)$ of f at x° [1] (see also [5]).

Definition 2 f is
(i) *inf-τ-stationary* at x° if $\tau[f](x^\circ) = 0$;
(ii) *weakly inf-τ-stationary* at x° if $\hat{\tau}[f](x^\circ) = 0$;
(iii) *inf-τ-regular* at x° if $\tau[f](x^\circ) < 0$;
(iv) *strongly inf-τ-regular* at x° if $\hat{\tau}[f](x^\circ) < 0$.

The relations between the constants (2), (3) and (4), (5), as well as between the corresponding stationarity and regularity concepts will be discussed in the next section.

Sup-Stationarity and Sup-Regularity

Similarly to (1)–(5) corresponding "maximization" constants can be defined. To do this one has to replace "inf," "lim inf," "lim sup," and $[\cdot]_-$ by "sup," "lim sup," "lim inf," and $[\cdot]_+$, respectively, in the corresponding definitions. The resulting constants are nonnegative. They are related to (1)–(5) by the following equalities:

$$\begin{aligned}
\theta_\rho^+[f](x^\circ) &= -\theta_\rho[-f](x^\circ) , \\
\theta^+[f](x^\circ) &= -\theta[-f](x^\circ) , \\
\hat{\theta}^+[f](x^\circ) &= -\hat{\theta}[-f](x^\circ) , \\
\tau^+[f](x^\circ) &= -\tau[-f](x^\circ) , \\
\hat{\tau}^+[f](x^\circ) &= -\hat{\tau}[-f](x^\circ)
\end{aligned}$$

and lead to similar *sup-stationarity* and *sup-regularity* concepts.

Of course, for a function f the set of sup-stationary (sup-regular) points is different in general from that of inf-stationary (inf-regular) points.

The "combined" concepts can also be of interest. It is natural to say that a function is stationary (in some sense) at a point if it is either inf-stationary or sup-stationary at this point. In constrast, the regularity property for a function is satisfied when this function is both inf-regular and sup-regular at the point.

Definition 3 f is
(i) *θ-stationary* at x° if $\max(\theta[f](x^\circ), \theta[-f](x^\circ)) = 0$;
(ii) *weakly θ-stationary* at x° if $\max(\hat{\theta}[f](x^\circ), \hat{\theta}[-f](x^\circ)) = 0$;
(iii) *θ-regular* at x° if $\max(\theta[f](x^\circ), \theta[-f](x^\circ)) < 0$;
(iv) *strongly θ-regular* at x° if $\max(\hat{\theta}[f](x^\circ), \hat{\theta}[-f](x^\circ)) < 0$;
(v) *τ-stationary* at x° if $\max(\tau[f](x^\circ), \tau[-f](x^\circ)) = 0$;
(vi) *weakly τ-stationary* at x° if $\max(\hat{\tau}[f](x^\circ), \hat{\tau}[-f](x^\circ)) = 0$;
(vii) *τ-regular* at x° if $\max(\tau[f](x^\circ), \tau[-f](x^\circ)) < 0$;
(viii) *strongly τ-regular* at x° if $\max(\hat{\tau}[f](x^\circ), \hat{\tau}[-f](x^\circ)) < 0$.

Strong inf-regularity can be interpreted in the following way: all points in a neighborhood of a given point have "descent sequences," and the rate of descent is uniform. In contrast to that, strong regularity is equivalent to the existence of both descent and ascent sequences with the uniformity property.

Dual Stationarity and Regularity

All definitions in the preceding subsections are primal space definitions. As in the classical analysis, dual characterizations of stationarity and regularity concepts are important. In the case of a normed linear space, such characterizations can be formulated in terms of *Fréchet subdifferentials*.

Let X be a normed linear space. Its (topological) dual is denoted X^*. $\langle \cdot, \cdot \rangle$ is the bilinear form defining the duality pairing. Recall that the Fréchet subdifferential of f at x° is defined as

$$\partial f(x^\circ) = \left\{ x^* \in X^* : \liminf_{x \to x^\circ} \frac{f(x) - f(x^\circ) - \langle x^*, x - x^\circ \rangle}{||x - x^\circ||} \geq 0 \right\} . \tag{6}$$

Definition 4 f is
(i) *inf-d-stationary* at x° if $0 \in \partial f(x^\circ)$;
(ii) *inf-d-regular* at x° if $0 \notin \partial f(x^\circ)$.

It follows immediately from the definitions that in the normed space setting inf-τ-stationarity (inf-τ-regularity) is equivalent to inf-d-stationarity (inf-d-regularity).

Somewhat more complicated constructions are needed for the characterization of weak stationarity and strong regularity.

Let us assume for simplicity that f is lower semicontinuous near x°.

In the general nonconvex setting the subdifferential mapping $\partial f(\cdot)$ fails to possess good (semi-)continuity properties. In fact, the set $\partial f(x)$ can be empty rather often. Based on (6) one can define a more robust derivativelike object:

$$\hat{\partial}_\delta f(x^\circ) = \bigcup_{\substack{x\in B_\delta(x^\circ)\\ |f(x)-f(x^\circ)|\le\delta}} \partial f(x)\,. \tag{7}$$

This object depends on a positive parameter δ and accumulates information on "differential" properties of f at nearby points, thus attaining some properties of the strict derivative. The set (7) is called the *strictδ-subdifferential* of f at x° (see [7,8,9]). In contrast to (6), set (7) can be nonconvex. However, it possesses certain subdifferential calculus.

Using (7) one more constant can be defined for characterizing stationarity/regularity properties of f:

$$\eta[f](x^\circ) = \liminf_{\delta\to 0}\{||x^*|| : x^* \in \hat{\partial}_\delta f(x^\circ)\}\,. \tag{8}$$

Unlike the constants considered in the preceding subsections, this constant is nonnegative.

Definition 5 f is
(i) *inf-η-stationary* at x° if $\eta[f](x^\circ) = 0$;
(ii) *inf-η-regular* at x° if $\eta[f](x^\circ) > 0$.

Note that the inf-η-stationary condition $\eta[f](x^\circ) = 0$ does not imply the inclusion $0 \in \hat{\partial}_\delta f(x^\circ)$.

Example 1 Take $f(x) = x$, if $x < 0$, and $f(x) = x^2$ otherwise. One has $\partial f(0) = \emptyset$, $0 \notin \hat{\partial}_\delta f(0)$ for any $\delta > 0$, while $\eta[f](0) = 0$.

Fortunately (8) happens to be closely related to (3) and (5).

Sup-d-stationarity and sup-η-stationarity as well as the corresponding regularity concepts can be defined in a similar way.

Formulation

Relations Between the "Elementary" Constants

Proposition 1 *The following assertions hold true:*
(i) $\tau[f](x^\circ) \le \theta[f](x^\circ)$;
(ii) *If* $\theta_\rho[f](x^\circ) = 0$ *for some* $\rho > 0$, *then* $\tau[f](x^\circ) = \theta[f](x^\circ) = 0$.

Proposition 1 (i) implies the relations between the corresponding stationarity and regularity concepts:
- inf-τ-stationarity $\Rightarrow$ inf-θ-stationarity;
- inf-θ-regularity $\Rightarrow$ inf-τ-regularity.

Proposition 1 (ii) means that at a point of local minimum a function is both inf-τ-stationary and inf-θ-stationary.

Inequality (i) in Proposition 1 can be strict even for functions from $\mathbb{R}$ to $\mathbb{R}$.

Example 2 Take $f(x) = -|x|$, if $|x| = 1/2^n$, $n = 1, 2, \ldots$, and $f(x) = 0$ otherwise. Obviously $\tau[f](0) = -1$. At the same time, for any $\rho \in \Xi_n = \{\rho\colon 1/2^n \le \rho < 1/2^{n-1}\}$ one has $\theta_\rho[f](0) = -1/2^n$ and

$$\sup_{\rho\in\Xi_n} \frac{\theta_\rho[f](0)}{\rho} = \frac{-1/2^n}{1/2^{n-1}} = -\frac{1}{2}\,.$$

Thus, $\theta[f](0) = -1/2$.

It is possible to modify the above example to make $\theta[f](0)$ equal zero.

Example 3 Take $f(x) = -|x|$, if $|x| = 1/n^n$, $n = 1, 2, \ldots$, and $f(x) = 0$ otherwise. One still has $\tau[f](0) = -1$ while $\theta[f](0) = 0$.

Thus, in the above example f is inf-τ-regular at 0 while being inf-θ-stationary at this point.

It is possible to modify the example further to make f continuous and even differentiable near 0 (but not strictly differentiable!) while keeping the inequality (i) in Proposition 1 strict.

Relations Between the "Strict" Constants

The relations between the elementary constants and their "strict" counterparts, as well as between the two "strict" constants, are given by the following theorem.

Theorem 1 *The following assertions hold true:*
(i) $\hat{\theta}[f](x^\circ) \ge \limsup_{x\xrightarrow{f}x^\circ} \theta[f](x)$,

(ii) $\hat{\tau}[f](x^\circ) = \limsup_{x \xrightarrow{f} x^\circ} \tau[f](x)$;

(iii) $\hat{\tau}[f](x^\circ) \le \hat{\theta}[f](x^\circ)$;

(iv) *If X is complete and f is lower semicontinuous near x°, then* $\hat{\tau}[f](x^\circ) = \hat{\theta}[f](x^\circ)$.

Parts (i) and (ii) of Theorem 1 imply the inequalities

$$\theta[f](x^\circ) \le \hat{\theta}[f](x^\circ)\,, \quad \tau[f](x^\circ) \le \hat{\tau}[f](x^\circ)\,,$$

and both of them can be strict.

Example 4 Take the function f from Example 1. Evidently, f attains a local minimum at $x_n = 1/2^n$ for any $n = 1, 2, \ldots$, and consequently, $\theta_\rho[f](x_n) = 0$ for some $\rho > 0$. It follows from Proposition 1 (ii) that $\tau[f](x_n) = \theta[f](x_n) = 0$. Consequently, $\hat{\tau}[f](0) = \hat{\theta}[f](0) = 0$. Recall that $\tau[f](0) = -1$ and $\theta[f](0) = -1/2$.

Inequalities (i) and (iii) in Theorem 1 can be strict, too.

Example 5 Define the function $f : \mathbb{R} \to \mathbb{R}$ in the following way: $f(x) = x$ if $x \le 0$, $f(x) = x - 1/n$ if $1/n < x \le 1/(n-1)$, $n = 2, 3, \ldots$, $f(x) = x - 1/2$ if $x > 1/2$. It is easy to see that $\theta[f](x) = \tau[f](x) = -1$ for any $x \in \mathbb{R}$. Then $\hat{\tau}[f](0) = -1$. On the other hand, take $x_n = 1/n + 1/n^2$, $\rho_n = 1/n$, $n = 1, 2, \ldots$ Then $f(x_n) = 1/n^2$, and consequently, $\theta_{\rho_n}[f](x_n) \ge -1/n^2$. It follows immediately that $\hat{\theta}[f](0) = 0$.

Due to part (iv) of Theorem 1, in the case of a lower semicontinuous function on a complete metric space two weak stationarity concepts as well as two strong regularity concepts coincide and the prefixes θ and τ can be omitted.

Corollary 1 *The following assertions hold true:*

(i) *Inf-θ-stationarity $\Rightarrow$ weak inf-θ-stationarity;*
strong inf-θ-regularity $\Rightarrow$ inf-θ-regularity;

(ii) *Inf-τ-stationarity $\Rightarrow$ weak inf-τ-stationarity;*
strong inf-τ-regularity $\Rightarrow$ inf-τ-regularity;

(iii) *Weak inf-τ-stationarity $\Rightarrow$ weak inf-θ-stationarity;*
strong inf-θ-regularity $\Rightarrow$ strong inf-τ-regularity;

(iv) *If X is complete and f is lower semicontinuous near x°, then*
weak inf-τ-stationarity $\Leftrightarrow$ weak inf-θ-stationarity;
strong inf-θ-regularity $\Leftrightarrow$ strong inf-τ-regularity.

The next "fuzzy" characterization of weak inf-τ-stationarity can be convenient for applications. It follows directly from definition (5).

Proposition 2 *f is weakly inf-τ-stationary at x° if and only if for any $\varepsilon > 0$ there exists an $x \in B_\varepsilon(x^\circ)$ such that $|f(x) - f(x^\circ)| \le \varepsilon$ and*

$$f(u) + \varepsilon d(u, x) \ge f(x) \quad \text{for all } u \text{ near } x\,. \tag{9}$$

Remark 2 A point x satisfying (9) is referred to in [12] (see also [6]) as a *local Ekeland point* of f (with factor ε). If all the conditions in Proposition 2 are satisfied, then x° is said to be a *stationary point* of f *with respect to minimization* [12]. Thus, stationarity with respect to minimization is equivalent to weak inf-τ-stationarity and, in the case of a lower semicontinuous function on a complete metric space, also to weak inf-θ-stationarity.

Relations Between the Primal and Dual Constants

Henceforth X is assumed to be a normed linear space. The next assertion is straightforward and has already been mentioned in the previous section.

Proposition 3

(i) *inf-τ-stationarity $\Leftrightarrow$ weak inf-d-stationarity;*

(ii) *inf-τ-regularity $\Leftrightarrow$ inf-d-regularity.*

Remark 3 Due to Propositions 1 and 3 the inclusion $0 \in \partial f(x^\circ)$ is sufficient for inf-θ-stationarity of f at x°. The opposite implication is not true in general (see Examples 2 and 3).

In what follows f is assumed to be lower semicontinuous near x°.

Theorem 2

(i) $\hat{\theta}[f](x^\circ) + \eta[f](x^\circ) \ge 0$.

(ii) *If X is Asplund, then*

$$\frac{\hat{\theta}[f](x^\circ)}{[1 + \hat{\theta}[f](x^\circ)]_+} + \eta[f](x^\circ) \le 0\,.$$

This theorem follows from [10], Theorem 2. The first part of the theorem is elementary. The proof of the second part is based on the application of the two fundamental results of variational analysis: the *Ekeland variational principle* [2] and the *fuzzy sum rule* due to Fabian [3].

Thus, in an Asplund space the constants $\hat{\theta}[f](x^\circ)$ and $\eta[f](x^\circ)$ can be zero or nonzero only simultaneously. Recall that a Banach space is called *Asplund* (see [4,13,14]) if any continuous convex function on it

is Fréchet differentiable on a dense G_δ subset. Note that in a Banach space $\hat{\theta}[f](x^\circ) = \hat{\tau}[f](x^\circ)$ due to Theorem 1.

Corollary 2

(i) *inf-η-stationarity $\Rightarrow$ weak inf-θ-stationarity;*
(ii) *strong inf-θ-regularity $\Rightarrow$ inf-η-regularity;*
(iii) *If X is Asplund, then*
weak inf-θ-stationarity $\Leftrightarrow$ weak inf-τ-stationarity $\Leftrightarrow$ inf-η-stationarity;
strong inf-θ-regularity $\Leftrightarrow$ strong inf-τ-regularity $\Leftrightarrow$ inf-η-regularity.

Differentiable Functions

The constants and corresponding stationarity/regularity concepts defined above take quite a traditional form when the function is assumed differentiable or convex. Fortunately, the number of different constants and concepts reduces significantly.

Theorem 3 *If f is Fréchet differentiable at x° with the derivative $\nabla f(x^\circ)$, then*

$$\theta[f](x^\circ) = \tau[f](x^\circ) = -\theta^+[f](x^\circ) = -\tau^+[f](x^\circ) = -||\nabla f(x^\circ)|| .$$

If, additionally, the derivative is strict, then

$$\hat{\theta}[f](x^\circ) = \hat{\tau}[f](x^\circ) = -\hat{\theta}^+[f](x^\circ) = -\hat{\tau}^+[f](x^\circ) = -||\nabla f(x^\circ)|| .$$

Recall that f is called *strictly differentiable* [13,15] at x° (with the derivative $\nabla f(x^\circ)$) if

$$\lim_{x\to x^\circ,\ u\to x^\circ} \frac{f(u) - f(x) - \langle \nabla f(x^\circ), u - x\rangle}{||u - x||} = 0.$$

This condition is stronger than the traditional Fréchet differentiability. Thus, condition $\nabla f(x^\circ) \neq 0$ does not guarantee strong regularity in the sense of Definition 1 (or Definition 2) unless f is strictly differentiable at x°.

