Derivative Free Algorithms for Nonsmooth and Global Optimization with Application in Cluster Analysis

Asef Nazari Ganjehlou

This thesis is submitted in total fulfilment of the requirement
for the degree of Doctor of Philosophy

School of Information Technology and Mathematical Sciences
University of Ballarat
PO Box 663
University Drive, Mount Helen
Ballarat, Victoria, Australia 3353

Submitted in February 2009

Abstract

Optimization theory deals with finding the local (global) minimizer of a function within a given set of points that are defined by constraints. When the functions describing objective and/or constraints are continuously differentiable, there are a mature theory and efficient methods to find a local optimal point. If one of those functions is not continuously differentiable (nonsmooth), finding descent directions and checking optimality conditions becomes very complicated. Despite the existence of an advanced theory, the design of efficient numerical methods in nonsmooth optimization is still a topic for research.

Over the last four decades, different methods have been developed to solve nonsmooth optimization problems. The subgradient method, bundle methods, the discrete gradient method and the more recent, the gradient sampling method, are among them. The subgradient method and the bundle methods are based on convexity assumption of the problem, but they do not provide satisfactory results for nonconvex nonsmooth problems; however, in real world, convexity assumption is a strict restriction. Despite the fact that there exist many efficient algorithms for solving global optimization problems, there are not special algorithms for solving nonsmooth global problems.

The other problem that originates from nonconvexity is global optimization. Most of the deterministic methods designed for optimization are able to find a local optimal point where we are interested in a global solution. Some deterministic, stochastic, heuristic and metaheuristic approaches exist to solve this problem, but they either get stuck in a suboptimal solution or take excessive CPU time to calculate a global solution. Hence, it is important to design efficient methods for solving nonsmooth and nonconvex problems.

Nonsmooth nonconvex problems appear in different contexts of practical applications. For example, minimization problems involving maximum or minimum functions

and nonlinear programming with penalty function are among them. Furthermore, many problems in Data Mining can be formulated as a nonsmooth and nonconvex problem. These problems have applications in machine learning, pattern recognition, image analysis, bioinformatics, etc. Other practical application examples would be the tax models in economical theory which consists of different pieces that are nonsmooth in their intersections and is a nonsmooth nonconvex problem or in telecommunication area, the problem of determining level-constrained hierarchical trees for network evaluation and multicast routing is also a nonsmooth nonconvex problem.

This thesis is devoted to the development of algorithms for solving nonsmooth nonconvex problems some of these algorithms are derivative free methods. More specifically, we develop an approximate subgradient, secant method for nonsmooth optimization and generalize it for global optimization and quasi secant method. Also, we propose an incremental algorithm for the clustering problem in Data Mining. The efficiency of the methods is established by comparing the numerical results with different existing methods based on some standard test functions. To show the practical application of the methods, they are especially tested on large-scale problems of clustering analysis.

Statement of Authorship

Except where explicit reference is made in the text of the thesis, this thesis contains no material published elsewhere or extracted in whole or in part from a thesis by which I have qualified for or been awarded another degree or diploma. No other person's work has been relied upon or used without due acknowledgment in the main text and bibliography of the thesis.

Signed: 

Dated: 08/10/2009

Asef Nazari Ganjehlou
Candidate

Acknowledgement

First of all, I would like to recognize Prof. Alex Rubinov for accepting me as one of his students and his great support during my journey from Iran to Australia. He was the epitome of kindness and humanity and at the same time a great Mathematician. I will always have his memory in my heart, and I will pray for him.

Words can not describe the appreciation I have toward my principal supervisor, Dr. Adil Bagirov. The time I have spent as a student under his supervision has definitely been an enjoyable part of my life. His care, support, knowledge, availability and guidance were unique. It has been an important honour for me to be his student at the University of Ballarat. Also, I would like to appreciate the effort of my associate supervisor, Dr. Musa Mammadov. I have learned a lot from him, and to him I express many thanks for his great guidance and support.

In addition to the support of my supervisors, this project would not have been completed without the financial support of the School of Information Technology and Mathematical Sciences (ITMS), specially under the management of Prof. Sid Morris. Sid, like a kind father and at the same time as a serious manager, allowed me to do my PhD study at the school and to have other opportunities to learn more and be highly qualified for my future career. Just saying thank you cannot describe what I mean and feel about Sid. Also, I should acknowledge all ITMS staff because of their kindness and support. It is impossible to mention all of them here, but I would like to mention some who have directly impacted on my exprience here. Prof. Mirka Miller, Prof. John Yearwood, David Yost, Philip Smith, Jason Giri, David Stratton, Sandra Herbert, Ewan Barker, Liping Ma and Alex Kruger.

I would also like to give my thanks to Diane Clingin for her great support in all my PhD studies.

Finally, my heartfelt appreciation goes to my parents and my wife, for their

courage, loving support and understanding. I am always thankful for what they have done for me.

Dedication

To my parents and my wife Sara

List of Publications

Journal papers

1. Bagirov A. and Nazari A., *An approximate subgradient algorithm for unconstrained nonsmooth nonconvex optimization*, Mathematical Methods of Operations Research, Volume 67, Number 2, April 2008 , pp. 187-206(20).

2. Bagirov A. and Nazari A. , *A Secant Method for Nonsmooth Optimization* (submitted).

3. Bagirov A. and Nazari A. , *A Quasi Secant Method for Global Optimization*, Optimization Methods & Software, 2009, pp. 1–16.

4. Bagirov A., Nazari A., Hakan Tor and Julian Ugon, *Truncated Codifferential Method for Nonsmooth Convex Optimization*, (submitted).

5. Bagirov A., Nazari A. and Hakan Tor, A *generalized subgradient method with piecewise linear subproblem*, (submitted).

Conference Papers

6. Nazari A. and Bagirov A. (2006). A Hybrid Discrete Gradient and Simulated Annealing Method for Global Optimization. The 5th Ballarat Workshop on Global and Nonsmooth Optimization: Theory, Methods and Applications.

7. Nazari A. and Bagirov A. (2007). An Approximate Subgradient Algorithm for Unconstrained Nnonsmooth Nonconvex Optimization. ASOR 19th National Conference of the Australian Society for Operations Research, RMIT, Melbourne.

8. Nazari A. and Bagirov A. A Secant Method for Nonsmooth Optimization, The 7th International Conference on Optimization: Techniques and Applications (ICOTA07), Japan 2007.

9. Nazari A., Ahmad S., Abdollahian M. and Zeephongsekul P., A Model to Estimate Proportion of Nonconformance for Multi-characteristics Product, 6th International Industrial Engineering Conference 2009, Sharif University, Iran

Table of Contents

List of Tables

List of Figures

Introduction

Optimization Theory is one of the advanced branches in Computational Mathematics. In the simplest description, optimization is searching for the best option among some (finite or infinite number of) possible choices. It has many applications in different areas such as Science, Engineering, Business, Management, Military, etc. Often the first step is to construct a model of the problem etc. of those areas in terms of objective functions and equality/inequality expressions; the model is called a mathematical programming problem. The next step is to design (or select among designed) algorithms based on the nature of the model to find the best (optimal) solution.

In classic theory of optimization, the differentiability of the involved functions is assumed. Nevertheless, there are many practical applications where the functions involved in the model are not continuously differentiable . For example, the clustering problem in Data Mining is a nonsmooth problem [27, 13], in the Telecommunication area, the problem of determining level-constrained hierarchical trees for network evaluation and multicast routing are also nonsmooth problems. In Engineering, complex contact situations involving several bodies with corners are described by nonsmooth problems. It has applications such as fragmentation, where angular fragments undergo complex collision sequences before they scatter [93], etc.

In addition to the nonsmoothness, many practical problems are nonconvex. Convexity is a very important remedy that makes the situation to some extent easy because in convex problems every local solution is a global solution as well. The first two problems, stated in the paragraph above, are nonsmooth and nonconvex. Some efficient methods for solving nonsmooth convex problems exist, but they are not always successful in nonconvex case. The complexity of nonconvex problems arises from combinatorial nature of multiple local solutions. Generally most of the algorithms are able to find a local solution whereas a global solution is needed.

We start by providing a general overview of the research problems to understand the basic ideas and motivations to conduct the research in designing optimization methods for nonsmooth nonconvex problems. The general form of the nonlinear optimization problems is

$$(P) \qquad \begin{cases} \text{minimize} & f(x) \\ \text{subject to} & x \in X \end{cases} \tag{0.0.1}$$

where $x \in \mathbb{R}^n$ is a vector of decision variables, and the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is considered to be locally Lipschitz on the feasible set $X \subset \mathbb{R}^n$. If the feasible set is $X = \mathbb{R}^n$, the optimization problem (0.0.1) is referred as an unconstrained optimization problem. The constrained optimization problem can be rewritten as:

$$\begin{cases} \text{minimize} & f(x) \\ \text{subject to} & h_i(x) = 0, \ \ i \in E, \\ & h_i(x) \leq 0, \ \ i \in I, \\ & x \in \mathbb{R}^n, \end{cases} \tag{0.0.2}$$

where E and I are the index sets of equality and inequality constraints, and $h_i(x)$, $i \in E \cup I$ are constraint functions. When both objective and constraint functions are linear functions, the problem (0.0.2) is called a linear programming problem. Otherwise, it is called a nonlinear programming problem. Based on different kinds of objective functions, constraints, feasible sets and decision variables, it could be considered different types of mathematical programming problems like integer programming, convex programming, etc.