Example 6 Take $f(x) = x + x^2 \sin(1/x)$, if $x \neq 0$ and $f(0) = 0$. This function is everywhere Fréchet differentiable and $\nabla f(0) = 1$. Thus, f is regular at zero. At the same time $\hat{\tau}[f](0) = \hat{\tau}^+[f](0) = 0$: there exists a sequence $x_k \to 0$ such that $\nabla f(x_k) \to 0$, and the assertion follows from Theorem 1, part (ii). Consequently, f is both weakly inf-stationary and weakly sup-stationary at zero.

Corollary 3 *If f is Fréchet differentiable at x° with the derivative $\nabla f(x^\circ)$, then the following conditions are equivalent:*
(i) *f is inf-θ-stationary at x°;*
(ii) *f is inf-τ-stationary at x°;*
(iii) *f is θ-stationary at x°;*
(iv) *f is τ-stationary at x°;*
(v) $\nabla f(x^\circ) = 0$.
If, additionally, the derivative is strict, then the above conditions are also equivalent to the following ones:
(vi) *f is weakly inf-θ-stationary at x°;*
(vii) *f is weakly inf-τ-stationary at x°;*
(viii) *f is weakly θ-stationary at x°;*
(xi) *f is weakly τ-stationary at x°.*

Remark 4 Stationarity and weak stationarity in the above corollary can be replaced with regularity and strong regularity, respectively, if one replaces the equality in (v) with the inequality $\nabla f(x^\circ) \neq 0$.

Convex Functions

In the convex case, as one might expect, all versions of inf-stationarity coincide and appear to be equivalent to just (local and global) minimality.

Theorem 4 *Let f be convex.*
(i) *If $\theta_\rho[f](x^\circ) < 0$ for some $\rho > 0$, then $\theta_\rho[f](x^\circ) < 0$ for all $\rho > 0$.*
(ii) *The functions $\rho \to \theta_\rho[f](x^\circ)/\rho$ and $\rho \to \theta^+_\rho[f](x^\circ)/\rho$ are nondecreasing on $\mathbb{R}_+\backslash\{0\}$.*
(iii) *The following equalities hold true:*

$$\hat{\theta}[f](x^\circ) = \hat{\tau}[f](x^\circ) = \theta[f](x^\circ) = \tau[f](x^\circ) = \inf_{\rho>0} \frac{\theta_\rho[f](x^\circ)}{\rho} = \inf_{x\neq x^\circ} \frac{[f(x) - f(x^\circ)]_-}{||x - x^\circ||},$$

$$\theta^+[f](x^\circ) = \tau^+[f](x^\circ) = \inf_{\rho>0} \frac{\theta^+_\rho[f](x^\circ)}{\rho} = \inf_{\rho>0} \sup_{||x-x^\circ||=\rho} \frac{[f(x) - f(x^\circ)]_+}{\rho}.$$

(iv) $\tau[f](x^\circ) + \tau^+[f](x^\circ) \geq 0$.
(v) $\hat{\tau}[f](x^\circ) + \hat{\tau}^+[f](x^\circ) \geq 0$.

(vi) *If* $\tau[f](x^\circ) + \tau^+[f](x^\circ) = 0$ *and* $\{x_k\} \subset X$ *is a sequence defining* $\tau[f](x^\circ)$*, that is* $x_k \to 0$ *and*

$$\tau[f](x^\circ) = \lim_{k\to\infty} \frac{f(x^\circ + x_k) - f(x^\circ)}{||x_k||}$$

then $\{-x_k\}$ *is a sequence defining* $\tau^+[f](x^\circ)$*:*

$$\tau^+[f](x^\circ) = \lim_{k\to\infty} \frac{f(x^\circ - x_k) - f(x^\circ)}{||x_k||}.$$

Corollary 4 *If* f *is convex, then the following conditions are equivalent:*

(i) f *attains a global minimum at* x°*;*
(ii) f *attains a local minimum at* x°*;*
(iii) f *is inf-*θ*-stationary at* x°*;*
(iv) f *is inf-*τ*-stationary at* x°*;*
(v) f *is* θ*-stationary at* x°*;*
(vi) f *is* τ*-stationary at* x°*;*
(vii) f *is weakly inf-*θ*-stationary at* x°*;*
(viii) f *is weakly inf-*τ*-stationary at* x°*;*
(ix) f *is weakly stationary at* x°*.*

Remark 5 The conditions $\tau[f](x^\circ) = \tau^+[f](x^\circ) = 0$ imply Fréchet differentiability of f at x° (with the derivative equal to zero). The weaker condition $\tau[f](x^\circ) + \tau^+[f](x^\circ) = 0$ in Theorem 4, (vi) implies linearity of the directional derivative of f along the direction of steepest descent (if the latter exists) with the opposite direction being automatically the direction of steepest ascent. This condition is not sufficient for differentiability of f at x° unless $X = \mathbb{R}$. Note also that the direction opposite to the direction of steepest ascent does not need to be a direction of steepest descent.

Example 7 ([1]) Take the function $f(x, y) = \max(x, y)$ on $\mathbb{R}^2$ and assume that $\mathbb{R}^2$ is equipped with the max type norm: $||x, y|| = \max(|x|, |y|)$. f is obviously not differentiable at 0. At the same time $\tau[f](0) = -1$, $\tau^+[f](0) = 1$. The vector $(-1, -1)$ defines the (unique) direction of steepest descent. The opposite vector $(1, 1)$ defines the direction of steepest ascent and f is linear along the line defined by these vectors. Note that the direction of steepest ascent is not unique. For instance, the vector $(1, 0)$ also defines the direction of steepest ascent, while the opposite vector does not define the direction of steepest descent and f is not linear along this line.

[1]The example was suggested by Alexander Rubinov (personal communication).

Remark 6 Stationarity and weak stationarity in assertions (iii)–(ix) of the above corollary can be replaced with regularity and strong regularity, respectively, if one replaces (i) and (ii) with the opposite assertions: x° is not a point of (local or global) minimum of f.

See also

► Nonsmooth Analysis: Fréchet Subdifferentials

References

1. De Giorgi E, Marino A, Tosques M (1980) Problemi di evoluzione in spazi metrici e curve di massima pendenza. Atti Accad. Nat. Lincei. Cl Sci Fiz Mat Natur 68:180–187
2. Ekeland I (1974) On the variational principle. J Math Anal Appl 47:324–353
3. Fabian M (1989) Subdifferentiability and trustworthiness in the light of a new variational principle of Borwein and Preiss. Acta Univ Carolinae 30:51–56
4. Fabian M (1997) Gâteaux Differentiability of Convex Functions and Topology. Weak Asplund Spaces. Canadian Mathematical Society Series of Monographs and Advanced Texts. Wiley, New York
5. Ioffe A D (2000) Metric regularity and subdifferential calculus. Russian Math Surveys 55:501–558
6. Klatte D, Kummer B (2002) Nonsmooth Equations in Optimization: Regularity, Calculus, Methods and Applications, vol 60 of Nonconvex Optimization and Its Applications. Kluwer, Dordrecht
7. Kruger AY (1996) On calculus of strict ε-semidifferentials. Dokl Akad Nauk Belarusi 40(4):34–39 (in Russian)
8. Kruger AY (2002) Strict (ε, δ)-semidifferentials and extremality conditions. Optimization 51:539–554
9. Kruger AY (2003) On Fréchet subdifferentials. J Math Sci (NY) 116:3:3325–3358. Optimization and related topics, 3
10. Kruger AY (2004) Weak stationarity: eliminating the gap between necessary and sufficient conditions. Optimization 53(2):147–164
11. Kruger AY (2006) Stationarity and regularity of real-valued functions. Appl Comput Math 5(1):79–93
12. Kummer B (2000) Inverse functions of pseudo regular mappings and regularity conditions. Math Program Ser B 88:313–339
13. Mordukhovich BS (2006) Variational Analysis and Generalized Differentiation. I Basic theory, vol 330 of Grundlehren der Mathematischen Wissenschaften (Fundamental Principles of Mathematical Sciences). Springer, Berlin
14. Phelps RR (1993) Convex Functions, Monotone Operators and Differentiability, 2nd edn. Lecture Notes in Mathematics, vol 1364. Springer, Berlin
15. Rockafellar RT, Wets RJ-B (1998) Variational Analysis. Springer, Berlin

Nonsmooth Optimization Approach to Clustering

ADIL BAGIROV
Centre for Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Victoria, Australia

MSC2000: 90C26, 90C56, 90C90

Article Outline

Introduction

Clustering is the *unsupervised* classification of the patterns. Cluster analysis deals with the problems of organization of a collection of patterns into clusters based on similarity. It has found many applications, including information retrieval, document extraction, image segmentation etc.

In cluster analysis we assume that we have been given a set A of a finite number of points of n-dimensional space $\mathbb{R}^n$, that is

$$A = \{a^1, \dots, a^m\}, \text{ where } a^i \in \mathbb{R}^n, \quad i = 1, \dots, m\,.$$

The subject of cluster analysis is the partition of the set A into a given number q of overlapping or disjoint subsets C_i, $i = 1, \dots, q$ with respect to predefined criteria such that

$$A = \bigcup_{i=1}^{q} C_i\,.$$

The sets $C_i, i = 1, \dots, q$ are called clusters. The clustering problem is said to be *hard clustering* if every data point belongs to one and only one cluster. Unlike hard clustering in the *fuzzy clustering* problem the clusters are allowed to overlap and instances have degrees of appearance in each cluster. In this paper we will exclusively consider the hard unconstrained clustering problem, that is we additionally assume that

$$C_i \bigcap C_k = \emptyset, \quad \forall i, k = 1, \dots, q,\ i \neq k\,.$$

and no constraints are imposed on the clusters C_i, $i = 1, \dots, q$. Thus every point $a \in A$ is contained in exactly one and only one set C_i.

Each cluster C_i can be identified by its center (or centroid). Then the clustering problem can be reduced to the following optimization problem (see [12,26]):

$$\text{minimize } \varphi(C, x) = \frac{1}{m} \sum_{i=1}^{q} \sum_{a \in C_i} \|x^i - a\|^2 \qquad (1)$$
$$\text{subject to } C \in \overline{C}, \quad x = (x^1, \dots, x^q) \in \mathbb{R}^{n \times q}$$

where $\|\cdot\|$ denotes the Euclidean norm, $C = \{C_1, \dots, C_q\}$ is a set of clusters, $\overline{C}$ is a set of all possible q-partitions of the set A, x^i is the center of the cluster C_i, $i = 1, \dots, q$:

$$x^i = \frac{1}{|C_i|} \sum_{a \in C_i} a\,,$$

and $|C_i|$ is a cardinality of the set C_i, $i = 1, \dots, q$. The problem (1) is also known as the minimum sum-of-squares clustering. The combinatorial formulation (1) of the minimum sum-of-squares clustering is not suitable for direct application of mathematical programming techniques. The problem (1) can be rewritten as the following mathematical programming problem:

$$\text{minimize } \psi(x, w) = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{q} w_{ij} \|x^j - a^i\|^2 \qquad (2)$$
$$\text{subject to } \sum_{j=1}^{q} w_{ij} = 1, \quad i = 1, \dots, m\,,$$

and

$$w_{ij} \in \{0, 1\}, \quad i = 1, \dots, m,\ j = 1, \dots, q\,.$$

Here

$$x^j = \frac{\sum_{i=1}^{m} w_{ij} a^i}{\sum_{i=1}^{m} w_{ij}}, \quad j = 1, \dots, q$$

and w_{ij} is the association weight of pattern a_i with cluster j (to be found), given by

$$w_{ij} = \begin{cases} 1 & \text{if pattern } a^i \text{ is allocated to cluster } j \\ 0 & \text{otherwise}\,. \end{cases}$$

w is an $m \times q$ matrix.

There exist different approaches to clustering including agglomerative and divisive hierarchical clustering algorithms as well as algorithms based on mathematical programming techniques. Descriptions of many of these algorithms can be found, for example, in [15,20,21,26].

Problem (2) is a global optimization problem. Therefore different algorithms of mathematical programming can be applied to solve this problem. Some review of these algorithms can be found in [16]. However, most of these algorithms are applicable for clustering on small data sets.

Different heuristics can be used for solving clustering problems on large data sets and k-means is one such algorithm. Different versions of this algorithm have been studied by many authors (see [26]). This is a fast algorithm. k-means gives good results when there are few clusters but deteriorates when there are many [16]. This algorithm achieves a local minimum of problem (1) (see [24]), however results of numerical experiments presented, for example, in [19] show that the best clustering found with k-means may be more than 50% worse than the best known one.

Much better results have been obtained with metaheuristics, such as simulated annealing, tabu search and genetic algorithms [23]. The simulated annealing approaches to clustering have been studied, for example, in [13,25,27]. Application of tabu search methods for solving clustering problem is studied in [1]. Genetic algorithms for clustering have been described in [23]. The results of numerical experiments, presented in paper [2] show that even for small problems of cluster analysis when the number of entities $m \leq 100$ and the number of clusters $q \leq 5$ these algorithms take 500–700 (sometimes several thousands) times more CPU time than the k-means algorithms. For relatively large data sets one can expect that this difference will increase. This makes metaheuristic algorithms of global optimization ineffective for solving many clustering problems.

The paper [18] develops variable neighborhood search algorithm and the paper [17] presents j-means algorithm which extends k-means by adding a jump move. The global k-means heuristic, which is an incremental approach to minimum sum-of-squares clustering problem, is developed in [22]. The incremental approach is also studied in the paper [19]. Results of numerical experiments presented show the high effectiveness of these algorithms for many clustering problems.

As mentioned above the problem (2) is the global optimization problem and the objective function in this problem is multimodal. However, global optimization techniques are highly time-consuming for solving many clustering problems. It is very important, therefore, to develop clustering algorithms that compute near global minimizers of the objective function. We propose the clustering algorithms based on nonsmooth optimization approach. The algorithms provide the capability of calculating clusters step-by-step, gradually increasing the number of data clusters until termination conditions are met, that is it allows one to calculate as many cluster as a data set contains with respect to some tolerance.

Formulation

The problems (1) and (2) can be reformulated as the following mathematical programming problem [7,8,12]

$$\begin{aligned} &\text{minimize } f(x^1,\ldots,x^q) \\ &\quad \text{subject to } x = (x^1,\ldots,x^q) \in \mathbb{R}^{n\times q}, \end{aligned} \tag{3}$$

where

$$f(x^1,\ldots,x^q) = \frac{1}{m}\sum_{i=1}^{m} \min_{j=1,\ldots,q} \|x^j - a^i\|^2 . \tag{4}$$

It is shown in [12] that problems (2) and (3) are equivalent. However, there are some differences between these two formulations:

- The number of variables in problem (2) is $(m+n)\times q$ whereas in problem (3) this number is only $n \times q$ and the number of variables does not depend on the number of instances. It should be noted that in many real-world databases the number of instances m is substantially greater than the number of attributes n.
- In the hard clustering problem (2) the coefficients w_{ij} are integer, that is the problem (2) contains both integer and continuous variables. In the nonsmooth optimization formulation variables are only continuous.
- Nonsmooth optimization formulation of the clustering problem allows one to easily consider different similarity measures.

All these circumstances can be considered as advantages of the nonsmooth optimization formulation (3).