The nature of optimization algorithms is iterative. Starting from an initial point $x^0 \in \mathbb{R}^n$, they construct a sequence $\{x^k\} \subset \mathbb{R}^n$ so that the limit point, $\lim_{k\to\infty} x^k = x^*$ if it is convergent, hopefully, is an optimal solution. If $f(x^{k+1}) < f(x^k)$ for all k, the method is called a descent method. The iterations are found by $x^{k+1} = x^k + \alpha^k g^k$ where α^k is the step size and g^k is a (descent) search direction. It is worthwhile to mention that different methods of optimization are categorized based on the way they find search direction and step length at each iteration.

Consider the unconstrained optimization problem:

$$\begin{cases} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathbb{R}^n, \end{cases} \tag{0.0.3}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is a locally Lipschitz; meaning there is no differentiability or convexity assumption about f. If f is continuously differentiable, the problem (0.0.3) is called a smooth problem. Otherwise, it is a nonsmooth problem.

If the function $f(x)$ is continuously differentiable, then $-\nabla f(x)$, where $\nabla f(x) \neq 0_n$, is always the (steepest) descent direction. Whenever the algorithm reaches a stationary point, it satisfies $\nabla f(x^*) = 0_n$. Hence, if the function is continuously differentiable, the gradient plays essential role in finding descent direction and formulating stopping criteria. The step size in this case could be found by an exact or inexact line search. However, $\nabla f(x)$ is not always numerically trustworthy and the accumulated errors of using approximations generally divert the results. Moreover, in nonsmooth problems, the gradient does not exist to be calculated at all points.

In the nonsmooth case, the direct use of gradient-based methods will generally fail [114]. The reason for this failure is that the gradient does not exist at all points. One possibility would be to approximate the function by smooth ones. In this kind of treatment, the approximation error could be high when the approximation is rough. The other possible tool here is using derivative free methods. These methods do not use any information about the gradient, and the search direction is decided just by comparing function values. However, most of them like Powell's method or Nelder and Mead's simplex method, are slow and their convergence can be proved only assuming that functions invloved are smooth [139].

Another approach to design algorithms for nonsmooth optimization is to use the concept of subdifferential which are generalizations of the gradient for nonsmooth optimization. The subdifferential could represent some local information about the function at a given point. The difficulty related to this approach is that the calculation of entire subdifferential generally is not possible whereas it is necessary to have the entire subdifferential to decide about descent directions and stationary points. Also in this approach, the opposite role of gradient in smooth problems, the negative arbitrary subgradient is not necessarily a descent direction. Furthermore, the stopping criteria is not easy to check.

In addition to the difficulty of nonsmoothness, in locally Lipschitz functions, there is the problem of multiple local and global optimal points. Based on the ability of optimization methods in finding local or global optimal points, the methods could be categorized local or global search methods. Finding the global minimizer is more difficult than the local one; this difficulty originates from the nonconvexity of objective function or constraints. For convex problems, any local solution is a global solution as well, and that is why convex problems are easier to deal with. However, there are many practical problems which have a nonconvex structure, which means they have many local minimizers, and we need a global optimal point.

In this thesis, we will focus on designing algorithms for the unconstrained nonsmooth nonconvex case of problem (0.0.3). The reason for restricting to unconstrained problems is that a constrained optimization problem could be easily transformed into an equivalent unconstrained problem by means of a penalty function technique. This is why in almost all nonsmooth optimization literature, the authors considered unconstrained optimization problems.

Over the last four decades, different methods have been developed to solve nonsmooth optimization problems. Subgradient Methods [118, 58, 130, 131], Bundle Methods and its variations [4, 119, 120, 80, 95, 96, 97, 112, 41, 43, 45, 46, 115, 47, 109, 128, 141, 143, 126, 135, 85], Gradient Sampling Method [31, 32, 34], methods based on smoothing techniques [16, 59] are among them.

Subgradient methods are quite simple, however they are not effective in solving many nonsmooth optimization problems. Algorithms based on smoothing techniques are applicable only to special classes of nonsmooth functions such as the maximum functions, max-min type functions. Bundle type algorithms are more general algorithms and they construct information about the subdifferential of the objective function using ideas known as bundling and aggregation. These algorithms are known to be very effective for nonsmooth convex problems. For nonconvex functions subgradient information is only locally meaningful and must be discounted when no longer relevant. Special rules have to be designed to identify those irrelevant subgradients. As a result bundle type algorithms are more complicated in the nonconvex case [34].

The gradient sampling method proposed in [34] does not require any rules for discounting irrelevant subgradients. It is a stochastic method. It relies on the fact

that locally Lipschitz functions are differentiable almost everywhere. At each iteration of this method a given number of gradients are evaluated from a given neighborhood of a current point.

From the application perspective, Data mining is one of the sources of challenging problems of both nonconvex and nonsmooth optimization problems. For example, the optimization approach toward unsupervised classification (clustering) leads to such problems. Clustering has broad application for instance in biology to classify plants and animals given their specifications, in marketing to find groups of customers with similar behavior, in insurance to identify groups of insurance policy with a high average claim cost or to identify frauds and etc. Also, the modeling of clustering problem in terms of nonsmooth nonconvex concepts has especial structures such as piecewise linear partially separable objective and/or constraint functions. It is important to design algorithms which could cope with large amounts of data and classify it into groups based on the similarities.

Despite existence of numerous methods to solve clustering problem such as the k-means algorithm, the global k-means algorithm, the branch and bound method, the dynamic programming approach, metaheuristics etc., they are slow, sensitive to starting points or applicable for small data sets. Additionally, the nonsmooth optimization approach to the clustering problem and related formulation suggests some important aspects like piecewise linearity which create the opportunity to develop methods which are fast, could deal with large data sets and are less sensitive to the starting points.

Research Objectives

According to discussion given above, the research objectives could be stated in three main goals:

1) Developing new derivative free algorithms for nonsmooth optimization

   As stated above, many practical applications exist which do not have the differentiability assumption. Efficient existing methods require a especial structure of the problems. Therefore, it is necessary to design methods which are applicable to wide range of nonsmooth problems according to the wide range of appearance of such problems in industry, engineering and etc.

2) Developing new algorithms for nonsmooth global optimization problems

   Because of great demand in different areas to find a global optimal point efficiently, it is important to develop algorithms which are able to locate the global minimizer even in nonsmooth problems.

3) Developing algorithms for cluster analysis

   As the the world is experiencing development in all branches, huge amounts of data and databases are created which makes it impossible to deal with them individually. Using data mining techniques, for example cluster analysis, it is easier to treat the entire data at once and make useful deductions and inferences about trends and patterns in it. In cluster analysis, we are dealing with the classification of data into different groups. It has broad application in machine learning, medical science (e.g. cancer and tumor detection), pattern recognition, image analysis and bioinformatics.

Our Contribution

The outcome of the research is to design some derivative free algorithms for nonsmooth local and global optimization problems. The designed algorithms have the capability to be applied in large-scale problems that have arisen in the clustering context. Their efficiency is established by numerical experiments which have been conducted on some standard academic and practical test functions. Also, using of these methods, we develop a new incremental algorithm for cluster analysis. Here is the list of designed algorithm with short explanation:

1. Approximate subgradient method

   The approximate subgradient method is a new algorithm for minimizing locally Lipschitz functions. This algorithm is as simple to implement as the subgradient method, and at the same time it is numerically efficient. Descent directions in this algorithm are computed by solving a system of linear inequalities. The convergence of the algorithm is proved for quasidifferentiable semismooth functions.

2. Secant method

   The secant method is a new algorithm to locally minimize nonsmooth, noncon-

vex functions. The notion of secants for nonsmooth functions is introduced. The secants are applied to find descent directions of locally Lipschitz functions. Then, the latter is used to design a minimization algorithm. It is proved that the iterates of this algorithm converges to Clarke stationary points. Also, we extended the method to a global search method and called it the global secant method.

3. Quasi secant method

   The notions of quasi secants are introduced and a minimization method is designed based on them. The quasi secants have the ability to exploit more global properties and information of the function.

4. New algorithm for the clustering

   Using the secant method, we developed a new incremental method to solve the clustering problem. In this method, at each step, the nonsmooth nonconvex problem of finding a starting points for cluster centers is solved using the secant method.

Outline of the Thesis

The remainder of this document is organized as follows. After review of background information in Chapter 1, Chapter 2 gives an explanation and literature review of some methods in nonsmooth and global optimization. In Chapter 3, we present the approximate subgradient method which is a local method for nonsmooth nonconvex optimization problems. Chapter 4, provides another method for nonsmooth nonconvex optimization. In this chapter the notion of secant is introduced and it is used to approximate the subdifferential. This chapter contains two version of the secant method. Chapter 5 will explain the quasi secant method. In Chapter 6, we present a short literature review of the clustering problem and propose an inceremental algorithm for solving it. Finally, we conclude the thesis by providing final remarks and some recommendations for future research.

Chapter 1

Background

In this chapter, we first provide some theoretical background regarding nonsmooth optimization including Clarke subdifferential and some classes of functions. The approximation of subgradients using the notion of discrete gradient is presented in Section 1.2.

1.1 Theoretical Background

In this section, after introducing some of the notions related to convex nonsmooth optimization, we present the generalization of those notions for nonconvex nonsmooth optimization. We briefly discuss the conditions for stationary points and descent direction. At the end, two important classes of nonsmooth functions, which have frequent appearance in application, are introduced.

Consider the following unconstrained optimization problem:

$$\begin{cases} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathbb{R}^n \end{cases} \tag{1.1.1}$$

A point x^* is called a local minimizer of f if there exists an $\varepsilon > 0$ such that

$$f(x^*) \leq f(x) \qquad \forall x \text{ with } \|x - x^*\| < \varepsilon.$$

In plain language, this means x^* is not worse than its neighbors. If the point is not worse than all other points, it is called global minimizer; it means,

$$f(x^*) \leq f(x) \qquad \forall x \in \mathbb{R}^n.$$

The nonsmooth optimization theory was initially developed for convex functions. The subdifferential of a convex function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $x \in Dom(f)$ is the set $\partial_C f(x)$ of vectors $\zeta \in \mathbb{R}^n$ so that

$$\partial_C f(x) = \{\zeta | \zeta \in \mathbb{R}^n, \ f(y) \geq f(x) + \langle \zeta, y - x \rangle \ \forall y \in \mathbb{R}^n\}.$$

Each member of this set is called a subgradient of f at x. However, if the function is not convex, this definition is not valid any more. It is constructive to mention that if the convex function f is continuously differentiable, then $\partial_C f(x) = \{\nabla f(x)\}$.

For convex functions, local linearizations based on gradients (or subgradients in nonsmooth case) are always produce underestimations. By taking a maximum over those linearizations defined at several points, one can get a lower piecewise linear underestimation for the original function. This property is not correct for nonconvex functions, and it is one of the reasons why many efficient algorithms for convex case are not successful in nonconvex case.

The Clarke Subdifferential

Locally Lipschitz functions are sufficiently general class for nonsmooth nonconvex functions. After the definition of locally Lipshitz functions, the generalization of subdifferential and related tools for such functions are presented.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is called a locally Lipschitz function if a positive constant L exists so that for all $x, y \in \mathbb{R}^n$

$$|f(x) - f(y)| \leq L\|x - y\|.$$

Let f be a locally Lipschitz function defined on $\mathbb{R}^n$. Clarke introduced the notion of subdifferential for such functions (see, for example, [64]). Since these functions are differentiable almost everywhere, the Clarke subdifferential is defined for them as follows:

$$\partial f(x) = \text{co} \left\{ v \in \mathbb{R}^n : \exists (x^k \in D(f), x^k \to x, k \to +\infty) : v = \lim_{k \to +\infty} \nabla f(x^k) \right\},$$

here $D(f)$ denotes the set where f is differentiable, co denotes the convex hull of a set. It is shown in [64] that the mapping $x \mapsto \partial f(x)$ is upper semicontinuous and bounded on bounded sets.

For locally Lipschitz functions, no classical directional derivatives 'need exist'. Therefore, the generalized directional derivative of f at x in the direction g is defined as

$$f^0(x,g) = \limsup_{y \to x, \alpha \to +0} \alpha^{-1}[f(y+\alpha g) - f(y)].$$

If the function f is locally Lipschitz , then the generalized directional derivative exists and

$$f^0(x,g) = \max\{\langle v, g\rangle : v \in \partial f(x)\}.$$

A function f is called a regular function on $\mathbb{R}^n$ if it is differentiable with respect to any direction $g \in \mathbb{R}^n$, and $f'(x,g) = f^0(x,g)$ for all $x, g \in \mathbb{R}^n$ where $f'(x,g)$ is the (one sided) directional derivative of the function f at the point x in the direction g:

$$f'(x,g) = \lim_{\alpha \to +0} \alpha^{-1}[f(x+\alpha g) - f(x)].$$

For a point x to be a minimizer point of the function f on $\mathbb{R}^n$, it is necessary that $0 \in \partial f(x)$. Also, a direction d is called a descent direction for f at $x \in \mathbb{R}^n$, if there $\varepsilon > 0$ exists such that for all $\lambda \in (0, \varepsilon]$

$$f(x+\lambda d) < f(x).$$

As it is seen from these definitions, to find a descent direction and to check a stationary point, one should have information about the whole subdifferential. However, it is not always easy to calculate the whole subdifferential. (For a study of some optimality conditions in nonsmooth optimization see [6]).

A function f is said to be upper semicontinuous at $x \in \mathbb{R}^n$ if for every sequence $\{x^k\}$ convergent to x the following holds

$$\limsup_{k \to \infty} f(x^k) \leq f(x),$$

and it is lower semicontinuous if

$$f(x) \leq \liminf_{k \to \infty} f(x^k).$$

An upper and lower semicontinuous function is continuous. A closed map is a function between two spaces which maps closed sets to closed sets.

Semismooth Functions

A function $f : \mathbb{R}^n \to \mathbb{R}^1$ is called semismooth at $x \in \mathbb{R}^n$ if it is locally Lipschitz at x, and for every $g \in \mathbb{R}^n$ the limit

$$\lim_{v \in \partial f(x+\alpha g'), g' \to g, \alpha \to +0} \langle v, g \rangle$$

exists. The class of semismooth functions contains convex, concave, max-type and min-type functions [142]. The semismooth function f is directionally differentiable, and

$$f'(x,g) = \lim_{v \in \partial f(x+\alpha g'), g' \to g, \alpha \to +0} \langle v, g \rangle.$$

Quasidifferentiable Functions

A function f is called quasidifferentiable at a point x if it is locally Lipschitz , directionally differentiable at this point and convex, compact sets $\underline{\partial} f(x)$ and $\overline{\partial} f(x)$ exist such that:

$$f'(x,g) = \max_{u \in \underline{\partial} f(x)} \langle u, g \rangle + \min_{v \in \overline{\partial} f(x)} \langle v, g \rangle.$$

The set $\underline{\partial} f(x)$ is called a subdifferential; the set $\overline{\partial} f(x)$ is called a superdifferential, and the pair $[\underline{\partial} f(x), \overline{\partial} f(x)]$ is called a quasidifferential of the function f at a point x [166].

1.2 Approximation of Subgradients

In this section, we consider an approach to approximate subdifferentials. This approach is based on the notion of a discrete gradient, which was introduced in [23] (see also [22, 9]). Here all propositions are given without proofs (for the proofs see [24] and [28]).

In this section, we will use the following notations:

$$S_\varepsilon(x) = \{y \in \mathbb{R}^n : \ \|x - y\| < \varepsilon\}, \quad \bar{S}_\varepsilon(x) = \{y \in \mathbb{R}^n : \ \|x - y\| \leq \varepsilon\}$$

$$S_\varepsilon = S_\varepsilon(0), \quad \bar{S}_\varepsilon = \bar{S}_\varepsilon(0), \quad S_1 = \{g \in \mathbb{R}^n : \|g\| = 1\},$$

$$\mathbb{R}^+ = \{t \in \mathbb{R}^1 : t > 0\}.$$

Let $G = \{e \in \mathbb{R}^n : e = (e_1, \ldots, e_n), |e_j| = 1,\ j = 1, \ldots, n\}$ be a set of all vertices of the unit hypercube in $\mathbb{R}^n$. We take $e \in G$ and consider the sequence of n vectors $e^j = e^j(\alpha),\ j = 1, \ldots, n$ with $\alpha \in (0, 1]$:

$$\begin{aligned} e^1 &= (\alpha e_1, 0, \ldots, 0), \\ e^2 &= (\alpha e_1, \alpha^2 e_2, 0, \ldots, 0), \\ \ldots &= \ldots\ldots\ldots \\ e^n &= (\alpha e_1, \alpha^2 e_2, \ldots, \alpha^n e_n). \end{aligned}$$

Assume $\lambda > 0$ be a given number. Let

$$P = \{z : z : \mathbb{R}^+ \to \mathbb{R}^+,\ \beta^{-1} z(\beta) \to +0,\ \beta \to +0\}$$

be the set of all univariate positive infinitesimal functions. We take any $g \in S_1$, $e \in G$ and compute $i \in \{1, \ldots, n\}$ such that $|g_i| = \max\{|g_k|,\ \ k = 1, \ldots, n\}$. For given $x \in \mathbb{R}^n$ and $z \in P$ consider a sequence of $n + 1$ points:

$$\begin{aligned} x^0 &= x + \lambda g, \\ x^1 &= x^0 + z(\lambda) e^1(\alpha), \\ \ldots &= \ldots \quad \ldots \\ x^n &= x^0 + z(\lambda) e^n(\alpha). \end{aligned}$$

Let f be a function defined on $\mathbb{R}^n$.