If $q > 1$, the objective function (4) in problem (3) is nonconvex and nonsmooth. If the number q of clusters and the number n of attributes are large, we have a large-scale global optimization problem. Moreover, the form of the objective function in this problem is complex enough not to be amenable to the direct application of general purpose global optimization methods. Therefore, in order to ensure the practicality of the nonsmooth optimization approach to clustering, proper identification and use of local optimization methods is very important. Clearly, such an approach does not guarantee a globally optimal solution to problem (3). On the other hand, this approach provides a "near" global minimum of the objective function that, in turn, provides a good enough clustering description of the data set under consideration.

Note also that a meaningful choice of the number of clusters is very important for clustering analysis. It is difficult to define *a priori* how many clusters represent the set A under consideration. In order to avoid this difficulty, a step-by-step calculation of clusters is implemented in algorithms discussed in the next section.

Methods

In this section we will describe two incremental clustering algorithms. Both algorithms are based on nonsmooth optimization approach to clustering. In the first algorithm nonsmooth optimization approach is used to find starting points for k-means algorithm. This algorithm is a modification of the global k-means algorithm proposed in [22]. The second algorithm is an optimization based clustering algorithm.

Modified Global *k*-Means Algorithm

k-means algorithm and its different variations are known to be fast algorithms for clustering and they are applicable to large data sets. In this subsection we propose a new version of k-means algorithm: the modified global k-means algorithm, which in its turn is the modification of the global k-means algorithm. First we briefly describe k-means and the global k-means algorithms.

k-means algorithm proceeds as follows

Step 1. Choose a seed solution consisting of k centroids (not necessarily belonging to A).

Step 2. Allocate data points $a^i \in A$ to its closest centroid and obtain k-partition of A.

Step 3. Recompute centroids for this new partition and go to to Step 2 until no more data points change their clusters.

Nonsmooth Optimization Approach to Clustering, Algorithm 1
***k*-means algorithm**

Step 1. Compute the centroid x^1 of the set A:

$$x^1 = \frac{1}{m}\sum_{i=1}^{m} a^i$$

and set $k = 1$.

Step 2. Set $k = k + 1$ and consider the centers $x^1, x^2, \ldots, x^{k-1}$ from the previous iteration.

Step 3. Consider each point a of A as a starting point for the k-th cluster center, thus obtaining m initial solutions with k points $(x^1, x^2, \ldots, x^{k-1}, a)$; apply k-means algorithm starting from each of them; keep the best k-partition obtained and its center $(x^1, x^2, \ldots, x^{k-1}, x^k)$.

Step 4. If $k = q$ stop, otherwise go to Step 2.

Nonsmooth Optimization Approach to Clustering, Algorithm 2
The global *k*-means algorithm

The effectiveness of this algorithm highly depends on a starting point. It converges only to a local solution which can significantly differ from the global one in large data sets.

The global k-means algorithm proposed in [22] is the modification of k-means algorithm and it computes clusters successively that is in order to compute k-th cluster centroid this algorithm uses centroids of $k - 1$ clusters from the previous iteration. To compute $q \leq m$ clusters this algorithm proceeds as follows.

This version of the algorithm is not applicable for clustering on middle sized and large data sets. Two

procedures were introduced to reduce its complexity (see [22]). We mention here only one of them. Let

$$d_{k-1}^i = \min\left\{\|x^1 - a^i\|^2, \ldots, \|x^{k-1} - a^i\|^2\right\}. \quad (5)$$

For each $a^i \in A$ we compute:

$$r_i = \sum_{j=1}^{m} \min\{0, \|a^i - a^j\|^2 - d_{k-1}^j\}$$

and we take the data point $a^l \in A$ for which

$$l = \operatorname{argmin}_{i=1,\ldots,m} r_i.$$

Then k-means algorithm is applied starting from the point $(x^1, x^2, \ldots, x^{k-1}, a^l)$ to find k cluster centers.

Now we will describe a new version of the global k-means algorithm where a starting point for k-th cluster center is computed using nonsmooth optimization approach. Let us consider the problem of finding k cluster center assuming that the centers $x^1, \ldots, x^{k-1}$ for $(k-1)$-clustering problem are known. We introduce the following function:

$$\bar{f}^k(y) = \frac{1}{m}\sum_{i=1}^{m} \min\left\{d_{k-1}^i, \|y - a^i\|^2\right\} \quad (6)$$

where $y \in \mathbb{R}^n$ stands for k-th cluster center and d_{k-1}^i is defined as in (5). Consider a set

$$\overline{D} = \left\{y \in \mathbb{R}^n : \|y - a^i\|^2 \geq d_{k-1}^i\right\}.$$

This is a set where the distance from any point y to any data point is no less than the distance between this data point and its cluster center. We also consider the following set

$$D_0 = \mathbb{R}^n \setminus \overline{D} \equiv \Big\{y \in \mathbb{R}^n : \exists I \subset \{1, \ldots, m\}, I \neq \emptyset : \|y - a^i\| < d_{k-1}^i \forall i \in I\Big\}.$$

The function $\bar{f}^k$ is a constant on the set $\overline{D}$ and its value over this set is

$$\bar{f}^k(y) = d_0 \equiv \frac{1}{m}\sum_{i=1}^{m} d_{k-1}^i, \ \forall y \in \overline{D}.$$

It is clear that $x^j \in \overline{D}$ for all $j = 1, \ldots, k-1$ and $a^i \in D_0$ for all $a^i \in A$, $a^i \neq x^j$, $j = 1, \ldots, k-1$. It is also clear that $f(y) < d_0$ for all $y \in D_0$.

Step 1. For any $a^i \in D_0 \bigcap A$ calculate the set $S_2(a^i)$, the centroid c^i of this set and calculate the value $\bar{f}^k(c^i)$ of the function $\bar{f}^k$ at this point.

Step 2. Compute

$$\bar{f}_{min}^k = \min_{a^i \in D_0 \bigcap A} \bar{f}^k(c^i),$$

$$a^j = \operatorname{argmin}_{a^i \in D_0 \bigcap A} \bar{f}^k(c^i).$$

and the corresponding center c^j.

Step 3. Compute the set $S_2(c^j)$ and its centroid.

Step 4. Recompute the set $S_2(c^j)$ and its centroid until no more data points escape this set or return to this set.

Nonsmooth Optimization Approach to Clustering, Algorithm 3
An algorithm for finding the initial point

Any point $y \in D_0$ can be taken as a starting point for the k-th cluster center. Probably more preferably among them is a global minimizer of the function $\bar{f}^k$. This function is a nonconvex and nonsmooth and its minimization is difficult task. We consider a scheme for finding its local minimizer.

For any $y \in D_0$ we consider the following sets:

$$S_1(y) = \left\{a^i \in A : \|y - a^i\|^2 = d_{k-1}^i\right\},$$
$$S_2(y) = \left\{a^i \in A : \|y - a^i\|^2 < d_{k-1}^i\right\},$$
$$S_3(y) = \left\{a^i \in A : \|y - a^i\|^2 > d_{k-1}^i\right\}.$$

Since $y \in D_0$ the set $S_2(y) \neq \emptyset$. We suggest the following algorithm to find a starting point for the k-th cluster center.

Now we can describe the modified global k-means algorithm.

It is clear that $f^{k*} \geq 0$ for all $k \geq 1$ and the sequence $\{f^{k*}\}$ is decreasing, that is,

$$f^{k+1,*} \leq f^{k,*} \text{ for all } k \geq 1.$$

The latter implies that after $\bar{k} > 0$ iterations the stopping criterion in Step 4 will be satisfied.

Step 1. (Initialization). Select a tolerance $\varepsilon > 0$. Calculate the centroid $x^{1*} \in \mathbb{R}^n$ of the set A. Let f^{1*} be the corresponding value of the objective function (4). Set $k = 1$.

Step 2. (Computation of the next cluster center). Let $x^{1*}, \ldots, x^{k*}$ be the cluster centers for k clustering problem. Apply Algorithm 3 to find an initial point $y^{k+1,0} \in \mathbb{R}^n$ for the $k+1$-th cluster center.

Step 3. (Refinement of all cluster centers). Take $x^{k+1,0} = (x^{1*}, \ldots, x^{k*}, y^{k+1,0})$ as a new starting point, apply k-means algorithm to solve clustering problem for $q = k+1$. Let $x^{1*}, \ldots, x^{k+1,*}$ be a solution to this problem and $f^{k+1,*}$ be the corresponding value of the objective function (4).

Step 4. (Stopping criterion). If

$$\frac{f^{k*} - f^{k+1,*}}{f^{1*}} < \varepsilon$$

then stop, otherwise set $k = k+1$ and go to Step 2.

Nonsmooth Optimization Approach to Clustering, Algorithm 4
Modified global *k*-means algorithm

Nonsmooth Optimization Clustering Algorithm

In this subsection we propose an algorithm for clustering where nonsmooth optimization techniques are used to find a starting point for the k cluster center and to solve k-clustering problems.

It is clear that $f^{k*} \geq 0$ for all $k \geq 1$ and the sequence $\{f^{k*}\}$ is decreasing that is,

$$f^{k+1,*} \leq f^{k,*} \text{ for all } k \geq 1.$$

The latter implies that after $\bar{k} > 0$ iterations the stopping criterion in Step 4 will be satisfied.

Remark 1 One of the important questions when one tries to apply Algorithms 4 and 5 is the choice of the tolerance $\varepsilon > 0$. Large values of ε can result in the appearance of large clusters whereas small values can produce small and artificial clusters.

Remark 2 Algorithms 2, 4 and 5 are incremental clustering algorithms. Main difference between Algorithms 2 and 4 is in the way they compute starting points for the next cluster center. Algorithm 2 uses data points whereas Algorithm 4 uses local minimizers of the function $\bar{f}^k$. Algorithm 5 uses nonsmooth optimization techniques for the finding of both starting points and k-partition of a data set.

Step 1. (Initialization). Select a tolerance $\varepsilon > 0$. Calculate the centroid $x^{1*} \in \mathbb{R}^n$ of the set A. Let f^{1*} be the corresponding value of the objective function (4). Set $k = 1$.

Step 2. (Computation of the next cluster center). Select a point $y^0 \in \mathbb{R}^n$ and solve the following minimization problem:

$$\text{minimize } \bar{f}^k(y) \text{ subject to } y \in \mathbb{R}^n \tag{7}$$

where $\bar{f}^k$ is defined by (6).

Step 3. (Refinement of all cluster centers). Let $\bar{y}^{k+1,*}$ be a solution to problem (7). Take $x^{k+1,0} = (x^{1*}, \ldots, x^{k*}, \bar{y}^{k+1,*})$ as a new starting point and solve the following minimization problem:

$$\text{minimize } f^{k+1}(x) \text{ subject to}$$

$$x = (x^1, \ldots, x^{k+1}) \in \mathbb{R}^{n \times (k+1)} \tag{8}$$

where

$$f^{k+1}(a) = \frac{1}{m} \sum_{i=1}^{m} \min_{j=1,\ldots,k+1} \|x^j - a^i\|^2 .$$

Step 4. (Stopping criterion). Let $x^{k+1,*}$ be a solution to the problem (8) and $f^{k+1,*}$ be the corresponding value of the objective function. If

$$\frac{f^{k*} - f^{k+1,*}}{f^{1*}} < \varepsilon$$

then stop, otherwise set $k = k+1$ and go to Step 2.

Nonsmooth Optimization Approach to Clustering, Algorithm 5
Nonsmooth optimization clustering algorithm

Remark 3 Computational results on gene expression data sets presented in [6] demonstrate that Algorithm 4 is more efficient than Algorithm 2. However, the former requires more computational time.

Remark 4 Results of numerical experiments presented in [11] demonstrate that Algorithm 5 is efficient for solving large scale clustering problems in a reasonable CPU time. Moreover, its success to locate global solutions is higher than that for Algorithms 2 and 4. However, this algorithm requires significantly more CPU time than other algorithms.

Solving Optimization Problems

The objective functions in problems (7) and (8) are nonsmooth and nonconvex. If the number of attributes and clusters are large then the problem (8) is large scale problem. Both objective functions are non-regular and the computation of even one their subgradient may become very difficult problem (for the definition of non-regular function, see [14]). Therefore, subgradient-based methods are not always efficient for solving problems (7) and (8). We use the discrete gradient method to solve these problems [3,4,5]. This is a derivative free method.

The objective functions in problems (7) and (8) are piecewise partially separable (for the definition of piecewise partially separable functions, see [9]). The discrete gradient method was modified taking into account this special structure of the objective functions. This modified discrete gradient method is described in [10].

Conclusions

In this paper we discussed a nonsmooth optimization approach to clustering problems. Many clustering problems are large scale global optimization problems. The nonsmooth optimization approach allows one to significantly reduce the number of variables in this problem. It also can easily handle different similarity measures.

We introduced two algorithms based on the nonsmooth optimization approach. Both algorithms are incremental clustering algorithms. As these algorithms compute clusters step by step, they allow the decision maker to easily vary the number of clusters according to the criteria suggested by the nature of the decision making situation not incurring the obvious costs of the increased complexity of the solution procedure. The suggested approach utilizes a stopping criterion that prevents the appearance of small and artificial clusters. Nonsmooth optimization problems from cluster analysis have special structure which allows one to design efficient algorithms for their solution.

References

1. Al-Sultan KS (1995) A tabu search approach to the clustering problem. Pattern Recognit 28(9):1443–1451
2. Al-Sultan KS, Khan MM (1996) Computational experience on four algorithms for the hard clustering problem. Pattern Recognit Lett 17:295–308
3. Bagirov AM (1999) Minimization methods for one class of nonsmooth functions and calculation of semi-equilibrium prices. In: Eberhard A et al (eds) Progress in Optimization: Contribution from Australasia. Kluwer, Dordrecht, pp 147–175
4. Bagirov AM (2002) A method for minimization of quasidifferentiable functions. Optim Method Softw 17(1):31–60
5. Bagirov AM (2003) Continuous subdifferential approximations and their applications. J Math Sci 115(5):2567–2609
6. Bagirov AM, Mardaneh K (2006) Modified global *k*-means algorithm for clustering in gene expression data sets. In: Proceedings of the 2006 Workshop on Intelligent Systems for Bioinformatics, Hobart, Australia, 4 Dec 2006, pp 23–28
7. Bagirov AM, Rubinov AM, Yearwood J (2001) Using global optimization to improve classification for medical diagnosis and prognosis. Top Heal Inf Manag 22:65–74
8. Bagirov AM, Rubinov AM, Yearwood J (2001) Global optimization approach to classification. Optim Eng 22:65–74
9. Bagirov AM, Ugon J (2006) Piecewise partially separable functions and a derivative-free method for large-scale nonsmooth optimization. J Glob Optim 35(2):163–195
10. Bagirov AM, Ugon J (2005) An algorithm for minimizing clustering functions. Optimization 54(4–5):351–368
11. Bagirov AM, Yearwood J (2006) A new nonsmooth optimisation algorithm for minimum sum-of-squares clustering problems. Eur J Oper Res 170(2):578–596
12. Bock HH (1974) Automatische Klassifikation. Vandenhoeck & Ruprecht, Göttingen
13. Brown DE, Entail CL (1992) A practical application of simulated annealing to the clustering problem. Pattern Recognit 25:401–412
14. Clarke FH (1983) Optimization and Nonsmooth Analysis. Wiley, New York
15. Dubes R, Jain AK (1976) Clustering techniques: the user's dilemma. Pattern Recognit 8:247–260
16. Hansen P, Jaumard B (1997) Cluster analysis and mathematical programming. Math Program 79(1–3):191–215
17. Hansen P, Mladenovic N (2001) *J*-means: a new heuristic for minimum sum-of-squares clustering. Pattern Recognit 4:405–413
18. Hansen P, Mladenovic N (2001) Variable neighborhood decomposition search. J Heuristic 7:335–350
19. Hansen P, Ngai E, Cheung BK, Mladenovic N (2005) Analysis of global *k*-means, an incremental heuristic for minimum sum-of-squares clustering. J Classif 22:287–310