Definition 1.2.1 *The discrete gradient of the function f at the point $x \in \mathbb{R}^n$ is the vector $\Gamma^i(x, g, e, z, \lambda, \alpha) = (\Gamma_1^i, \ldots, \Gamma_n^i) \in \mathbb{R}^n, g \in S_1$ with the following coordinates:*

$$\Gamma_j^i = [z(\lambda)\alpha^j e_j)]^{-1} \left[f(x^j) - f(x^{j-1})\right],\ \ j = 1, \ldots, n,\ \ j \neq i,$$

$$\Gamma_i^i = (\lambda g_i)^{-1} \left[f(x + \lambda g) - f(x) - \lambda \sum_{j=1, j\neq i}^{n} \Gamma_j^i g_j\right].$$

It follows from Definition 1.2.1 that

$$f(x + \lambda g) - f(x) = \lambda \langle \Gamma^i(x, g, e, z, \lambda, \alpha), g\rangle \tag{1.2.1}$$

for all $g \in S_1,\ e \in G,\ z \in P,\ \lambda > 0,\ \alpha > 0$.

Remark 1.2.1 One can see that the discrete gradient is defined with respect to a given direction $g \in S_1$, and in order to compute it first we define a sequence of points $x^0, \ldots, x^n$, and compute the values of the function f at these points that is we compute $n+2$ values of this function including the point x. The i-th coordinate is defined so that it satisfies the equality (1.2.1) which can be considered as some version of the mean value theorem.

Proposition 1.2.1 *Let f be a locally Lipschitz function defined on $\mathbb{R}^n$ and $L > 0$ is its Lipschitz constant. Then for any $x \in \mathbb{R}^n$, $g \in S_1$, $e \in G$, $\lambda > 0$, $z \in P$, $\alpha > 0$*

$$\|\Gamma^i\| \leq \overline{C}, \quad \overline{C} = C(n)L, \quad C(n) = (n^2 + 2n^{3/2} - 2n^{1/2})^{1/2}.$$

For a given $\alpha > 0$ we define the following set:

$$B(x,\alpha) = \{v \in \mathbb{R}^n : \ \exists (g \in S_1, \ e \in G, \ z_k \in P, \ z_k \to +0, \ \lambda_k \to +0, \ k \to +\infty),$$

$$v = \lim_{k \to +\infty} \Gamma^i(x, g, e, z_k, \lambda_k, \alpha)\}. \tag{1.2.2}$$

From now on, we consider a function f defined on $\mathbb{R}^n$ and assume that this function is quasidifferentiable. We also assume that both sets $\underline{\partial} f(x)$ and $\overline{\partial} f(x)$ are polytopes at any $x \in \mathbb{R}^n$ that is at a point $x \in \mathbb{R}^n$ there exist sets

$$A = \{a^1, \ldots, a^m\}, \quad a^i \in \mathbb{R}^n, \quad i = 1, \ldots, m, \ m \geq 1,$$

$$B = \{b^1, \ldots, b^p\}, \quad b^j \in \mathbb{R}^n, \quad j = 1, \ldots, p, \ p \geq 1$$

such that

$$\underline{\partial} f(x) = \operatorname{co} A, \quad \overline{\partial} f(x) = \operatorname{co} B.$$

We denote by $\mathcal{F}$ the class of all semismooth, quasidifferentiable functions whose subdifferential and superdifferential are polytopes at any $x \in \mathbb{R}^n$. This class contains, for example, functions represented as a maximum, minimizer or max-min of a finite number of smooth functions.

Proposition 1.2.2 *Assume that $f \in \mathcal{F}$. Then at a given point x there exists $\alpha_0 > 0$ such that*

$$co\, B(x,\alpha) \subset \partial f(x), \ \forall \ \alpha \in (0, \alpha_0].$$

Remark 1.2.2 After fixing $g \in S_1$ and $e \in G$, the discrete gradient contains three parameters: $\lambda > 0$, $z \in P$ and $\alpha > 0$. $z \in P$ is used to exploit semismoothness of the function f and it can be chosen sufficiently small. Also, if $f \in \mathcal{F}$, then for any $\delta > 0$ there exists $\alpha_0 > 0$ such that $\alpha \in (0, \alpha_0]$ for all $y \in S_\delta(x)$. In the sequel we assume that $z \in P$ and $\alpha > 0$ are sufficiently small.

For a given $\lambda > 0$ consider the following set at a point $x \in \mathbb{R}^n$:

$$D_0(x, \lambda) = \operatorname{cl co} \left\{ v \in \mathbb{R}^n : \exists (g \in S_1, e \in G, z \in P) : \quad v = \Gamma^i(x, g, e, \lambda, z, \alpha) \right\}.$$

Proposition 1.2.1 implies that for a locally Lipschitz function, the set $D_0(x, \lambda)$ is compact and convex for any $x \in \mathbb{R}^n$.

Corollary 1.2.1 *Assume that $f \in \mathcal{F}$ and in the equality*

$$f(x + \lambda g) - f(x) = \lambda f'(x, g) + o(\lambda, g), \ g \in S_1,$$

$\lambda^{-1} o(\lambda, g) \to 0$ as $\lambda \to +0$ uniformly with respect to $g \in S_1$. Then for any $\varepsilon > 0$ there exists $\lambda_0 > 0$ such that $D_0(x, \lambda) \subset \partial f(x) + S_\varepsilon$ for all $\lambda \in (0, \lambda_0)$.

Corollary 1.2.1 shows that the set $D_0(x, \lambda)$ is an approximation to the subdifferential $\partial f(x)$ for sufficiently small $\lambda > 0$. However, it is true at a given point x, but not in some of its neighborhood. In order to get convergence results for a minimization algorithm based on discrete gradients we need some relationship between the sets $D_0(x, \lambda)$ and $\partial f(x)$ in some neighborhood of a given point x. We will consider functions satisfying the following assumption.

Assumption 1.2.1 *Let $x \in \mathbb{R}^n$ be a given point. For any $\varepsilon > 0$ there exist $\delta > 0$ and $\lambda_0 > 0$ such that*

$$D_0(y, \lambda) \subset \partial f(x + \bar{S}_\varepsilon) + S_\varepsilon \tag{1.2.3}$$

for all $y \in S_\delta(x)$ and $\lambda \in (0, \lambda_0)$. Here

$$\partial f(x + \bar{S}_\varepsilon) = \bigcup_{y \in \bar{S}_\varepsilon(x)} \partial f(y).$$

Remark 1.2.3 The set $D_0(x, \lambda)$, $\lambda > 0$ can be used to compute descent directions of the function f. However, the computation of this set is very time-consuming. In the Chapter 3, for example, we propose an algorithm for computation of descent directions which uses only a few discrete gradients from $D_0(x, \lambda)$.

Chapter 2

Literature Review

In this chapter, we provide information about some algorithms for nonsmooth, global, derivative free optimization. Besides their description, we present a short literature review about their appearance and expansion.

2.1 Nonsmooth Optimization Methods

This section provides information about some methods in nonsmooth optimization. The subgradient method, bundle method, discrete gradient method and gradient sampling are among them. The general description of the algorithms and some historical review is presented.

2.1.1 Subgradient Method

The subgradient method is a simple method for minimizing nonsmooth convex functions, and considered as the first method in this area. The basic idea behind the subgradient method is to generalize methods for smooth problems by using subgradient instead of gradients. It is very similar to the ordinary gradient method, which takes steps in the direction of antigradient toward local minimum for minimizing differentiable functions, but there are some considerable differences. The primary dissimilarity is the fact that the step lengths are fixed ahead of time. This means the subgradient method does not use any information computed during the time which the method is running, and it does not use line search. The other difference is that the subgradient method is not a descent method. It is possible (quite often) that the

function value increases during the process.

The subgradient method has some disabilities that make it weak in practice. First, unlike the antigradient, the negative direction of an arbitrary subgradient is not necessarily a descent direction. For this reason, the method is not considered a descent method, and it is unrealistic to use line search to find appropriate step sizes. The other weakness is related to lack of implementable stopping criteria. Only one arbitrary subgradient does not have enough information about optimality condition $0 \in \partial f(x)$. Because of these two facts, the step sizes are chosen ahead of time.

The subgradient method is much slower than Newton-like Methods. However, the range of problems that can be minimized by this method is far too wide. Furthermore, under some assumptions the sequence of points generated by this method is convergent to the optimal solution.

From a historical point of view, this method was developed by Shor [130] in the 60s and 70s, and from then because of its simplicity, it was developed and used extensively by researchers. There exist two major extensions of subgradient methods: descent-based methods (as stated above, similar to gradient-type methods and based on the function descent) [173, 29] and non-descent methods that are based on the distance decrease from optimal set [167, 121, 30, 80, 131, 58].

Recently, Chang [163] suggested the surrogate subgradient method and related framework for separable and coupled subproblems. Cavalcente and Yamada [148] used the adaptive projected subgradient method to find the adaptive filters that refine a given estimate of the optimal filter by suppressing a sequence of closed convex functions in Multiaccess Interface suppressions. Ruszczynski [5] considered a version of the subgradient method for convex nonsmooth optimization involving averaging. There are many other applications including [36, 146, 2, 168, 156, 157, 170]. Also, there is broad application of the subgradient method in road network design [156, 158].

The Method

Consider the following optimization problem:

$$\begin{cases} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathbb{R}^n \end{cases}$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a convex nonsmooth function. To minimize the function, the subgradient method uses the iterations

$$x^{k+1} = x^k - \alpha_k g^k,$$

where $x^k \in Dom(f)$ is the $k-$th iterate and $g^k \in \partial f(x^k)$ is any subgradient of f at x^k. The simplest description of the subgradient method is that at each iteration, it takes one step in the direction of anti-subgradient with predefined step size α_k.

It is possible that $-g^k$ is not a descent direction, or even if it is, the predefined step sizes could lead to $f(x^{k+1}) > f(x^k)$. In such cases, the common strategy is to keep track of the best solutions found so far. If we consider

$$f^k_{best} = \min\{f^{k-1}_{best}, f(x^k)\},$$

then $\{f^k_{best}\}^{\infty}_0$ will be a nonincreasing sequence.

Algorithm 2.1.1 Subgradient Algorithm.

Step 1. Select a starting point $x^1 \in \mathbb{R}^n$, and put $k = 1$.

Step 2. Calculate $f(x^k)$ and an arbitrary $g^k \in \partial f(x^k)$.

Step 3. Based on the step size rule compute $x^{k+1} = x^k - \alpha_k g^k$

Step 4. If some stopping criteria are satisfied, stop. Otherwise, go to Step 2, and put $k = k + 1$.

Step Size Rules

As stated above, the step size in the subgradient method is chosen in a different way compared to ordinary gradient methods. To understand the main idea behind the selection of step size rules, it is essential to consider the following facts. Let x^* be the optimal point, and x^k be the k-th estimate of x^*. Then $\|x^{k+1} - x^*\| < \|x^k - x^*\|$ whenever $0 < \alpha_k < 2[f(x^k) - f(x^*)]/\|g^k\|$. Furthermore, at each iteration k we have $\|x^0 - x^k\| \leq \sum^k_{j=0} \alpha_k$. Therefore, to guarantee global convergence, we should have $\alpha_k \downarrow 0$ as $k \rightarrow \infty$ and $\sum^{\infty}_{j=0} \alpha_k = \infty$ [114]. Here some of the most common step size rules used in the literature are listed.

1. Constant step size. Consider $\alpha_k = \alpha$ which α is a positive number independent of k.

2. Constant step length. Consider $\alpha_k = \gamma_k / \|g^k\|_2$, where $\gamma_k > 0$ and $\gamma_k = \|x^{k+1} - x^k\|$.

3. Square summable but not summable. The step size satisfies

$$\alpha_k \geq 0, \qquad \sum_{k=1}^{\infty} \alpha_k^2 < \infty, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

4. Nonsummable diminishing. The step size satisfies

$$\alpha_k \geq 0, \qquad \lim_{k \to \infty} \alpha_k = 0, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

5. Nonsummable diminishing step length. The step sizes are chosen as $\alpha_k = \gamma_k / \|g^k\|_2$, where

$$\gamma_k \geq 0, \qquad \lim_{k \to \infty} \gamma_k = 0, \qquad \sum_{k=1}^{\infty} \gamma_k = \infty.$$

There are some more choices for step length based on the different variants of the method. The common feature of all possible step lengths is that they are determined before running the algorithm. Therefore, it is to a large extent independent of current point, search direction and the data calculated during running time.

Convergence Results

Under some assumptions, it is possible to investigate the convergence properties of the subgradient method. If there is a point x^* which is the optimal point of the function ($f(x^*) = \min_{x \in \mathbb{R}^n} f(x)$ or $x^* \in X^*$), the subgradients are bounded ($\exists S > 0 \ \ s.t. \ \|g^k\| \leq S, \ \ \forall k$) and there exists an upper bound on the distance x^1 and the optimal set X^* ($\exists R \ \ s.t. \ \|x^1 - x^*\|_2 \leq R$), it is easy to prove that

$$f_{best}^k - f^* \leq \frac{R^2 + S^2 \sum_{i=1}^{k} \alpha_i^2}{2 \sum_{i=1}^{k} \alpha_i} \tag{2.1.1}$$

Using the equation (2.1.1) the convergence results of the subgradient method will be summarized as follows:

1. Constant step size α.

   The method is convergent to within $S^2\alpha/2$ of the optimal value, i.e. $|f_{best}^k - f^*| \leq S^2\alpha/2$.

2. Constant step length.

   The method is convergent to within $S\gamma/2$ of the optimal value, i.e. $|f^k_{best} - f^*| \leq S\gamma/2$.

3. Square summable but not summable, nonsummable diminishing and nonsummable diminishing step length.

   In these case the method is convergent to the optimal value, i.e. $f^k_{best} \to f^*$.

It would be interesting to think about the best possible choice for α_i so that the right hand side of equation (2.1.1) is minimized. Because that expression is a convex combination of α_is, it achieves the minimum when all α_k are equal. In other words, if we put $\alpha_i = R/S\sqrt{k} \;\; i = 1, \ldots, k$, we get the righthand side minimized and we have $|f^k_{best} - f^*| \leq RS/\sqrt{k}$.

In the case when the optimal value f^* is known, Polyak [30] suggests a step size that gets benefit from this knowledge. There are many situations, for example solving system of nonlinear equations using optimization, we know that the optimal value is zero if system is continuous. Furthermore, sometimes we can estimate the optimal value and use that knowledge in the calculation of the step length. The suggested step size in presence of optimal value f^* is

$$\alpha_k = \frac{f(x^k) - f^*}{\|g^k\|_2^2}.$$

It is not hard to prove that the sequence of $f(x^k)$ will converge to f^* [30]. The subgradient method with stated step size is called alternating projection.

Constrained Convex Optimization Problems

It is worthwhile to mention, there is an extension of the subgradient method to solve constrained convex optimization problem in terms of projection. Consider the following constrained minimization problem:

$$\begin{cases} \text{minimize} & f(x) \\ \text{subject to} & x \in X \end{cases}$$

The projected subgradient method to solve this problem consists of iterates generated by

$$x^{k+1} = P_X(x^k - \alpha_k g^k)$$

where P_X is the projection operator on set X of constraints. It is very simple to describe the algorithm. At each iteration, after calculation of $x^k - \alpha_k g^k$, it is projected into set X using the projection operator P_X. The convergence proof in the projected subgradient method is even easier, as by projection the iterates get closer and closer to all points of the set X and especially to the optimal point.

2.1.2 Bundle Methods

Bundle methods are the most efficient methods to solve a nonsmooth convex problem at the moment. They are based on the Clarke [64] subdifferential theory developed for Lipschitz continuous functions. The main idea behind the bundle method is to collect information about the subdifferential of the function using subgradients met in previous iterations into a bundle. In this way, rather than dealing with just one subgradient (like the subgradient method) the subdifferential is approximated by a bunch of subgradients.

The bundle method originated from Lemaréchal's ε-steepest descent method [113] which was a combination of the cutting plane method [101] and conjugate subgradient methods [112]. Because of the the difficulty of determining tolerance in ε-steepest descent method, the generalized cutting plane method was designed by Lemaréchal [42] and developed by Kiwiel [95]. In the later method a convex piecewise linear approximation of the objective function is considered by means of subgradients and a descent direction is found by solving a quadratic problem. Also, in [95], Kiwiel suggested two strategies, namely subgradeint selection and aggregation, to limit the number of stored subgradients.

In spite of different backgrounds for the methods suggested by Lemaréchal and Kiwiel, both methods solve a quadratic direction finding problem to find the search direction at each iteration. As stated about determining the tolerance for Lemaréchal's method, Kiwiel's method suffered from sensitivity to the scaling of the objective function and the uncertain number of line searches. All late versions of bundle methods are designed to overcome those difficulties.

Two more extensions of the bundle method are the proximal bundle method of Kiwiel [96] based on the proximal point algorithm [153] and bundle trust region developed by Schramm and Zowe [71] which is the combination of bundle method and

trust region method. These methods are very similar, and they are different just in technical implementations.

Developing different versions of bundle method is continuing. Among them are the infeasible bundle method of Sagastizabal and Solodov [49], proximal-projection bundle method and proximal bundle method with approximate subgradient of Kiwiel [98, 99], limited memory bundle method Haraala et al. [74, 75] and limited memory interior point bundle method of Karmitsa (Haraala after marriage) et al. [94], that is a combination of variable metric bundle method and interior point approach.