20. Houkins DM, Muller MW, ten Krooden JA (1982) Cluster analysis. In: Topics in Applied Multivariate Analysis. Cambridge University Press, Cambridge
21. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323
22. Likas A, Vlassis M, Verbeek J (2003) The global *k*-means clustering algorithm. Pattern Recognit 36:451–461
23. Reeves CR (ed) (1993) Modern Heuristic Techniques for Combinatorial Problems. Blackwell, London
24. Selim SZ, Ismail MA (1984) *K*-means-type algorithm: generalized convergence theorem and characterization of local optimality. IEEE Trans Pattern Anal Mach Intell 6:81–87
25. Selim SZ, Al-Sultan KS (1991) A simulated annealing algorithm for the clustering. Pattern Recognit 24(10):1003–1008
26. Spath H (1980) Cluster Analysis Algorithms. Ellis Horwood Limited, Chichester
27. Sun LX, Xie YL, Song XH, Wang JH, Yu RQ (1994) Cluster analysis by simulated annealing. Comput Chem 18:103–108

Nonsmooth and Smoothing Methods for Nonlinear Complementarity Problems and Variational Inequalities

NSM

Liqun Qi, Defeng Sun
School of Math., University New South Wales, Sydney, Australia

MSC2000: 90C33, 90C30

Article Outline

Keywords

Optimization; Nonsmooth methods; Smoothing methods; Variational inequalities; Complementarity problem

Given a nonempty closed set $X \subset \mathbf{R}^n$ and a mapping $F\colon \mathbf{R}^n \to \mathbf{R}^n$, assumed to be continuously differentiable, the *variational inequalities* (abbreviated: VI) are to find an $x^* \in X$ such that

$$(x - x^*)^\top F(x^*) \geq 0\,, \quad \forall x \in X\,. \tag{1}$$

When $X = \mathbf{R}^n_+$ (the positive orthant), (1) is equivalent to the *nonlinear complementarity problem* (abbreviated: NCP): Find $x^* \in \mathbf{R}^n$ such that

$$x^* \geq 0, \quad F(x^*) \geq 0, \quad x^{*\top} F(x^*) = 0\,. \tag{2}$$

Usually, X is represented by several inequalities and equalities. By considering the Karush–Kuhn–Tucker (KKT) conditions of (1) if necessary, X is assumed to be a closed convex subset of $\mathbf{R}^n$ here. It has been proved by B.C. Eaves in [9] that solving (1) is equivalent to finding a solution of the equation

$$H(x) := x - \Pi_X[x - F(x)] = 0\,, \tag{3}$$

where Π_X is the orthogonal projection onto X. When $X = \mathbf{R}^n_+$, (3) becomes

$$H(x) = \min(x, F(x)) = 0\,, \tag{4}$$

where the operation min is taken componentwisely. This means that finding a solution of NCP is equivalent to finding a root of (4). It also has been shown by A. Fischer in [10] that finding a solution of NCP is equivalent to finding a root of another equation, namely of:

$$H_i(x) := \phi(x_i, F_i(x)) = 0, \quad i = 1, \ldots, n\,, \tag{5}$$

where ϕ is called the *Fischer–Burmeister function* (FB function), defined in [10] as

$$\phi(a, b) = \sqrt{a^2 + b^2} - (a + b), \quad a, b \in \mathbf{R}\,.$$

Due to the nonsmoothness of the orthogonal projection operator Π_X, the min function and the FB function, the function H defined either in (3), in (4) or in (5) is, in general, not smooth (i. e. continuously differentiable) no matter how smooth F is. This prevents one from using classical Newton methods to find solutions of these (nonsmooth) equations.

Suppose that $H\colon \mathbf{R}^n \to \mathbf{R}^n$ is a locally Lipschitz function (the function H defined in either (3), (4) or (5) is a locally Lipschitz function) but is not necessarily smooth. By Rademacher's theorem, H is almost everywhere differentiable. Let

$$D_H = \{x\colon\ H \text{ is differentiable at } x\}\,.$$

Then the generalized Jacobian of H at x in the sense of F.H. Clarke [6] can be defined by

$$\partial H(x) = \operatorname{conv} \partial_B H(x) ,$$

where $\partial_B H(x)$ [20] is defined by

$$\partial_B H(x) = \left\{ \lim_{\substack{x^j \to x \\ x^j \in D_H}} H'(x^j) \right\} .$$

The *nonsmooth Newton method* for solving

$$H(x) = 0, \quad x \in \mathbf{R}^n , \tag{6}$$

can be defined as follows: Having the vector $x^k \in \mathbf{R}^n$, compute x^{k+1} by

$$x^{k+1} = x^k - V_k^{-1} H(x^k) , \tag{7}$$

where $V_k \in \partial H(x^k)$. The nonsmooth Newton method (7) reduces to the classical Newton method for a system of equations if H is continuously differentiable. The classical Newton method has the favorable feature that the sequence $\{x^k\}$ generated by (7) is locally superlinearly (quadratically) convergent to a solution x^* of $H(x) = 0$ if $H'(x^*)$ is nonsingular (and H' is Lipschitz continuous) [8,18]. However, in general the iterative method (7) is not convergent for nonsmooth equations (6). See [16] for a counterexample.

In order to establish some superlinearly convergent results for the nonsmooth Newton method (7), we use the concept of *semismoothness*. Let H be directionally differentiable at x. H is said to be *semismooth* at x if

$$Vd - H'(x; d) = o(\|d\|), \quad d \to 0 ,$$

and H is called strongly semismooth at x if

$$Vd - H'(x; d) = O(\|d\|^2), \quad d \to 0 ,$$

where $V \in \partial H(x + d)$. Semismoothness was originally introduced by R. Mifflin [17] for functionals. L. Qi and J. Sun [24] extended the concept of semismoothness to vector-valued functions. See [19] for several forms of semismooth equations. Using semismoothness, they [24] presented the following convergence theorem for the generalized Newton method (7):

Theorem 1 *Suppose that $H(x^*) = 0$ and that all $V \in \partial H(x^*)$ are nonsingular. Then the generalized Newton method (7) is Q-superlinearly convergent in a neighborhood of x^* if H is semismooth at x^*, and quadratically convergent if H is strongly semismooth at $x^{\wedge} *$.*

Note that the nonsingularity of $\partial H(x^*)$ in the above theorem is somewhat restrictive in some cases. Qi [20] presented a modified version of (7), which may be stated as follows

$$x^{k+1} = x^k - V_k^{-1} H(x^k) , \tag{8}$$

where $V_k \in \partial_B H(x^k)$. The difference of this version from (7) is that V_k is chosen from $\partial_B H(x^k)$ rather than the convex hull of $\partial_B H(x^k)$. Analogous to the above theorem, Qi [20] established the following result:

Theorem 2 *Suppose that $H(x^*) = 0$ and that all $V \in \partial_B H(x^*)$ are nonsingular. Then the generalized Newton method (8) is Q-superlinearly convergent in a neighborhood of x^* if H is semismooth at x^*, and quadratically convergent at x^* if H is strongly semismooth at x^*.*

In general, neither (7) nor (8) can be globalized because θ is not necessarily continuously differentiable, where for any $x \in \mathbf{R}^n$, $\theta(x) = \|H(x)\|^2/2$. However, if θ is continuously differentiable (e. g., θ is defined via (5); [14]), the nonsmooth Newton direction is a descent direction of θ and thus globalized methods can be designed. See [7] for a line search model and [13] for a trust region model.

The feature of *smoothing methods* is to construct a smoothing approximation function G: $\mathbf{R}^n \times \mathbf{R}_{++} \to \mathbf{R}^n$ of H such that for any $\varepsilon > 0$ and $x \in \mathbf{R}^n$, $G(\varepsilon, \cdot)$ is continuously differentiable on $\mathbf{R}^n$ and satisfies

$$\|H(x) - G(\varepsilon, x)\| \to 0 \quad \text{as } \varepsilon \downarrow 0 , \tag{9}$$

and then to find a solution of $H(x) = 0$ by (inexactly) solving the following problems for a given positive sequence $\{\varepsilon^k\}$ with $\varepsilon^k \to 0$ as $k \to \infty$,

$$G(\varepsilon^k, x) = 0 . \tag{10}$$

It was suggested in [21] to use the convolution to construct smooth approximations of the nonsmooth function H. A function Φ: $\mathbf{R}^n \to \mathbf{R}_+$ is called a *kernel function* if

$$\int_{\mathbf{R}^n} \Phi(x) \, \mathrm{d}x = 1 .$$

Suppose Φ is a smooth kernel function. Define Θ: $\mathbf{R}_{++} \times \mathbf{R}^n \to \mathbf{R}_+$ by

$$\Theta(\varepsilon, x) = \varepsilon^{-n}\Phi(\varepsilon^{-1}x)\,,$$

where $(\varepsilon, x) \in \mathbf{R}_{++} \times \mathbf{R}^n$. Then a smooth approximation of the projection operator Π_X can be described by

$$P(\varepsilon, x) = \int_{\mathbf{R}^n} \Pi_X(x - y)\Theta(\varepsilon, y)\,\mathrm{d}y\,, \tag{11}$$

where $(\varepsilon, x) \in \mathbf{R}_{++} \times \mathbf{R}^n$. Suppose that

$$\kappa := \int_{\mathbf{R}^n} \|y\|\,\Phi(y)\,\mathrm{d}y < +\infty\,.$$

Then for any $x \in \mathbf{R}^n$ and $\varepsilon > 0$ one has

$$\|P(\varepsilon, x) - \Pi_X(x)\| \le \kappa\varepsilon\,.$$

In general, $P(\varepsilon, x)$ is intractable because a multidimensional integration is involved. However, it can be written explicitly if X is of special structure (for example, X is a rectangular) and Φ is chosen particularly (see [3,12]). In fact, already in 1986 S. Smale [26] gave a smooth function $\frac{(w+\sqrt{w^2+\varepsilon^2})}{2}$ to approximate $\max(0, w)$, $w \in \mathbf{R}$, and used it to study linear complementarity problems. Also see [2,15]. The paper [1] stimulates much recent study about smoothing methods for solving NCP and VI. For the convenience of discussion, for any $\varepsilon < 0$, define $P(\varepsilon, x) = P(-\varepsilon, x)$ and $P(0, x) = \Pi_X(x)$, $x \in \mathbf{R}$. Then the smooth approximation G: $\mathbf{R}^{n+1} \to \mathbf{R}^n$ of H defined in (6) can be described by

$$G(\varepsilon, x) = x - P(\varepsilon, x - F(x))\,, \tag{12}$$

where $(\varepsilon, x) \in \mathbf{R} \times \mathbf{R}^n$. Thus, for any $(\varepsilon, x) \in \mathbf{R} \times \mathbf{R}^n$,

$$\|G(\varepsilon, x) - H(x)\| \le \kappa\varepsilon\,.$$

See [21] for a general case if H is not of the form defined in (3).

Note that, for variational inequalities, if the function F is only defined on X and not well defined outside X, then the function H defined in (3) is not well defined on $\mathbf{R}^n$. In this case, one can use the normal map introduced by S.M. Robinson [25] to overcome this difficulty. It is also noted that solving (1) is equivalent to finding a solution of the equation

$$H(x) := x - \Pi_X[x - F(\Pi_X(x))] = 0\,, \tag{13}$$

where $x \in \mathbf{R}^n$. The above-defined H only requires F being defined on X instead of on $\mathbf{R}^n$ as required by the function defined in (3). Unlike Robinson's normal map, the above map does not need to work on a transformed space, it works on the original space directly. By using the definition of P, the smoothing approximation G: $\mathbf{R}^{n+1} \to \mathbf{R}^n$ of H defined in (13) can be described by

$$G(\varepsilon, x) = x - P(\varepsilon, x - F(P(\varepsilon, x)))\,, \tag{14}$$

where $(\varepsilon, x) \in \mathbf{R} \times \mathbf{R}^n$.

The first globally and superlinearly (quadratically) convergent smoothing Newton method was proposed by X. Chen, Qi and D. Sun in [4], where the authors exploited a *Jacobian consistency property* and applied this property to an infinite sequence of smoothing approximation functions to get high-order convergent methods. The smoothing function defined by (12) satisfies the Jacobian consistence property while the one defined in (14) does not satisfy this property. The method in [4] was further studied by Chen and Y. Ye in [5].

Suppose that G: $\mathbf{R}^{n+1} \to \mathbf{R}^n$ is a smoothing approximation of H. Define E: $\mathbf{R}^n \to \mathbf{R}^n$ by

$$E(z) := \begin{pmatrix} \varepsilon \\ G(\varepsilon, x) \end{pmatrix}\,, \tag{15}$$

where $z := (\varepsilon, x) \in \mathbf{R} \times \mathbf{R}^n$. Then solving $H(x) = 0$ is equivalent to finding a solution of $E(\varepsilon, x) = 0$. Note that E is continuously differentiable at any $(\varepsilon, x) \in \mathbf{R} \times \mathbf{R}^n$ with $\varepsilon \ne 0$ and is possibly nonsmooth at $(0, x) \in \mathbf{R} \times \mathbf{R}^n$. We call E a *smoothing-nonsmooth reformulation* of H. Then classical Newton methods for solving smoothing equations can be used to solve $E(z) = 0$ with one additional requirement: ε must be positive during the process of iteration. The latter can be done by solving a slightly modified Newton equation:

$$E(z) + E'(z)\Delta z = \beta\overline{z}\,, \tag{16}$$

where $\beta \in (0, \infty)$ and $\overline{z} := (\overline{\varepsilon}, 0)$ with $\overline{\varepsilon} > 0$. It is obvious that one should control ε such that it neither converges too fast (no stronger global convergence results guaranteed) nor too slow (no fast local convergence results guaranteed). A special line search model involving β and $\overline{z}$ was designed in [23] to achieve this: Choose $\overline{\varepsilon} \in \mathbf{R}_{++}$ and $\gamma \in (0, 1)$ such that $\gamma\overline{\varepsilon} < 1$. Choose constants $\delta, \sigma \in (0, 1)$. Let $\varepsilon^0 := \overline{\varepsilon}, x^0 \in \mathbf{R}^n$ be an arbitrary point. For $k = 0, 1, \ldots$, find a solution Δz^k of (16) with

$z := z^k$ and $\beta := \gamma \min\{1, e(z)\}$, where for any $y \in \mathbf{R}^{n+1}$, $e(y) = \|E(y)\|^2$. Let l_k be the smallest nonnegative integer l satisfying

$$e(z^k + \delta^l \Delta z^k) \le [1 - 2\sigma(1 - \gamma\bar{\varepsilon})\delta^l]e(z^k) .$$

Define $z^k := z^k + \delta^{l_k} \Delta z^k$. It is often verified that G is also semismooth everywhere jointly with ε and x [23]. So the semismooth theory of nonsmooth Newton methods for solving nonsmooth equations can be used to obtain superlinear (quadratic) convergence of (ε, x) for the above smoothing Newton method while the global convergence is based on the particular designed line search. See [23] for details.

Conclusion

In this paper semismooth Newton methods and smoothing Newton methods for solving NCP and VI based on nonsmooth equations have been briefly reviewed. These topics are still undergoing a very fast development. See [22] for an up-to-date review. Another nonsmooth approach for solving NCP and VI is to reformulate these problems as unconstrained optimization problems whose objective functions are once but not twice differentiable, (see [11]).

See also

- ► Composite Nonsmooth Optimization
- ► Nonconvex-Nonsmooth Calculus of Variations
- ► Solving Hemivariational Inequalities by Nonsmooth Optimization Methods

References

1. Burke J, Xu S (1999) A polynomial time interior-point path-following algorithm for LCP based on Chen–Harker–Kanzow smoothing techniques. Math Program 86:91–103
2. Chen B, Harker PT (1993) A non-interior-point continuation method for linear complementarity problems. SIAM J Matrix Anal Appl 14:1168–1190
3. Chen C, Mangasarian OL (1996) A class of smoothing functions for nonlinear and mixed complementarity problems. Comput Optim Appl 5:97–138
4. Chen X, Qi L, Sun D (1998) Global and superlinear convergence of the smoothing Newton method and its application to general box constrained variational inequalities. Math Comput 67:519–540
5. Chen X, Ye Y (1999) On homotopy-smoothing methods for variational inequalities. SIAM J Control Optim 37:589–616
6. Clarke FH (1983) Optimization and nonsmooth analysis. Wiley, New York
7. DeLuca T, Facchinei F, Kanzow C (1996) A semismooth equation approach to the solution of nonlinear complementarity problems. Math Program 75:407–439
8. Dennis JE, Schnabel RB (1983) Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, Englewood Cliffs
9. Eaves BC (1971) On the basic theorem of complementarity. Math Program 1:68–75
10. Fischer A (1992) A special Newton-type optimization method. Optim 24:269–284
11. Fukushima M (1996) Merit functions for variational inequality and complementarity problems. In: Pillo GDi, Giannessi F (eds) Nonlinear Optimization and Applications. Plenum, New York, pp 155–170
12. Gabriel SA, Moré JJ (1997) Smoothing of mixed complementarity. In: Ferris MC, Pang JS (eds) Complementarity and Variational Problems: State of the Art. SIAM, Philadelphia, pp 105–116
13. Jiang H, Fukushima M, Qi L, Sun D (1998) A trust region method for solving generalized complementarity problems. SIAM J Optim 8:140–157
14. Kanzow C (1994) An unconstrained optimization technique for large-scale linearly constrained convex minimization. Computing 53:101–117
15. Kanzow C (1996) Some noninterior continuation methods for linear complementarity problems. SIAM J Matrix Anal Appl 17:851–868
16. Kummer B (1988) Newton's method for non-differentiable functions. In: Guddat J, Bank B, Hollatz H, Kall P, Klatte D, Kummer B, Lommatzsch K, Tammer L, Vlach M, Zimmerman K (eds) Adv. Mathematical Optimization. Akademie, Berlin, pp 114–125
17. Mifflin R (1977) Semismooth and semiconvex functions in constrained optimization. SIAM J Control Optim 15:957–972
18. Ortega JM, Rheinboldt WC (1970) Iterative solution of nonlinear equations in several variables. Acad. Press, New York
19. Pang J-S, Qi L (1993) Nonsmooth equations: Motivation and algorithms. SIAM J Optim 3:443–465
20. Qi L (1993) Convergence analysis of some algorithms for solving nonsmooth equations. Math Oper Res 18:227–244
21. Qi L, Chen X (1995) A globally convergent successive approximation method for severely nonsmooth equations. SIAM J Control Optim 33:402–418
22. Qi L, Sun D (1999) A survey of some nonsmooth equations and smoothing Newton methods. In: Hill R, Eberhard A, Glover B, Ralph D (eds) Progress in Optimization. Kluwer, Dordrecht, pp 121–146
23. Qi L, Sun D, Zhou G (2000) A new look at smoothing Newton methods for nonlinear complementarity problems and box constrained variational inequalities. Math Program 87:1–35

24. Qi L, Sun J (1993) A nonsmooth version of Newton's method. Math Program 58:353–367
25. Robinson SM (1992) Normal maps induced by linear transformation. Math Oper Res 17:691–714
26. Smale S (1986) Algorithms for solving equations. In: Proc. Internat. Congress Math, pp 172–195

NP-complete Problems and Proof Methodology

SANATAN RAI, GEORGE VAIRAKTARAKIS
Department OR and Operations Management, Case Western Reserve University, Cleveland, USA

MSC2000: 90C60, 68Q25

Article Outline

Keywords

Computational complexity; Reducibility; Polynomial time reduction; NP-hard problem; NP-complete problem; strong NP-completeness; ordinary NP-completeness

Given a combinatorial problem, one tries to exploit its structure so as to develop a solution algorithm that guarantees identifying an optimal solution for every instance of the problem. For some problems, the inherent structure is such that one can develop an algorithm that progressively builds an optimal solution, or selects among a small number of candidate solutions. Such algorithms are quite desirable as their computer time requirements are small, and often a bounded polynomial function of the number n of parameters needed to specify the problem (e. g. $\mathcal{O}(n)$ or $\mathcal{O}(n^2)$; see [9] for details on algorithmic *complexity*). Thus, such algorithms are called *polynomial algorithms*. Problems solvable by a polynomial algorithm may be solved quickly on a computer.

Unfortunately, not all combinatorial problems possess enough structure to allow for a polynomial algorithm. Hence, when we encounter a new problem for which we cannot identify enough structure, we would like to know whether this lack of structure is due to the problem itself, or to incomplete analysis. To address this issue, one idea is to compare the structure of the problem at hand with the structure of other well known and notoriously hard problems; such problems are known in literature as *NP-complete problems*. Specifically, if we can show that our problem is 'equivalent' to an NP-complete problem, then any algorithm that solves our problem can be used to solve the hard one and vice versa. Then, we can justifiably suggest that our problem is very difficult. With this information, we can either continue focusing our efforts in finding a *polynomial time optimal algorithm* (admitting that our chances for success are low) or consider heuristics or enumeration techniques.

If we are ever able to find a polynomial time algorithm for a *NP*-complete problem, we will make one of the most important discoveries in human knowledge. This will mean that we are able to solve all hard combinatorial problems very quickly (for details about the relationship between the class of polynomially solvable problems and the class of *NP*-complete problems see [4]). If we believe that this is unlikely, then we focus our analyses on enumeration techniques. Hence, in the latter case, the equivalence between the problem at hand and the *NP*-complete problem have dictated our approach towards the problem at hand. Since 1971 when the foundations of complexity theory were developed by S.A. Cook in [2], all the papers that have appeared in the literature have taken the latter route – namely, they focus on heuristics and/or enumeration techniques. In this sense, complexity theory is a very useful tool for determining our approach towards solving difficult combinatorial problems. In this article, we present some of the fundamental techniques that have been used in the literature to prove equivalence among problems. We start in the next section by presenting a list of combinatorial optimization problems. Then, we describe some basic methodology for theorem proving in complexity theory, and conclude with a few illustrative example proofs.

Some Known *NP*-Complete Problems

In what follows we present some problems commonly used in the literature to prove equivalence between problems. All of these problems are notoriously hard, they belong to the class of *NP*-complete problems, and hence, no polynomial algorithm is known for them. In the rest of this article we present methods for proving the equivalence between selected pairs among of these problems. Our selected problems span a sample of areas in combinatorial optimization including set partition, logic, graph theory, network theory, and scheduling theory.

The following two problems are representative of problems in set partition. The problems are stated in their decision form, i. e., we only require a 'yes/no' answer to resolve them. Each *instance* is described by the input data required to define the problem, and each *question* requires a 'yes/no' answer. This presentation of combinatorial problems follows the presentation form adopted in [4] which was the first text devoted to a systematic compilation of hard combinatorial problems.

Definition 1 (*Partition*) INSTANCE: Set $A = \{a_1, \ldots, a_n\}$ of elements, and a set function $s: A \to Z^+$.

QUESTION: Does there exist a set $A' \subset A$ such that

$$\sum_{a \in A'} s(a) = \frac{1}{2} \sum_{a \in A} s(a) \,?$$

Definition 2 (3-*Partition*) INSTANCE: Set $A = \{a_1, \ldots, a_{3n}\}$ of elements, set function $s: A \to Z^+$, and threshold value B.

QUESTION: Are there subsets $A_k \subset A$, $k = 1, \ldots, n$, such that

$$\sum_{a \in A_i} s(a) = B \text{ and } |A_k| = 3 \text{ for } 1 \le i \le n \,?$$

These problems are among the most popular problems found in complexity theory. Note that in 'Partition', A is partitioned in two sets with no restriction on the number of elements per set. For this reason, 'Partition' is often referred to in the literature as 2-*partition*. In contrast, '3-partition' involves the partition of A in n sets each consisting of precisely three elements.

Definition 3 (3-*Satisfiability*) INSTANCE: A Boolean expression B in literals x_i, $i = 1, \ldots, q$,

$$B = (p_{11} \vee p_{12} \vee p_{13}) \wedge \cdots \wedge (p_{n1} \vee p_{n2} \vee p_{n3}) = \wedge_{i=1}^{n} \vee_{j=1}^{3} p_{ij}$$

where each p_{ij} is either x_k or its negation $\overline{x}_k$ for some $1 \le k \le q$.

QUESTION: Is there an assignment for the literals x_k such that B is true?

This problem is also referred to in the literature as 3-*Sat*. The related problem where every clause has an arbitrary number of literals (rather than precisely three) is known in the literature as the *satisfiability* problem, or *Sat*, and has the distinction of being the first *NP*-complete problem (see [2]). Note that, if we were able to solve 'Sat' in polynomial time, then we would be able to determine the truth value of all possible statements in propositional calculus. Effectively, we could cast every imaginable theorem in propositional form, and let a computer answer it. This would be equivalent to theorem proving using computers.

Definition 4 (*Maximum clique*) INSTANCE: Graph $G = (V, E)$ and positive integer k.

QUESTION: Does there exist a complete subgraph of G on k vertices?

The 'maximum clique' is a very important problem in graph theory with applications in diverse fields. The following problem is encountered when one wants to identify a path (with certain properties) in a given network.

Definition 5 (*Impossible pairs constrained path problem* (IPP)) INSTANCE: Directed graph $G = (V, A)$ with source s and sink t, and pairs of nodes (a_i, b_i) for $i = 1, \ldots, n$.

QUESTION: Does there exist a directed $s - t$ path containing at most one node from each pair (a_i, b_i) for $i = 1, \ldots, n$?

The following problems are found in scheduling theory where a set of jobs is to be processed in a production system so as to optimize a given objective. We use the standard 3-field notation $\alpha/\beta/\gamma$ (see [6]) where α denotes the number and type of processors, β describes the job characteristics, and γ the objective function. For example, $\alpha = 1$ indicates a single processor, and α

= Pm denotes m identical processors operated in parallel. Job characteristics include completion deadlines, start times, precedence constraints among jobs, or processing characteristics. A popular scheduling objective is the minimization of the *makespan*, $C_{\max}$ – the maximum completion time, where the maximum is taken over all jobs. Evidently, $C_{\max}$ may be the preferred objective when a manager wants to maximize the utilization of processors.

Definition 6 $\mathbf{1/r_i, d_i/C_i \le d_i}$ INSTANCE: Set $J = \{J_1, \ldots, J_n\}$ of jobs, each with a processing time p_i, a duedate d_i, and a release time r_i, $1 \le i \le n$.

QUESTION: Is there a schedule of the jobs in J such that each job starts after time r_i and completes at time $C_i \le d_i$?

In the above problem one wants a single processor schedule where every job J_i starts no earlier than time r_i and finishes no later than time d_i. Such schedule would allow on time delivery of jobs to the customers.

Definition 7 $\mathbf{Pm//C_{max}}$ INSTANCE: Set $M = \{M_1, \ldots, M_m\}$ of parallel identical processors, set $J = \{J_1, \ldots, J_n\}$ of jobs each with a processing time p_i, and threshold value B.

QUESTION: Is there a nonpreemptive assignment of the n jobs to the m processors so that at any time every machine processes at most one job, and the completion time of J_i is $C_i \le B$ for every $1 \le i \le n$?

Here we seek a schedule of the n jobs on the m processors so as to minimize the completion time of the last job. Every processor can process at most one job at a time, and every job must be processed in its entirety without being interrupted by any of the processors.

Definition 8 $\mathbf{P//prec, p_i = 1/C_{max}}$ INSTANCE: Set $J = \{J_1, \ldots, J_n\}$ of jobs with processing time $p_i = 1$ for $1 \le i \le n$, and set A of precedence constraints between jobs in J. Also, set P of parallel identical processors, and a threshold value B.

QUESTION: Is there a nonpreemptive assignment of jobs to processors so that at any time every machine processes at most one job, the job precedence constraints are satisfied, and the completion time of J_i is $C_i \le B$ for every $1 \le i \le n$?

Unlike the previous problem, the number of parallel identical processors is not specified in this problem, i. e., $\alpha = P$. Every job has unit processing time. These unit jobs must satisfy a set of precedence constraints. Among all nonpreemptive schedules that satisfy these constraints, we seek one that minimizes the completion time of the last job.

The following section presents a general methodology for *NP*-completeness proofs with the problems described above as examples.

Methodology for *NP*-Completeness Proofs

We start by presenting the four basic steps of a complexity proof. Such proofs demonstrate that a new problem Π can be transformed to a known *NP*-complete problem $P \in \mathcal{NPC}$. To indicate this reduction we use the notation $P \propto \Pi$.

1) Show that $\Pi \in \mathcal{NP}$.
2) Construct a transformation from P to Π.
3) Show that the transformation in step 2 can be effected by a polynomial time algorithm.
4) Show that there exists a solution S_P for P, if and only if there exists a solution S_Π for Π, and that the transformations S_P to S_Π and vice versa is done by a pseudopolynomial algorithm for *strong NP-complete reductions*, and by a polynomial algorithm for *ordinary NP-complete reductions*.

Step 1 requires that, given a solution S_Π of Π we can check whether S_Π provides a 'yes' or 'no' answer for Π, using a polynomial algorithm. Given an arbitrary instance I of P, step 2 requires constructing an instance I' of Π. Step 3 requires that the construction of I' from I is polynomial on the number of input data required to specify I. Finally, Step 4 requires proving that, given a solution S_P for the instance I, we can construct a solution S_Π for I' and vice versa. In almost all reductions that have appeared in literature, steps 1–3 are quite simple and usually straightforward, while step 4 often requires considerable creativity.

Step 4 refers to strong and ordinary *NP*-complete problems. In a nutshell, this is one of many classifications of *NP*-complete problems into smaller subclasses. For a detailed description of these classes see [4]. In practical terms, an ordinary *NP*-complete problem can be solved using implicit enumeration algorithms like dynamic programming. In this case, the complexity of

the algorithm is not polynomial on the *length of input data*, but it is polynomial on the *size* of these data. For instance, Partition is a *NP*-complete problem solvable by dynamic programming in $\mathcal{O}(n \sum_i s(a_i))$ time (see [8]). Evidently, this complexity is polynomial on the size $\sum_i s(a_i)$ of the data. To see that this complexity bound is not polynomial on the length of the data, consider the binary encoding scheme. In this scheme each $s(a_i)$ can be represented by a string of length $\mathcal{O}(\log s(a_i))$, and hence $s(a_1), \ldots, s(a_n)$ can be described by a string of length $\mathcal{O}(\sum_i \log s(a_i))$ which is no greater than $\mathcal{O}(n \log B)$ where $B = \sum_i s(a_i)$. We see that the time complexity $\mathcal{O}(nB)$ of the dynamic program (DP)

- is polynomial on the size B of the data.
- but not polynomial on the length of the input data for the instance I of 'partition', where length(I) = $\mathcal{O}(n \log B)$.

Notice that the complexity of this DP, $\mathcal{O}(nB)$, is not bounded by any polynomial function of $n \log B$. When the complexity of an algorithm is polynomial on the size of the data, but not the length of the input, we refer to the algorithm as a *pseudopolynomial algorithm*. A *NP*-complete problem solvable by a pseudopolynomial algorithm is called *ordinary NP-complete*. Else, the problem is *strongly NP-complete*.

As indicated by its complexity, solving 'partition' is easier than solving any problem not solvable by implicit enumeration. Such problems require explicit enumeration algorithms like branch and bound. In the list of problems given earlier, 'partition' is the only problem solvable by dynamic programming.

Given the complexity status of the known *NP*-complete problem P, we can determine whether a new problem Π is strongly or ordinary *NP*-complete, if one of the following happens:

- P is strongly *NP*-complete, and $P \propto \Pi$. Then, Π is strongly *NP*-complete.
- P is ordinary *NP*-complete, $P \propto \Pi$, and a pseudopolynomial algorithm exists for Π. Then, Π is ordinary *NP*-complete.