The Method

The bundle method is a descent method originating from the conjugate subgradient method [135]. At the k-th iteration of this method the search direction is determined by

$$d_k = -\sum_{i \in I_k} \lambda_i^{(k)} g_i, \quad g_i \in \partial f(x_k), \quad i \in I_k$$

where I_k is an index set and $\lambda_i^{(k)}$ are obtained by solving the subproblem

$$\min \| \sum_{i \in I_k} \lambda_i g_i \|_2^2$$

$$s.t. \sum_{i \in I_k} \lambda_i = 1 \quad \lambda_i \geq 0.$$

As stated before, the idea in the bundle method is to use a bunch of subgradients instead of individual one. Suppose a certain number of points are met by different steps of conjugate subgradient method. At each point, the function value and an arbitrary subgradient is calculated. Consider the following subproblem:

$$\min \| \sum_{i=1}^{k} \lambda_i g_i \| \tag{2.1.2}$$

$$s.t. \sum_{i \in I_k} \lambda_i = 1 \quad \lambda_i \geq 0$$

$$\sum_{i \in I_k} \lambda_i t_i^{(k)} \leq \overline{\varepsilon} \quad \lambda_i \geq 0$$

where $\overline{\varepsilon}$ is a given constant and weight factors $t_i^{(k)} \quad i = 1, \ldots, k$ should be found. The search direction in the bundle method is determined by

$$d_k = -\sum_{i \in I_k} \lambda_i^{(k)} g_i \tag{2.1.3}$$

Algorithm 2.1.2 Bundle Algorithm.

Step 1. Given initial point $x_1 \in \mathbb{R}^n$, compute $g_1 \in \partial f(x_1)$. Choose $0 < m_2 < m_1 < 0.5$, $0 < m_3 < 1$, $\varepsilon > 0$, $\eta > 0$, $k := 1$ and $t_1^{(1)} = 1$.

Step 2. Solve subproblem 2.1.2 to find $\lambda_i^{(k)}$ and compute d_k by 2.1.3. If $\|d_k\| \leq \eta$ stop.

Step 3. Compute $y_k = x_k + \alpha_k d_k$ such that

$$f(y_k) \leq f(x_k) - m_2 \alpha_k \|d_k\|_2^2 \tag{2.1.4}$$

holds or

$$f(y_k) - \alpha_k g_{k+1}^T d_k \geq f(x_k) - \varepsilon,$$

where $g_{k+1} \in \partial f(y_k)$. If (2.1.4) does not hold, then go to step 5.

Step 4.

$$x_{k+1} := y_k,$$

$$t_{k+1}^{(k+1)} = 1,$$

$$t_j^{(k+1)} = t_j^k + f(x_{k+1}) - f(x_k) - \alpha_k g_j^T d_k, \quad j = 1, \ldots, k.$$

Set $k := k + 1$, go to step 2.

Step 5.

$$x_{k+1} := x_k,$$

$$t_j^{(k+1)} = t_j^{(k)}, \quad (j = 1, \ldots, k),$$

$$t_{k+1}^{(k+1)} = f(x_k) - f(y_k) + \alpha_k g_{k+1}^T d_k.$$

Set $k := k + 1$, go to step 2. [169]

The convergence of the Bundle method was proven by Lemaréchal [44] as the following theorem:

Theorem 2.1.1 *Let $f(x)$ be convex and $\|\partial f(x)\|$ be bounded on some open set containing the set $\{x | f(x) < f(x_1)\}$. Algorithm 2.1.2 will terminate in finitely many iterations, i.e., there exists $k \in \mathbb{N}$ such that $f(x_k) \leq f^* + \varepsilon$.*

2.1.3 Discrete Gradient Method (DGM)

The notion of discrete gradient was introduced in [23] (for more details see [22, 9]). Its definition was given in the Section 1.2 in this thesis. The Discreet Gradient Method is based on Discrete gradient and trying to approximate subgradients and subdifferential.

Consider the following unconstrained minimization problem:

$$\begin{cases} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathbb{R}^n \end{cases}$$

where the function f is assumed to be a semismooth quasidifferentiable function. An important step in the discrete gradient method is the computation of a descent direction of the objective function f. Therefore, first, an algorithm is described for the computation of this direction.

Let $z \in P, \lambda > 0, \alpha \in (0, 1]$, the number $c \in (0, 1)$ and a tolerance $\delta > 0$ be given.

Algorithm 2.1.3 An algorithm for the computation of the descent direction.

Step 1. Choose any $g^1 \in S_1, e \in G$, compute $i = \text{argmax}\{|g_j|,\ j = 1, \ldots, n\}$ and a discrete gradient $v^1 = \Gamma^i(x, g^1, e, z, \lambda, \alpha)$. Set $\overline{D}_1(x) = \{v^1\}$ and $k = 1$.

Step 2. Calculate the vector $\|w^k\|^2 = \min\{\|w\|^2 : w \in \overline{D}_k(x)\}$. If

$$\|w^k\| \leq \delta, \tag{2.1.5}$$

then stop. Otherwise go to Step 3.

Step 3. Calculate the search direction by $g^{k+1} = -\|w^k\|^{-1} w^k$.

Step 4. If

$$f(x + \lambda g^{k+1}) - f(x) \leq -c\lambda\|w^k\|, \tag{2.1.6}$$

then stop. Otherwise go to Step 5.

Step 5. Compute $i = \text{argmax}\{|g_j^{k+1}| :\ j = 1, \ldots, n\}$ and a discrete gradient

$$v^{k+1} = \Gamma^i(x, g^{k+1}, e, z, \lambda, \alpha),$$

construct the set $\overline{D}_{k+1}(x) = \text{co}\{\overline{D}_k(x) \bigcup \{v^{k+1}\}\}$, set $k = k + 1$ and go to Step 2.

Some explanation seems necessary about Algorithm 2.1.3. In Step 1, the first discrete gradient is calculated with respect to an initial direction $g^1 \in \mathbb{R}^n$. The distance between the convex hull $\overline{D}_k(x)$ of all calculated discrete gradients and the origin is calculated in Step 2. This problem can be solved using Wolfe's algorithm ([136]). If this distance is less than the tolerance $\delta > 0$ then the point x is accepted as an approximate stationary point (Step 2), otherwise another search direction is calculated in Step 3. In Step 4, we check whether this direction is a descent direction. If this is true, the algorithm stops and the descent direction has been calculated, otherwise another discrete gradient ia calculated with respect to this direction in Step 5, and the set $\overline{D}_k(x)$ is updated. At each iteration k, the approximation $\overline{D}_k(x)$ of the subdifferential of the function f is improved. It can be proved that Algorithm 2.1.3 is terminating (see [22, 9]).

Now we have the necessary components to describe the Discrete Gradient Method. Let sequences $\delta_k > 0$, $z_k \in P$, $\lambda_k > 0$, $\delta_k \to +0$, $z_k \to +0$, $\lambda_k \to +0$, $k \to +\infty$, sufficiently small number $\alpha > 0$ and numbers $c_1 \in (0,1), c_2 \in (0, c_1]$ be given.

Algorithm 2.1.4 Discrete Gradient Method (DGM)

Step 1. Choose any starting point $x^0 \in \mathbb{R}^n$ and set $k = 0$.

Step 2. Set $s = 0$ and $x_s^k = x^k$.

Step 3. Apply Algorithm 2.1.3 for the computation of the descent direction at $x = x_s^k, \delta = \delta_k, z = z_k, \lambda = \lambda_k, c = c_1$. This algorithm terminates after a finite number of iterations $l > 0$. As a result we get the set $\overline{D}_l(x_s^k)$ and an element v_s^k such that

$$\|v_s^k\| = \min\{\|v\| : v \in \overline{D}_l(x_s^k)\}.$$

Furthermore either $\|v_s^k\| \leq \delta_k$ or for the search direction $g_s^k = -\|v_s^k\|^{-1} v_s^k$

$$f(x_s^k + \lambda_k g_s^k) - f(x_s^k) \leq -c_1 \lambda_k \|v_s^k\|. \tag{2.1.7}$$

Step 4. If

$$\|v_s^k\| \leq \delta_k \tag{2.1.8}$$

then set $x^{k+1} = x_s^k, k = k + 1$ and go to Step 2. Otherwise go to Step 5.

Step 5. Construct the following iteration $x_{s+1}^k = x_s^k + \sigma_s g_s^k$, where σ_s is defined as follows

$$\sigma_s = \text{argmax}\left\{\sigma \geq 0 : \ f(x_s^k + \sigma g_s^k) - f(x_s^k) \leq -c_2\sigma\|v_s^k\|\right\}.$$

Step 6. Set $s = s + 1$ and go to Step 3.

For the point $x^0 \in \mathbb{R}^n$ we consider the set $M(x^0) = \{x \in \mathbb{R}^n : f(x) \leq f(x^0)\}$.