As indicated in the following tree, all other outcomes result to incomplete determination of the exact complexity status of problem Π.

Example Proofs

In this section we present some simple applications of the four step reduction process outlined previously.

Example 9 Partition $\propto P2//C_{\max}$. Step 1 requires us to show that $P2//C_{\max} \in \mathcal{NP}$, i. e., given a schedule S of the n jobs, we can check whether the associated makespan $C_{\max}(S) \leq B$ in polynomial time. To perform the check, we need to find the completion time of the last job processed by each of the processors. This requires no more than n additions involving the processing times of the jobs in J. Hence, $C_{\max}(S)$ can be computed in $\mathcal{O}(n)$ time, and subsequently, whether $C_{\max}(S) \leq B$ or not can be established in $\mathcal{O}(1)$ time. Hence, $P2//C_{\max} \in \mathcal{NP}$. This completes step 1.

For step 2 we must construct an instance of $P2//C_{\max}$, given an instance of 'partition'. Let I be an instance of 'partition', and $s(a_1), \ldots, s(a_n)$ be the values of the n elements $a_1, \ldots, a_n$ in I. We construct an instance I' of $P2//C_{\max}$ as follows. Let $p_i = s(a_i)$, $i \in \{1, \ldots, n\}$. The number of processors is $m = 2$ and the number of jobs is n. The threshold value B is set to $B = (1/2) \sum_i p_i$. This completes Step 2.

This construction of I' required $n + 2$ assignments, and $n + 1$ basic operations to compute B. Evidently, the total amount of effort required to construct I' is $\mathcal{O}(n)$. This concludes step 3.

In step 4, we need to show that, there exists a solution for the instance I of 'partition' if and only if there exists a solution for the instance I' of $P2//C_{\max}$. Indeed, let A_1, A_2 be a partition of A such that

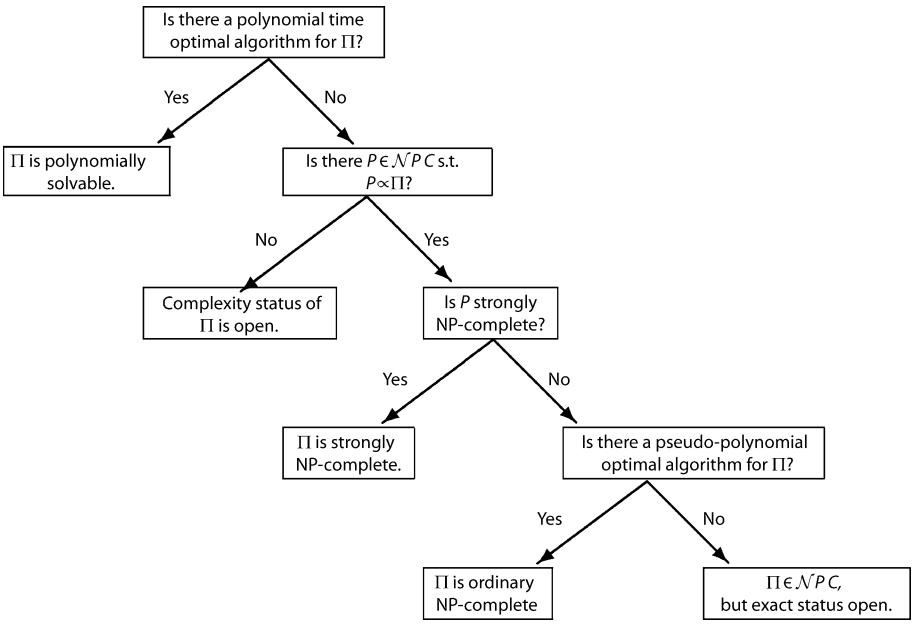
$$\sum_{a_i \in A_1} s(a_i) = \sum_{a_i \in A_2} s(a_i) = \frac{1}{2} \sum_{a_i \in A} s(a_i) .$$

From this, we can construct a solution for I' by assigning all jobs in $J^1 = \{J_i : a_i \in A_1\}$ to be processed (in any order) by processor M_1, and all jobs in $J^2 = \{J_i : a_i \in A_2\}$ to be processed (in any order) by processor M_2. Let S be the resulting schedule for $P2//C_{\max}$. By definition of A_1 and A_2,

$$\sum_{J_i \in J^1} p_i = \sum_{J_i \in J^2} p_i = \frac{1}{2} \sum_i p_i = B .$$

Clearly, the schedule S is constructed from A_1, A_2 in $\mathcal{O}(n)$ time. Similarly, given a schedule S that solves $P2//C_{\max}$ one can construct the partition A_1, A_2 in $\mathcal{O}(n)$ time as well.

Since 'partition' is an ordinary *NP*-complete problem, to completely determine the status of $P2//C_{\max}$, we will have to develop a pseudopolynomial algorithm for



NP-complete Problems and Proof Methodology, Figure 1
Establishing the complexity status of a problem Π

it. Such algorithm can in fact be developed (see [1]) which means that $P2//C_{max}$ belongs to the class of ordinary *NP*-complete problems.

Example 10 3-partition $\propto 1/r_i, d_i/C_i \leq d_i$ For Step 1, consider a given schedule S for $1/r_i, d_i/C_i \leq d_i$. By scanning the n jobs of S in sequence, we can obtain the start and completion times of the n jobs in $\mathcal{O}(n)$ time. Then, we can perform the n comparisons $r_i \leq C_i \leq d_i$ in $\mathcal{O}(n)$ time. Hence, $1/r_i, d_i/C_i \leq d_i \in \mathcal{NP}$. This completes step 1.

For Step 2, suppose we are given an instance I of 3-partition, that is, a set $A = \{a_1, \ldots, a_{3n}\}$ with associated values $s(a_1), \ldots, s(a_{3n})$, and threshold value B. For reasons that will become clear later, we make the following assumptions. Firstly, $s(a_i) < B$ for every $1 \leq i \leq 3n$, otherwise we can immediately conclude that there is no solution for the instance I of 3-partition. Secondly, we assume that the elements $a_1, \ldots, a_{3n}$ possess the property

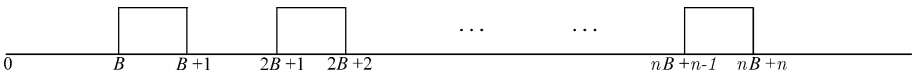
$*)$ $B/4 < s(a_i) < B/2$ for $i = 1, \ldots, 3n$.

Indeed, if this condition does not hold, we can replace the value of each element a_i by $s'(a_i) = s(a_i) + B$, and the threshold value B by $B' = 4B$. Then, it is easy to see that, $B < s'(a_i) < 2B$ (because $s(a_i) < B$), and hence $B'/4 < s'(a_i) < B'/2$ for $i = 1, \ldots, 3n$. In this case, the elements $a_1, \ldots, a_{3n}$ possess the required property with respect to the values $s'(a_i)$ and the threshold B'. Hence, without loss of generality, we assume that the elements of the instance I of 3-partition satisfy property $*$). This ensures that, if $\sum_{a_i \in A_k \subset A} s(a_i) = B$, then $|A_k| = 3$.

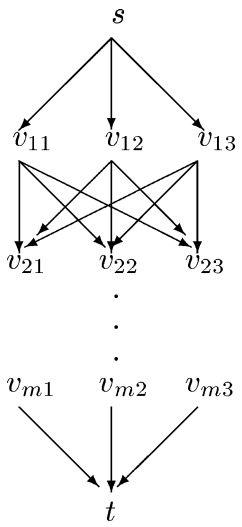
From I, we construct the instance I' of $1/r_i, d_i/C_i \leq d_i \in \mathcal{NP}$ with $4n$ jobs as follows. Set, $p_i = s(a_i)$, $r_i = 0$ and $d_i = nB + n - 1$ for $i = 1, \ldots, 3n$. Also include in I' n 'filler' jobs, $F_1, \ldots, F_n$ with processing times $p_{F_i} = 1$, start times $r_{F_i} = iB + i - 1$ and due-dates $d_{F_i} = iB + i$ for $i = 1, \ldots, n$. Note that $p_{F_i} = d_{F_i} - r_{F_i} = 1$ so the filler jobs have no slack. This completes step 2.

Clearly, The construction of I' is done in $\mathcal{O}(n)$ time. To prove step 4, let A_k, $k = 1, \ldots, n$ be a solution of I. By definition of the 3-partition problem, each set A_k is such that $|A_k| = 3$, and $\sum_{a \in A_k} s(a) = B$, $k = 1, \ldots, n$. From this, construct a schedule S for I' as follows. Let $J^k = \{J_i: a_i \in A_k\}$ for $k = 1, \ldots, n$, and schedule the jobs in J^k to be processed (in any order) starting at time $(k-1)B + k - 1$ and finishing at time $kB + k - 1$ as in Fig. 2.

Evidently, the schedule S is constructed in $\mathcal{O}(n)$ time. Alternatively, if S is a schedule that solves I', then the filler jobs should partition the $3n$ regular jobs in n distinct sets, say $J^k = \{J_i : J_i$ starts after F_{k-1} and is com-



NP-complete Problems and Proof Methodology, Figure 2
Instance of 1/r_i/C_i



NP-complete Problems and Proof Methodology, Figure 3
Instance of IPP

pleted before F_k} for $k = 1, \ldots, n$. Note that for $k = 1$, J^1 contains jobs that start after time $t = 0$ (F_0 is not defined) and complete prior to F_1. As the filler jobs have no slack, in each J_k the sum of the processing times must be precisely B units. By construction of the instance I, each J^k must contain precisely 3 jobs. Therefore, the sets $A_k = \{a_i \in A: J_i \in J^k\}$ for $k = 1, \ldots, n$ partition A into n triplets such that $\sum_{a_i \in A_k} s(a_i) = \sum_{J_i \in J^k} p_i = B$ for $k = 1, \ldots, n$, i. e., the collection $\{A_k\}_{k=1}^n$ solves I. Hence, given a schedule S that solves I', we can construct a partition $\{A_k\}_{k=1}^n$ that solves I, in $\mathcal{O}(n)$ time. This completes step 4.

This example demonstrates that we can often construct the instance I so that it satisfies certain properties that may become handy in proving step 4. As seen so far, quite often steps 1 and 3, as well as parts of step 4 are trivial, in such cases the details are often omitted. A typical example is provided next.

Example 11 3-Sat $\propto$ *IPP* Given a Boolean expression B, construct a digraph G as indicated in Fig. 3.

Specifically, the arc set of G is

$$E = \{(s, v_{1j}): \ j = 1, 2, 3\} \cup \{(v_{ij}, v_{i+1,k}): \ 1 \le i \le m-1; \ 1 \le j, k \le 3\} \cup \{(v_{mj}, t): \ j = 1, 2, 3\}.$$

Also, let the set of restricted pairs be $M = \{(v_{ij}, v_{kl}): \ p_{ij} = \overline{p}_{kl}\}$.

Let $P = sv_{1l_1} \ldots v_{ml_m} t$ be a constrained path. By making p_{il_i} true for $i = 1, \ldots, m$, we force B to be true. Since P satisfies the constraints in M, these assignments do not conflict. Therefore the existence of a path P implies the satisfiability of B and vice versa. This construction took only $\mathcal{O}(m)$ time, and the reduction is complete.

Our last example demonstrates how complexity theory can be used to derive approximation results.

Example 12 Max Clique $\propto$ *P/prec,* $p_j = 1/C_{max}$ Given $G = (V, E)$ construct a digraph D as follows. Introduce a job J_v for every $v \in V$, a job J_e for every $e \in E$, and an arc $J_v \to J_e$ whenever v is an endpoint of e. Let $l = \binom{k}{2}$ be the number of edges in a k-clique, $k' = |V| - k$ the number of nodes not in the clique and $l' = |E| - l$, the number of edges not in the clique.

Let the number of processors be $m = \max(k, l+k', l') + 1$. Introduce three sets of dummy nodes

$$X_x, \quad x = 1, \ldots, m-k,$$
$$Y_y, \quad y = 1, \ldots, m-l-k',$$
$$Z_z, \quad z = 1, \ldots, m-l'.$$

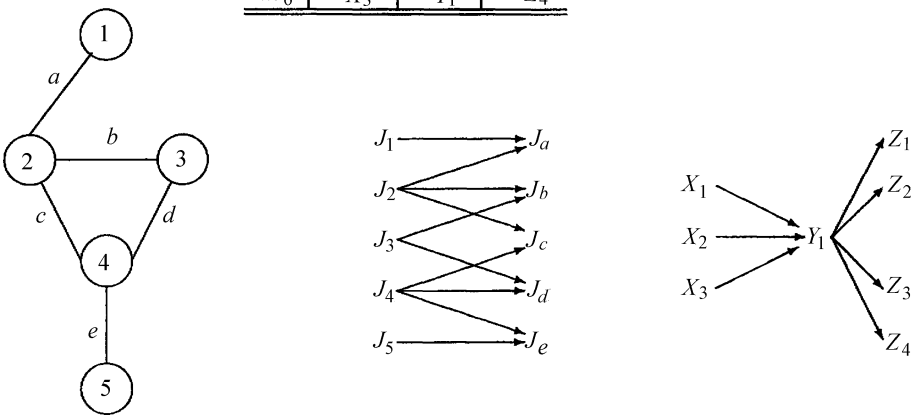
and the arcs $X_x \to Y_y \to Z_z$. Note that the total number of jobs is $3m$.

We will show that there exists a feasible schedule for the instance I of $P/\text{prec}, p_j = 1/C_{\max} = 3$ if and only if there exists a clique of size k. Or, in other words, there is a clique of size k if and only if we can complete the jobs in 3 periods.

($\Leftarrow$). Suppose a k-clique exists. Then,

- Schedule the jobs corresponding to the k clique vertices in period 1 (see Fig. 4).

	1	2	3
m_1	J_2	J_b	J_a
m_2	J_3	J_c	J_e
m_3	J_4	J_d	Z_1
m_4	X_1	J_1	Z_2
m_5	X_2	J_5	Z_3
m_6	X_3	Y_1	Z_4



NP-complete Problems and Proof Methodology, Figure 4
Instance of *P*/prec, $p = 1/C_{max}$

- Schedule the $m-k$ jobs corresponding to each X_x in the remaining processors in the first period.
- Schedule the l jobs corresponding to the clique edges in period 2, schedule the $k' = m - k$ remaining vertex jobs in period 2, and the $m - l - k'$ jobs from Yy in period 2.
- Finally, schedule the $l' = |\mathrm{E}| - l$ remaining edge jobs in period 3 and the $m - l'$ jobs from Z_z in period 3 (see Fig. 4).

This is a feasible schedule on three periods.

($\Rightarrow$). We prove this by contradiction. Suppose no k-clique exists. We will demonstrate that no 3-period schedule can exist either. Since there are m machines and $3m$ jobs, in a three period schedule all machines must be busy for all the three periods. Certainly, due to the precedence constraints $X_x \rightarrow Y_y \rightarrow Z_z$, all $m - k$ jobs must be done in period 1, all Y_y in period 2 and all the Z_z in period 3. Then, the remaining k slots in period 1 must be taken by vertex jobs. Since no k-clique exists, the edge jobs that can be scheduled in period 2 is no greater than $l - 1$. Then the jobs scheduled in period 2 are the $l - 1$ edge jobs, all the Y_y jobs and the $m - k$ remaining vertex jobs. The total number of these jobs does not exceed $m - 1$. Hence, there must be at least one machine that stays idle for a period. Hence no schedule on three periods can exist, which is a contradiction.

This completes the reduction, and since 'Max Clique' is strongly *NP*-complete, so is *P*/prec, $p_j = 1/C_{max}$.