Theorem 2.1.2 *Assume that the function f is semismooth and quasidifferentiable, its subdifferential and superdifferential are polytopes at any $x \in \mathbb{R}^n$ and the set $M(x^0)$ is bounded for starting points $x^0 \in \mathbb{R}^n$. Then every accumulation point of $\{x^k\}$ belongs to the set $X^0 = \{x \in \mathbb{R}^n : 0 \in \partial f(x)\}$.*

Remark 2.1.1 One of the important parameters in the definition of the discrete gradient is $\lambda > 0$. It follows from Proposition 1.2.2 that sufficiently small values of λ allow one to get approximations to subgradients. Despite the fact that large values of λ cannot be used to approximate subgradients, they can be used to compute descent directions. In the discrete gradient method we take any $\lambda_0 \in (0, 1)$, some $\beta \in (0, 1)$ and compute λ_k, $k \geq 1$ as follows:

$$\lambda_k = \beta^k \lambda_0, \quad k \geq 1.$$

Thus, in the discrete gradient method approximations to subgradients are only used at the final stages of the method. Therefore, it is not a subgradient-based method, and it is a derivative-free method.

2.1.4 Gradient Sampling Method

The idea of stochastic gradient goes back to the works of Shor [130] and Ermoliev [174]. Later in [31] Burke, Lewis and Overton used gradient sampling techniques to approximate the Clarke subdifferential for a locally lipschitz function. In 2005, they published a paper containing a gradient sampling method for nonsmooth nonconvex optimization [34] and they presented some convergence results. Later, there is diverse application of this method in different contexts such as [35, 33, 34, 19]. More recently, Kiwiel strengthened the convergence results and slightly modified the gradient sampling method [100].

The Method

By the Radamacher Theorem, a locally Lipschitz function is differentiable almost every where. Therefore, at a randomly selected point, with probability 1, the subgradient is unique, which means it is equal to the gradient. Considering this fact, in the gradient sampling method, at a given iterate, the gradient is calculated on a set of randomly generated nearby points. This process would provide local information of the function, and an ε-steepest descent direction is constructed by solving a quadratic programming problem, where ε is the sample radius. The next iterate is found by applying a line search toward the descent direction.

Suppose $L = \{x|f(x) \leq f(\tilde{x})\}$ is the bounded level set, D is the set that f is differentiable on and B is the closed unit ball.

Algorithm 2.1.5 Gradient sampling Method.

Step 0.(Initialization)
Let $x^0 \in L \cap D$, $\gamma, \beta \in (0,1)$, $\mu, \theta \in (0,1]$, $\varepsilon_0 > 0$, $\nu_0 \geq 0$, $k = 0$ and $m \in \{n+1, n+2, \ldots\}$.

Step 1. (Approximate the Clarke ε-subdifferential by gradient sampling)
Let $u^{k1}, \ldots, u^{km}$ be sampled independently and uniformly from B, and set

$$x^{k0} = x^k, \quad x^{kj} = x^k + \varepsilon_k u^{kj}, \quad j = 1, \ldots, m.$$

If for some $j = 1, \ldots, m$ the point $x^{kj} \notin D$, then STOP; otherwise, set

$$G_k = conv\{\nabla f(x^{k0}), \nabla f(x^{k1}), \ldots, \nabla f(x^{km})\},$$

and go to step 2.

Step 2.(Compute a search direction)
Let $g^k \in G_k$ solve the quadratic program $\min_{g \in G_k} \|g\|^2$, i.e.,

$$\|g^k\| = dist(0|G_k).$$

If $\nu_k = \|g^k\| = 0$, STOP. If $\|g^k\| \leq \nu_k$, set $t_k = 0$, $\nu_{k+1} = \theta\nu_k$, and $\varepsilon_{k+1} = \mu\varepsilon_k$, and go to the step 4; otherwise, set $\nu_{k+1} = \nu_k$, $\varepsilon_{k+1} = \varepsilon_k$, and $d^k = -g^k/\|g^k\|$ and go to step 3.

Step 3.(Compute a step length)

Set

$$t_k = \max_{s\in\{0,1,2,\ldots\}} \gamma^s$$

and

$$f(x^k + \gamma^s d^k) < f(x^k) - \beta\gamma^s \|g^k\|,$$

and go to the step 4.

Step 4.(Update)

If $x^k + t_k d^k \in D$, set $x^{k+1} = x^k + t_k d^k$, $k = k + 1$, and go to Step 1. If $x^k + t_k d^k \notin D$, let $\widehat{x}^k$ be any point in $x^k + \varepsilon_k B$ satisfying $\widehat{x}^k + t_k d^k \in D$ and

$$f(\widehat{x}^k + t_k d^k) < f(x^k) - \beta t_k \|g^k\|$$

(such an $\widehat{x}^k$ exists due to the continuity of f). Then set $x^{k+1} = \widehat{x}^k + t_k d^k$, $k = k + 1$, and go to Step 1.

The main convergence result follows. Let $\rho_\varepsilon(x) = dist(0|G_\varepsilon(x))$ be the distance between the origin and $G_\varepsilon(x)$.

Theorem 2.1.3 *(convergence for fixed sampling radius). If $\{x^k\}$ is a sequence generated by the GS algorithm with $\varepsilon_0 = \varepsilon$, $\nu_0 = 0$ and $\mu = 1$, then with probability 1 either the algorithm terminates finitely at some iteration k_0 with $\rho_\varepsilon(x^{k_0}) = 0$ or there is a subsequence $J \subset \mathbb{N}$ such that $\rho_\varepsilon(x^k) \to_j 0$ and every cluster point $\overline{x}$ of the subsequence $\{x^k\}_J$ satisfies $0 \in \overline{\partial}_\varepsilon f(\overline{x})$ (the ε subdifferential).*

2.2 Derivative Free Optimization

The exact definition of derivative free optimization is not clear. Considering this concept from different perspectives would lead to different categories of included methods. What the name implies is to not use any exact or approximate information of derivatives. Based on the literature dealing with optimization methods which do not use any information about the derivatives, there exist two main categories: derivative free methods and direct search methods. Roughly speaking, derivative free methods are those do not require derivative information and they use approximation of the objective function (not necessarily Taylor expansions). This means they rely on local

models of the objective function and constraints. However, direct search methods rely on the objective function values and decide about descent direction by comparing its value in current iterate with the best found so far. Also, this kind of method uses simple decrease rather than sufficient decrease, which is used in other derivative free methods.

Derivative Free Optimization (DFO) and direct search methods were designed especially to solve optimization problems whose objective function are computed by a "black box" or simulation (for example in modelling complex physical systems), so there is no function to calculate gradient. Each call of the black box is expensive and hence estimating the gradient using finite difference is more costly. More difficulties arise when the objective function value may be computed with noise, so the finite difference, even though expensive, may not be accurate.

2.2.1 Direct Search Method

Over the last decade different direct search algorithms have been developed. Among them generalized pattern search (GPS) algorithms are known to be most efficient. Apparently the first direct search methods are due to pattern search of Hooke and Jeeves [140] and simplex algorithm of Nelder and Mead [86]. Later this method was modified significantly by Torzcon and Lewis [164, 149, 151] and more recently by Audet and Dennis [37, 38].

Pattern search methods are a subclass of direct search algorithms. Direct search methods do not explicitly use exact information or approximation of derivative.

In 1997 Torczon [164] introduced the class of generalized pattern search (GPS) methods for solving unconstrained Nonlinear Programming (NLP) and showed that, this type of algorithm can cover coordinate search, evolutionary operation in factorial design [68], Hooke and Jeeves' algorithm [140] and the multidirectional search algorithm [88]. In that paper, Torczon showed that if all iterates lie in a compact set and the objective function f is continuously differentiable in a neighborhood of the level set $\{x \in R^n : f(x) \leq f(x_0)\}$, where x_0 is starting point, then a subsequence of iterates $\{x_k\}$ produced by GPS converges to a point $\overline{x}$ that satisfies $\nabla f(\overline{x}) = 0$ when step size parameter (mesh size) is decreasing to zero.

In [149], Lewis and Torczon generalized their algorithm by applying the theory

of positive linear dependence of Davis [40] to reduce the number of trial points at each iteration. They also introduced a heuristic, called rank ordering, in which, after evaluating f in directions forming a basis, they generate the new direction based on difference between the best and worst directions.

Lewis and Torczon have extended the GPS method to solve both bound [150] and linearly constrained problems [151]. They showed that by choosing appropriate search directions, if the objective function f is continuously differentiable in a neighborhood of the level set $\{x \in R^n : f(x) \leq f(x_0)\}$, the algorithm is guaranteed to produce a subsequence of iterates converging to a limit point $\overline{x}$ satisfying $\nabla f(\overline{x})^T(x - \overline{x}) \geq 0$ for any feasible x. Also, for nonlinear objective functions with nonlinear constraints, Lewis and Torczon [152] developed a derivative-free augmented Lagrangian version of GPS. For this algorithm they proved that under some assumptions, some subsequences of iterates converge to a KKT first order stationary point.

Audet and Dennis [37] presented an equivalent version of GPS for bound and linearly constraints, in which the strength of the results depends on local continuity and smoothness of the objective function. By using these concepts, their proofs are shorter and simpler. In [38] they implement a filter method into GPS to handle general nonlinear constraints.