Observe that, in Example 12, the number m of processors is not fixed. Therefore, Example 12 does not settle the complexity status for the problem in which m is specified beforehand. For example, it does not resolve the complexity status of *P*3/prec, $p_j = 1/C_{max}$. A careful examination of Example 12 can enable us to construct error bounds. Suppose we develop a heuristic H to solve *P*/prec, $p_j = 1/C_{max}$. Then, given an instance of Max Clique, we generate the associated instance of *P*/prec, $p_j = 1/C_{max}$ described in Example 12, and apply H on it. Since H cannot always solve *P*/prec, $p_j = 1/C_{max}$ optimally (unless $\mathcal{P} = \mathcal{NP}$), there will be instances where an optimal schedule on three periods exists, but our heuristic H fails to find it. Since the instance of *P*/prec, $p_j = 1/C_{max}$ produced in Example 12 uses integer processing times, whenever H fails to produce an optimal solution, its makespan C_H will be $C_H \geq 4$ rather than 3. Equivalently, the worst-case error bound for H must be $p = C_H/C^* \geq 4/3$. This observation is true for any pos-

sible heuristic for $P/\text{prec}, p_j = 1/C_{\max}$. Therefore, unless $\mathcal{P} = \mathcal{NP}$, no heuristic algorithm can exist with worst-case error bound $p < 4/3$. Therefore, research efforts for worst-case analyses for $P/\text{prec}, p_j = 1/C_{\max}$ must be focused on values of $p = 4/3$ or larger.

Conclusion

In this article we presented some basic techniques used in proving *NP*-completeness results. Following Cook's seminal paper [2], the first list of reductions for combinatorial problems was compiled in [5]. The four examples described in this article are illustrative of diverse optimization areas. They can be found in [3,4,7], and [6], respectively.

See also

- ▶ Complexity Classes in Optimization
- ▶ Complexity of Degeneracy
- ▶ Complexity of Gradients, Jacobians, and Hessians
- ▶ Complexity Theory
- ▶ Complexity Theory: Quadratic Programming
- ▶ Computational Complexity Theory
- ▶ Fractional Combinatorial Optimization
- ▶ Information-Based Complexity and Information-Based Optimization
- ▶ Kolmogorov Complexity
- ▶ Mixed Integer Nonlinear Programming
- ▶ Parallel Computing: Complexity Classes

References

1. Cheng TCE, Sin CCS (1990) A state-of-the-art review of parallel-machine scheduling research. Europ J Oper Res 47:271–292
2. Cook SA (1971) The complexity of theorem proving procedures. In: Proc. 3rd Annual ACM Symposium on Theory of Computing. ACM, New York, pp 151–158
3. Gabow HN, Maheshwari SN, Osterweil L (1976) On two problems in the generation of program test paths. IEEE Trans. Software Engin., London, pp 227–231
4. Garey MR, Johnson DS (1979) Computers and intractability. Freeman, New York
5. Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of Computer Computations. Plenum, New York, pp 85–103
6. Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (1993) Sequencing and scheduling: Algorithms and complexity. In: Graves SC, Rinnooy Kan AHG, Zipkin P (eds) Handbook Oper. Res. and Management Sci.: Logistics of Production and Inventory. North-Holland, Amsterdam, pp 445–522
7. Lenstra JK, Rinnooy Kan AHG, Brucker P (1977) Complexity of machine scheduling problems. Ann Discret Math 1:343–362
8. Martello S, Toth P (1990) Knapsack problems: Algorithms and computer implementations. Wiley, New York
9. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization: Algorithms and complexity. Prentice-Hall, Englewood Cliffs

Numerical Methods for Unary Optimization

EMILIO SPEDICATO[1], NAIYANG DENG[2], JIANZHONG ZHANG[3], SHAOLIN XI[4]

[1] Department Math., University Bergamo, Bergamo, Italy

[2] Department Basic Sci., China Agricultural University, Beijing, China

[3] Department of Mathematics, City University of Hong Kong, Kowloon Tong, Hong Kong

[4] Computer Institute, Beijing Polytechnic University, Beijing, China

MSC2000: 90C30

Article Outline

Keywords

Unary optimization; Partial-update Newton method; Goldfarb–Wang algorithm; Inexact Newton method

We give a survey of numerical methods for the *unary optimization* problem: $\min f(x) = \sum_{i=1}^{m} U_i(\alpha_i(x))$, $x \in \mathbf{R}^n$, where $U(\cdot)$ is a function of a single argument and $\alpha_i(x) = a_i^\top x$, $a_i \in \mathbf{R}^n$, $i = 1, \ldots, m$.

Introduction

Consider the unconstrained optimization problem:

$$\min f(x), \quad x \in \mathbf{R}^n . \tag{1}$$

If $f(x)$ takes the form

$$f(x) = \sum_{i=1}^{m} U_i(\alpha_i(x)), \quad x \in \mathbf{R}^n , \tag{2}$$

where $U_i(\cdot)$ is a smooth function of a single argument and $\alpha_i(x) = a_i^\top x$, $a_i \in \mathbf{R}^n$, $i = 1, \ldots, m$, then we call problem (1) a *unary optimization problem*. Usually $m \geq n$. Unary optimization problems appear in many practical fields such as linear robust regression, portfolio selection and chemical equilibrium, see, e. g. [1,2,9].

The unary optimization problem was first proposed by G.P. McCormick and A. Sofer [9]. They exploited the special structure of derivatives of the unary function and applied the resulting algorithm to the solution of the classical chemical equilibrium problem. D. Goldfarb and S. Wang [7] first proposed an implementable algorithm specially for the unary optimization problem. Using the special structure of unary functions, in order to save computation cost, they modified the regular Newton method to develop a *partial-update Newton method*. In recent years some Chinese scholars worked on this subject and published several papers. We will review the state of art in this new interesting field.

Goldfarb–Wang Algorithm GW

If the unary functions $U_i(\cdot)$, $i = 1, \ldots, m$, in (2) are all differentiable and the Hessian matrix $\nabla^2 f(x)$ is nonsingular for all $x \in \mathbf{R}^n$, then we can use Newton method to solve the problem (1), i. e., starting from the current point x^k, we compute the new iterate by

$$x^{k+1} = x^k + s^k ,$$

where the increment s^k is obtained by solving the Newton equation

$$\nabla^2 f(x^k)s = -\nabla f(x^k) . \tag{3}$$

Goldfarb and Wang noticed that the Hessian $\nabla^2 f(x)$ of the unary function has the following form:

$$\nabla^2 f(x) = \sum_{i=1}^{m} \phi_i(x) a_i a_i^\top , \tag{4}$$

where

$$\phi_i(x) = d^2 U_i \frac{\alpha_i(x)}{d\alpha_i^2} . \tag{5}$$

Let $\Phi(x) = \mathrm{diag}(\phi_1(x), \ldots, \phi_m(x))$, and $A^\top = (a_1, \ldots, a_m)$. Then we can write

$$\nabla^2 f(x) = A^\top \Phi(x) A . \tag{6}$$

Notice that only $\phi_i(x)$ on the right-hand side of (4) or (6) may vary when x changes and the rank of the matrix $\phi_i(x)\, a_i a_i^\top$ is one for any nonzero value ϕ_i. Suppose that only the jth diagonal element of $\Phi(x^k)$ and $\Phi(x^{k-1})$ differs at iteration k, i. e.,

$$A^\top \Phi(x^k) A = A^\top \Phi(x^{k-1}) A + (\phi_j(x^k) - \phi_j(x^{k-1})) a_j a_j^\top , \tag{7}$$

or equivalently,

$$\nabla^2 f(x^k) = \nabla^2 f(x^{k-1}) + (\phi_j(x^k) - \phi_{j-1}(x^k)) a_j a_j^\top . \tag{8}$$

Therefore, $\nabla^2 f(x^k)^{-1}$ can be obtained from $\nabla^2 f(x^{k-1})^{-1}$ by the well-known *Sherman–Morrison rank-one update formula*, and the next iteration can be obtained in only $O(n^2)$ arithmetic operations after evaluating ∇f and ϕ_i, $i = 1, \ldots, m$.

The essential point of the GW algorithm is, in the kth iteration, to construct an approximation B^k to the Hessian $\nabla^2 f(x^k)$. Goldfarb and Wang compute the increment s^k by solving the approximate Newton equation

$$B^k s = -\nabla f(x^k) , \tag{9}$$

where $B^k = A^\top \Phi^k A$, and $\Phi^k = \mathrm{diag}(\phi_1^k, \ldots, \phi_m^k)$. So their method can be viewed as a special type of *inexact Newton method*. The diagonal matrix Φ^k is introduced to approximate $\Phi(x^k)$, and is defined by setting, for $i = 1, \ldots, m$, $\phi_i^0 = \phi_i(x^0)$, and

$$\phi_i^{k+1} = \phi_i^k , \tag{10}$$

if $\phi_i(x^{k+1})$ is 'replaceable' by ϕ_i^k; and

$$\phi_i^{k+1} = \phi_i(x^{k+1}) \quad \text{otherwise}, \tag{11}$$

for $k \geq 0$. Choleski factorization is then used to solve (9). Notice that

$$B^{k+1} = B^k + \sum_{i \in S^k} (\phi_i^{k+1} - \phi_i^k) a_i a_i^\top . \tag{12}$$

where S^k is the set of indices i such that $\phi_i(x^{k+1})$ is not 'replaceable' by ϕ_i^k, $i = 1, \ldots, m$. We can exploit rank-one updates starting from the Choleski factorization of B^k, which needs $O(|S^k|n^2)$ operations, where $|S^k|$ is the number of elements in S^k. So it is possible to save computation cost if $|S^k|$ is not very large.

Consider the assumptions:

A1) there exists a point $x^* \in \mathbf{R}^n$ with $\nabla f(x^*) = 0$;

A2) $\nabla^2 f(x^*)$ is nonsingular;

A3) $\Phi(x)$ is continuous in a neighborhood of x^*.

Goldfarb and Wang proved the following

Theorem 1 *Let assumptions A1)–A3) hold. Then there exists $\delta > 0$ such that if $\|x^0 - x^*\| \le \delta$, the sequence x^k generated by the partial-update Newton method converges to $x*$. Moreover, the convergence is linear, i. e.,*

$$\left\|x^{k+1} - x^k\right\| \le t \left\|x^k - x^*\right\| , \quad \forall k \tag{13}$$

where $0 < t < 1$.

In order to get a higher rate of convergence, they proposed two 'replacement' criteria for the method:

- Criterion 1: For $i = 1, \ldots, m$, ϕ_i^k is replaceable by ϕ_i^{k-1} if

$$\frac{\left|\phi_i(x^k) - \phi_i^{k-1}\right|}{\left|\phi_i(x^k) - \phi_i^{k-1}\right| + \left\|\nabla f(x^k)\right\|} < \eta , \tag{14}$$

  where $0 < \eta < 1$.
- Criterion 2: For $i = 1, \ldots, m$, ϕ_i^k is replaceable by ϕ_i^{k-1} if $k \le p$ or

$$\frac{\left|\phi_i(x^k) - \phi_i^{k-1}\right|}{\max_{k-p+1 \le j \le k} \left|\phi_i(x^k) - \phi_i(x^{j-1})\right|} \le 1 , \tag{15}$$

  where p is a given positive integer.

Consider the additional assumption

A4) $\Phi(x)$ is Lipschitz continuous at x^*.

The authors proved the following

Theorem 2 *Let assumptions A1), A2) and A4) hold and let x^k be the sequence generated by GW algorithm. Then*

1) *x^k is locally quadratically convergent to x^* if Criterion 1 is used;*
2) *x^k is locally superlinearly convergent to x^* with R-order at least r_p, where r_p is the unique positive root of $t^{p+1} - t^p - 1 = 0$ if Criterion 2 is used.*

In order to obtain global convergence of the method, the authors modified the 'working approximation' Φ^k of $\Phi(x)$ to a $\widehat{\Phi}^k$, i. e., they modified the replacement criterion to ensure $A^\top \widehat{\Phi}^k A$ to be positive definite. But the modified globally convergent algorithm is only R-linearly convergent.

The authors' numerical results show the method takes less time to solve some types of problems than the modified Newton methods, but not always so.

Analysis of the Replacement Criterion

The GW algorithm opened a new field for unary optimization, but, to our knowledge, during nearly five years no more papers in this field were published. Recently (as of 1999), J.Z. Zhang, N.Y. Deng and L.H. Chen [10] discussed the efficiency of the GW algorithm. They asked whether it was more efficient than Newton's method when x^k approaches the solution x^*. They showed that, generally speaking, we can not expect the number $\|S^k\|$ to be small enough such that the computation cost to factorize B^{k+1} by rank-one updates is less than that to factorize $\nabla^2 f(x^{k+1})$ directly. The efficiency of GW algorithm heavily depends on the magnitude of the number $|S^k|$, so the frequency by which the replacement takes place is a key point. Based on this idea, the authors extended the Criterion 1 of GW algorithm to the so-called

- Criterion R_α: For $i = 1, \ldots, m$ we replace $\phi_i(x^{k+1})$ by ϕ_i^k if

$$\left|\phi_i(x^{k+1}) - \phi_i^k\right| < \sigma \left\|\nabla f(x^{k+1})\right\|^\alpha , \tag{16}$$

  where σ and α are two constants satisfying $\sigma > 0$ and $\alpha \in (0, 1]$.

It is easy to see that the GW's Criterion 1 is a special case of Criterion R_α with $\alpha = 1$ and $\sigma = \eta/(1 - \eta)$.

Consider the assumptions

A1) For $i = 1, \ldots, m$, $U_i(\cdot)$ is three times continuously differentiable;

A2) problem (1) has a solution x^* and $\nabla^2 f(x^*)$ is positive definite;

A3) the initial point x^0 is close enough to the solution x^*.

They proved two theorems:

Theorem 3 *Let assumptions A1)–A3) hold. Consider the sequence x^k generated by Algorithm R_α with $\alpha > \overline{\alpha}_{J+1}$, where J is a nonnegative integer and $\overline{\alpha}_{J+1}$ is the*

unique positive root of the equation

$$(1+\alpha)^{J+1}\alpha = 1\,. \tag{17}$$

Suppose that for a fixed i, we have
1) $d^3U_i(\alpha_i(x^*)/d\alpha_i^3 \neq 0$, and
2) there exists $\overline{K} > 0$ such that

$$\phi_i^k \neq \phi_i(x^*)\,, \forall k > \overline{K}\,. \tag{18}$$

Then
a) for any $K' > 0$, there exists an integer $k > K'$ such that

$$\phi_i^k = \phi_i(x^k)\,; \tag{19}$$

b) for any $\epsilon > 0$, there exists a $K > 0$ such that if

$$\overline{k} > K\,, \tag{20}$$

$$\phi_i^{\overline{k}} = \phi_i(x^{\overline{k}})\,, \tag{21}$$

and

$$\left| a_i^\top \frac{s^{\overline{k}}}{\left\| s^{\overline{k}} \right\|} \right| > \epsilon\,. \tag{22}$$

Then, for $x^k = x^{\overline{k}+1}, x^{\overline{k}+2}, \ldots, \phi_i(x^k)$ is replaceable by ϕ_i^{k-1} successively for at most J times.

Theorem 4 *The conditions are the same as Theorem 3. Suppose there exist M_1, $M_2 > 0$ such that when k is large enough,*

$$M_1 \left\| x^k - x^* \right\|^{1+\alpha} \leq \left\| x^{k+1} - x^* \right\| \tag{23}$$

$$\leq M_2 \left\| x^k - x^* \right\|^{1+\alpha}\,. \tag{24}$$

Then there exists $\overline{K} > 0$, such that if

$$\overline{k} > \overline{K} \tag{25}$$

and

$$\phi_i^{\overline{k}} = \phi_i(x^{\overline{k}})\,. \tag{26}$$

Then for $x^k = x^{\overline{k}+1}, x^{\overline{k}+2}, \ldots, \phi_i(x^k)$ is replaceable by ϕ_i^{k-1} successively for at least J times.

When $J = 0$, then the positive root of $(1 + \alpha)\alpha = 1$ is $\overline{\alpha}_1 = \frac{(\sqrt{5}-1)}{2}$. For GW's Criterion 1, $\alpha = 1 > \overline{\alpha}_1$, so from Theorem 3, $\phi_i(x^k)$ will never be replaceable by ϕ_i^{k-1} when k is large enough. So the GW algorithm with Criterion 1 cannot be expected to be more efficient than Newton method. The authors' numerical results also supported this conclusion.

But the authors pointed out, the idea of the GW algorithm to use the approximate Hessian B^k is promising, and efficient algorithms can hopefully be constructed.

Using the Trust Region Approach

In GW algorithm, in order to ensure global convergence, the authors modified the criteria such that the approximate Hessian B^k be positive definite. However, this prevents the algorithms from having a locally superlinear convergence, may lead to B^k being a poor approximation and may increase the computation cost. Motivated by this observation, Chen, Deng and Zhang [3] removed the requirement for B^k to be positive definite, and proposed two modified partial-update algorithms based on trust region stabilization. They also improved the replacement criteria of GW algorithm and constructed the matrix B^k in a different way such that it may be a better approximation to the Hessian.

Besides the replacement of the line search technique by the trust region strategy, the main differences between their algorithms and the GW algorithm are as follows:

a) As mentioned above, in order to get a better approximation to $\nabla^2 f(x^k)$, they abandoned the positive definiteness requirement for B^k. This change requires an efficient method to handle a trust region subproblem with indefinite matrix, so they used the indefinite dogleg method suggested in [11]. The main feature of the method is that it solves the TR subproblem approximately. It first uses a Bunch–Parlett (B-P) factorization for B^k, and then forms a dogleg curve. Instead of minimizing the objective function within the whole trust region, it makes a curvilinear search along the dogleg curve in the region.
b) They introduced a threshold value $l \approx n/6$ to control the algorithm to get the B-P factorization directly or to use a rank-one update, this motivated by the fact that the number of multiplications of the two methods is

$$c_n = \frac{1}{6}n^3 + O(n^2)$$

1 Set parameters $p > 0$, positive integers w and l. Choose the initial point $x^0 \in \mathbf{R}^n$. Set $k = 0$.

2 IF $\nabla f(x^k) = 0$, stop.

3 IF $k = 0$, go to Step 4; ELSE, define the set

$$S^k = \left\{ i : \left|\phi_i(x^k) - \phi_i^{k-1}\right| > \left|\left|\nabla f(x^k)\right|\right|^{1/p} \right\}. \tag{30}$$

IF

$$\left|S^k\right| \le w, \tag{31}$$

where $|\ S^k\ |$ is the number of elements in the set S^k, go to Step 5; otherwise go to Step 4.

4 Set

$$\phi_i^k = \phi_i(x^k), \quad i = 1, \ldots, m, \quad B^k = \nabla^2 f(x^k). \tag{32}$$

Compute the Choleski factorization $\nabla^2 f(x^k) = L_k D_k L_k^\top$ and the increment s^k by solving the Newton equation

$$\nabla^2 f(x^k) s = -\nabla f(x^k).$$

Set $x^{k+1} = x^k + s^k$. Go to Step 7.

5 Set

$$\phi_i^k = \begin{cases} \phi_i(x^k), & i \in S^k; \\ \phi_i^{k-1}, & i \notin S^k, \end{cases} \tag{33}$$

and

$$B^k = B^{k-1} + \sum_{i \in S^k} (\phi_i(x^k) - \phi_i^{k-1}) a_i a_i^\top. \tag{34}$$

Compute the Choleski factorization $B^k = L_k D_k L_k^\top$ which is obtained by successive $|\ S^k\ |$ rank-one corrections from the Choleski factorization

$$B^{k-1} = L_{k-1} D_{k-1} L_{k-1}^\top.$$

6 The increment $\tilde{s}^k$ is given by

$$\tilde{s}^k = \psi(\nabla^2 f(x^k),\ \nabla f(x^k), B^k, l), \tag{35}$$

where $\psi(\nabla^2 f(x^k),\ \nabla f(x^k), B^k, l)$ is a mapping defined by solving approximately the Newton equation

$$\nabla^2 f(x) s = -\nabla f(x) \tag{36}$$

with preconditioned conjugate gradient inner iterations, and the preconditioner is given by the inverse of B. The initial point is selected as $s = 0$. The number of inner iterations is l. The PCG method used here is a standard one, e.g. see [8, Algorithm 2.5.1]. Set $x^{k+1} = x^k + \tilde{s}^k$.

7 Set $k = k + 1$, and then go to Step 2.

◄ **Numerical Methods for Unary Optimization, Algorithm 1 Deng-Wang-Zhang algorithm**

and

$$c_1 = n^2 + O(n)$$

respectively, and $c_n/c_1 = n/6$. That is, if $k = 1$ or $|S^k| > l$, one performs the B-P factorization of B^k directly; otherwise, one uses the rank-one updates of the B-P factorization $|S^k|$ times to get the factorization of the matrix

$$B^k = B^{\tau_k} + \sum_{i \in S^k} (\phi_i^k - \phi_i^{\tau_k}) a_i a_i^\top$$

where τ_k is the index of the iteration at which the latest B-P factorization was performed.

c) They considered the effect of a_i on the replacement criteria. From the expression

$$\nabla^2 f(x) = \sum_{i=1}^m [\|a_i\|^2 \phi_i(x)] \left(\frac{a_i}{\|a_i\|}\right) \left(\frac{a_i}{\|a_i\|}\right)^\top \tag{27}$$

they define S^k by

$$S^k = \{ i : \frac{\|a_i\|^2 \left|\phi_i(x^k) - \phi_i^{\tau_k}\right|}{[\|a_i\|^2 \left|\phi_i(x^k) - \phi_i^{\tau_k}\right| + \min\{\gamma, \left\|\nabla f(x^k)\right\|\}]} > \eta, \quad 1 \le i \le m \}, \tag{28}$$

where $\gamma > 0$ is a constant. Then they gave two modified replacement criteria:

– Modified replacement criterion 1: For $k = 1$ and $i \in \{1, \ldots, m\}$, set $\phi_i^1 = \phi_i(x^1)$. For $k > 1$, define ϕ_i^k as follows:

1) If $|S^k| \le l$, set

$$\phi_i^k = \begin{cases} \phi_i(x^k) & \text{if } i \in S^k, \\ \phi_i^{\tau_k} & \text{if } i \notin S^k. \end{cases}$$

2) Otherwise, set

$$\phi_i^k = \phi_i(x^k).$$

2) Modified replacement criterion 2: The same as modified replacement criterion 1 except choose

an extra integer parameter $p > 0$ and if $k \le p$, S^k is the same; otherwise, define

$$S^k = \{i : \left|\phi_i(x^k) - \phi_i^{\tau_k}\right| \ge \max_{k-p+1\le j\le k} \left|\phi_i(x^k) - \phi_i(x^{j-1})\right|, \quad 0 \le i \le m\} . \tag{29}$$

Consider the assumptions

[A1] For $i = 1, \ldots, m$, $U(\cdot)$ is twice continuously differentiable;

[A2] For any real constant $\overline{f}$, the level set $\{x : f(x) \le \overline{f}\}$ is compact;

[A3] For $i = 1, \ldots, m$, the function $\phi_i(x)$ is Lipschitz continuous.

Under these assumptions the authors proved that the two algorithms are globally convergent, and if x^k converges to x^* and $\nabla^2 f(x^*)$ is positive definite, then the convergence rate of the first algorithm is quadratic, while for the second algorithm, x^k is locally superlinearly convergent to x^* with R-order at least r_p, where r_p is the unique positive root of $t^{p+1} - t^p - 1 = 0$.

The authors' numerical experiments showed that the two algorithms outperform the trust region method which uses GW's criteria 1 and 2.

An Inexact Newton Method Using Preconditioned Conjugate Gradient Techniques

More recently (1999), Deng, Wang and Zhang [6] developed Goldfarb and Wang's idea further and exploited the preconditioned conjugate gradient technique proposed in [5] and [4] to derive a new inexact Newton method. They showed that, when $n \ge 31$, the local behavior of this algorithm is superior to that of both the GW algorithms and the algorithm mentioned in the above section [3] from the efficiency point of view. Their algorithm model is given below.

Suppose problem (1) has a solution x^* and the following conditions are valid in a neighborhood of x^*:

[A1] For $i = 1, \ldots, m$, the second derivative $\phi_i(x)$ of $U_i(\cdot)$ defined by (5) is Lipschitz continuous with the constant L;

[A2] $\nabla^2 f(x^*) > 0$, i. e. the Hessian is symmetric positive definite.

The authors proved the following *local quadratic convergence theorem* about the Deng–Wang–Zhang algorithm.

Theorem 5 *Let Assumptions A1)–A2) hold. If $l \ge p$, then there are positive scalars δ and M such that if $\|x^0 - x^*\| \le \delta$, the Deng–Wang–Zhang Algorithm (Algorithm 1) is well-defined and, furthermore, the sequence $\{x^k\}$ generated by it satisfies*

$$\left\|x^{k+1} - x^*\right\| \le M \left\|x^k - x^*\right\|^2 .$$

In other words, the sequence $\{x^k\}$ converges to x^* with q-order at least 2.

The authors also studied the precisely quadratic convergence of the Deng–Wang–Zhang Algorithm, that is essential for the estimation of its computation cost. Consider the following additional assumption:

[A3] For any nonzero $h \in \mathbf{R}^n$, we have

$$\sum_{i=1}^{m} \phi_i'(x^*)(a_i^\top h)^2 a_i \ne 0 . \tag{37}$$

The authors proved

Theorem 6 *Consider the sequence $\{x^k\}$ generated by the Deng–Wang–Zhang Algorithm with $l > p$. Let Assumptions (A1)–(A3) hold. Then there is a positive scalar δ_1 and $M_1 > 0$ such that if $\|x^0 - x^*\| \le \delta_1$, then for any $k = 0, 1, \ldots$*

$$M_1 \left\|x^k - x^*\right\|^2 \le \left\|x^{k+1} - x^*\right\| \le M \left\|x^k - x^*\right\|^2 , \tag{38}$$

where M is defined in Theorem 5. In other words, the sequence $\{x^k\}$ converges to x^* with Q-order equal to 2.

In order to obtain an implementable algorithm, the authors specified the values of the parameters l, p and w in the Algorithm with the following purposes in mind:

1) retain the quadratic convergence;
2) keep the computation cost per iteration as little as possible.

Here the computation cost is only concerned with the arithmetic operations in solving the Newton equation in Step 4 and Steps 5–6. For simplicity, they only considered the number of multiplicative operations. For example, for Step 4, the arithmetic operations refer to the number Q_N of multiplications to compute the solution s^k to (36) by a direct Choleski factorization, which is

$$Q_N = Q_N(n) = \frac{n^3}{6} + \frac{3n^2}{2} - \frac{2n}{3} . \tag{39}$$

Numerical Methods for Unary Optimization, Table 1

$n =$	100	200	400	500	800	1000	2000
$w =$	0	0	0	0	0	0	0
$p =$	(2, 3)	(4, 5)	(4, 5)	(8, 9)	(8, 9)	(8, 9)	(16, 17)
$l =$	3	5	5	9	9	9	17
$\frac{\overline{Q}(n)}{Q_N(n)} \leq$	0.67	0.53	0.43	0.41	0.35	0.33	0.28

Similarly, the arithmetic operations in one conjugate gradient inner iteration in Step 6 refers to the number of multiplications, denoted by Q_{CG}, in calculating the right-hand side of (35), but with l there being replaced by 1:

$$Q_{CG} = Q_{CG}(n) = 2n^2 + 6n + 2\,. \tag{40}$$

Using the minimization of an upper bound of the average computation cost per iteration, the authors obtained the optimal parameter values of w, p and l, then established an implementable algorithm for problem (1) with $n \geq 9$ as follows.

The same as the Deng-Wang-Zhang algorithm, except that the parameters are specified as follows:

1 The parameter w: set $w = 0$.

2 The parameter p: when $19 \geq n \geq 9$, set $p \in (0, 1)$; when $30 \geq n \geq 20$, set $p \in (0, 2)$; when $n \geq 31$, set $p \in (2^{i^*}, 2^{i^*} + 1)$, where the integer i^* can be determined as follows: Let $N = [Q_N/Q_{CG}]$, divide interval $(0, N)$ by the points 2^i, $i = 1, \ldots,$ into subintervals $(0, 2], (2^1, 2^{1+1}], \ldots, (2^{j-1}, 2^j], (2^j, N]$, where $2^j + 1 \leq N \leq 2^{j+1}$, then compare the values

$$u(i) = \frac{Q_N}{1+i} + \frac{i(2^i + 1)Q_{CQ}}{1+i}$$

for $i = 1, \ldots, j$. The integer i^* is defined as the index corresponding to the smallest value u^* of $u(i)$: $u(i^*) = u^*$.

3 The parameter l: set $l = l(p)$, where $l(p)$ is defined by $p + 1 \geq l(p) > p$.

Then the authors proved the following theorem about the computation cost:

Theorem 7 *Let A1)–A3) hold. Then compared with the corresponding number $Q_N = Q_N(n)$ of Newton's method with Choleski factorization, the average arithmetic operations per iteration $\overline{Q} = \overline{Q}(n)$ of the improved Deng–Wang–Zhang algorithm has the following properties:*

1) When $30 \geq n \geq 9$, $\overline{Q}(n) \leq Q_N(n)$;

2) when $n \geq 31$, $\overline{Q}(n) < Q_N(n)$;

3) $\lim_{n\to\infty} \frac{\overline{Q}(n)}{Q_N(n)} \to 0$.

Corresponding to the theoretical result in Theorem 7, some numerical values are listed in Table 1. For example, for $n = 200$, we have $i^* = 2$, $p \in (4, 5)$, $l = 5$ and $u^* = u(2) = 0.53Q_N$, which means that the average cost per iteration of the improved Deng–Wang–Zhang algorithm is no more than 0.53 Q_N.

The authors also pointed out that the parameter selection problem in the Deng–Wang–Zhang algorithm is worth of further study from both theoretical and numerical points of view. Better performance than the one listed in Table 1 is expected due to a better parameter selection method.

See also

- ▶ Broyden Family of Methods and the BFGS Update
- ▶ Unconstrained Nonlinear Optimization: Newton–Cauchy Framework
- ▶ Unconstrained Optimization in Neural Network Training

References

1. Bandler W, Chen SH, Bienacki RM, Gao L, Madsen K, Yu H (1993) Robust circuit optimization using Huber functions. In: IEEE Trans Macrowave Theory and Techniques, pp 2279–2288

2. Byrd RH (1981/2) Algorithms for robust regression. In: Powell MJD (ed) Nonlinear Optimization. Acad. Press, New York, pp 79–84
3. Chen LH, Deng NY, Zhang JZ (1998) Modified partial-update Newton-type algorithms for unary optimization. J Optim Th Appl 97:385–406
4. Deng NY, Wang ZZ (1998) Can Newton's method be surpassed? Chinese Sci Bull 43:132–134
5. Deng NY, Wang ZZ (1998/9) Newton's method can be beaten. Quaderno DMSIA Univ. Bergamo, Bergamo
6. Deng NY, Wang ZZ, Zhang JZ (1999) An improved inexact Newton's method for unary optimization. Techn Report, Dept Math City Univ Hong Kong
7. Goldfarb D, Wang S (1993) Partial-update Newton method and unary factorable and partially separable optimization. SIAM Optim 3:382–397
8. Kelley CT (1995) Iterative methods for linear and nonlinear equations. SIAM, Philadelphia
9. McCormick GP, Sofer A (1991) Optimization with unary functions. Math Program 52:167–178
10. Zhang JZ, Deng NY, Chen LH (1999) On the replacement criteria for unary optimization. Dept Math City Univ Hong Kong, Techn Report MA-99-06
11. Zhang JZ, Xu CX (1999) A class of indefinite dogleg path methods for unconstrained optimization. SIAM J Optim 9(3):646–667