A key point to Audet and Dennis, GPS algorithms is that they explicitly separate the search step from the poll step within the iteration, in which, any finite search strategy (including none) on the mesh may be employed without special effect on the convergence results. The flexibility and freedom in the search step enable the user to apply heuristics and knowledge of the problem to accelerate its convergence rate.

Recently, Audet, Dennis and Abramson in [123] and [39], have worked on mesh adaptive direct search (MADS). MADS is a version of GPS in which local exploration, called polling, is carried out in a dense set of directions in the space of optimization variables. They prove in [39] that a subsequence of MADS iterates converges to a limit point satisfying second-order necessary or sufficient optimality conditions.

Pattern Search Method

Consider

$$\begin{cases} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathbb{R}^n, \end{cases}$$

where $f : R^n \rightarrow \mathbb{R}$ is continuously differentiable. Direct search methods for this problem are those that neither compute nor explicitly approximate derivatives of f. A special subclass of direct search methods is pattern search methods. Examples of those methods include coordinate search with fixed stepsize, evolutionary operation using factorial design, and original pattern search from Hooke and Jeeves. All the methods perform search using a pattern of points independent of f.

The Pattern

The basic requirement for pattern search is a lattice T (or mesh) such that if $\{x_1, \ldots, x_N\}$ are the first N iterations of the method, then there exists a scale factor ϕ_N such that $\{x_1 - x_0, x_2 - x_1, \ldots, x_N - x_{N-1}\}$ all lie in the scaled lattice $\phi_N T$.

We are going to present an abstract version of pattern search methods. To define a pattern we need two matrices, a basic matrix and a generating matrix. The basic matrix could be any nonsingular matrix $B \in R^{n \times n}$ and the generating matrix is a matrix $C_k \in Z^{n \times p}$ where $p > 2n$. Consider the following partition of C_k:

$$C_k = [M_k \ -M_k \ L_k] = [\Gamma_k \ L_k]. \tag{2.2.1}$$

It is required that $M_k \in \mathbf{M} \subset Z^{n \times n}$, where $\mathbf{M}$ is a finite set of nonsingular matrices, and $L_k \in Z^{n \times (p-2n)}$ and contains at least one column, the column of zeros.

A pattern P_k is then defined by the columns of the matrix $P_k = BC_k$. Because B and C_k have rank n, the columns of P_k span R^n. Using 2.2.1 we can partition P_k as follows:

$$P_k = BC_k = [BM_k \ -BM_k \ BL_k] = [B\Gamma_k \ BL_k]. \tag{2.2.2}$$

Given stepsize $\Delta_k \in R$, $\Delta_k > 0$, we define a trial step s_k^i to be any vector of the form:

$$s_k^i = \Delta_k B c_k^i$$

where c_k^i is the i-th column of $C_k = [c_k^1 \ \ldots \ c_k^p]$. Note that Bc_k^i is the direction of the step and Δ_k is steplength parameter. At iteration k, the trail points would be $x_k^i = x_k + s_k^i$ where x_k is the current iterate.

The Exploratory Moves

The pattern search method does the search by using exploratory moves around the current point x_k before declaring the new iteration or updating parameters. These moves can be interpreted as sampling the function around x_k in a deterministic way to find $x_{k+1} = x_k + s_k$ with lower function value. From the convergence point of view we should consider some assumptions for exploratory moves:

1. $s_k \in \Delta_k P_k \equiv \Delta B C_k \equiv \Delta_k [B\Gamma_k \;\; BL_k]$
2. $\min\{f(x_k + y),\ y \in \Delta_k B\Gamma_k\} < f(x_k), \text{then}\ \ f(x_k + s_k) < f(x_k)$

Under these assumptions we can prove that

$$\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0,$$

even though we just need a simple decrease on f. To obtain $\lim_{k\to\infty} \|\nabla f(x_k)\| = 0$ we need a stronger assumption on exploratory moves and matrices [164].

Generalized Pattern Search Method (GPS)

The algorithms of GPS for unconstrained optimization is as follow:

Algorithm 2.2.1 The Generalized Pattern Search (GPS) Method.

Let $x_0 \in \mathbb{R}^n$ and $\Delta_k > 0$.

For k=0,1,... do

Step 1. compute $f(x_k)$

Step 2. Determine a step s_k using an exploratory move

Step 3. compute $\rho_k = f(x_k) - f(x_k + s_k)$.

Step 4. If $\rho_k > 0$ then $x_{k+1} = x_k + s_k$. Otherwise $x_{k+1} = x_k$.

Step 5. Update C_k and Δ_k.

To define a special pattern search method, one should define the basis matrix B, the generating matrix C_k, the exploratory moves for S_k and updating of C_k and Δ_k. The general scheme for updating steplength is keeping it fixed after a successful move and halving it if there is no improvement in iteration.

Coordinate Search Method

Coordinate search or compass search is the simplest case in GPS. Let k be the iteration index, x_k the current iterate and x_0 the initial guess. Let $D_\oplus$ denote the set of $2n$ coordinate directions, defined as the positive and negative unit coordinate vectors:

$$D_\oplus = \{e_1, e_2, \ldots, e_n, -e_1, -e_2, \ldots, -e_n\}.$$

Let Δ_k denote the steplength control parameter.

Algorithm 2.2.2 The Coordinate Search Method.

Let $x_0 \in R^n$ and $\Delta_0 > \Delta_{tol} > 0$.

For $k = 1, 2, \ldots$ do

Step 1. If there is $d_k \in D_\oplus$ such that $f(x_k + \Delta_k d_k) < f(x_k)$ then set $x_{k+1} = x_k + \Delta_k d_k$ and $\Delta_{k+1} = \Delta_k$.

Step 2. Otherwise, if $f(x_k + \Delta_k d_k) \geq f(x_k)$ for all $d \in D_\oplus$, set $x_{k+1} = x_k$ and $\Delta_{k+1} = \frac{1}{2}\Delta_k$.

Step 3. If $\Delta_k < \Delta_{tol}$, then terminate.

There are different ways to see if there is any $d_k \in D_\oplus$ satisfying a simple decrease $f(x_{k+1}) < f(x_k)$. One may consider all $2n$ trial points and choose the highest decrease (this can happen sequentially or using parallel computing). Another possibility is evaluating sequentially and stopping after the first simple decrease. Whenever the decrease direction is found, we call it a successful iteration.

Regardless of the process of evaluation of the objective function, the value of Δ_k is not reduced unless every trial point has been evaluated. In this case the iteration is called unsuccessful.

From a matrix point of view, we can say coordinate search is a special case of GPS. Consider $B = I$. The generating matrix $C_k = C$ contains all possible combinations of $\{-1, 0, 1\}$ in its columns. Thus C has $p = 3^n$ columns. We define $M = I$, and L consists of the remaining $3^n - 2n$ columns of C.

Hooke and Jeeves' Pattern Search Algorithm

In addition to the general notion of a direct search method, Hooke and Jeeves introduced an original pattern search method [140]. The pattern search of Hook and Jeeves could be considered as a variant of coordinate search that uses a pattern step to accelerate the progress of the algorithm by incorporating previous successful iterations. It is a deterministic local search method.

The Hook and Jeeves' pattern search is an opportunistic process. If the previous iteration is successful, then the current iteration begins by conducting coordinate search about a speculative point $x_k + (x_k - x_{k-1})$, rather than around the current iterate x_k (called pattern step).

If we have x_{k-1} and x_k, the algorithm takes a step $x_k - x_{k-1}$ from x_k. The function is evaluated at this trial point and accepted even if $f(x_k + (x_k - x_{k-1})) \geq f(x_k)$. Then the algorithm does a coordinate search around this temporary point. If the coordinate search about the trial point is successful, the point returned is the new iterate x_{k+1}. If not, the method reduces to coordinate search around x_k. In the unsuccessful case, the algorithm will reduce steplength until some stopping criteria are met.

Nelder-Mead Simplex Method

Much of the early work on direct search methods arose in the statistics community. The relation between unconstrained optimization and statistical design of experiments is explored in [165]. For instance, Box's paper on evolutionary operation (EVOP) [68] proposed the construction of a two-level factorial design around the current best point in order to capture some local second-order effects. The problem is that in this method the number of function evaluations will increase exponentially(2^n for each iteration).

In response to the costly Box's EVOP, Spendly, Hext and Himsworth [155] observed that a simplex ($n+1$ points in R^n) is the minimum number of points required by statistical design to capture first-order information. They proposed an optimization algorithm based on reflecting the vertex in the simplex with the highest (v_n) through the centroid of the opposite face ($c = \frac{1}{n}(v_0 + \cdots + v_{n-1})$) to get next trial point as $v_r = v_n + 2(c - v_n)$. Then we compare $f(v_r)$ with $f(v_{n-1})$ and in case of decrease we accept it.

The simplex algorithm of Nelder and Mead [86] is a version of this basic idea that

allows a simple line search of the form $v_n + \alpha(c - v_n)$ with a set of four possible choices for α. Typical choices are $\alpha \in \{\frac{1}{2}, \frac{3}{2}, 2, 3\}$. The line search has the effect of allowing the shape of the simplex to deform. Generally we can see the method as a sequence of simplices, but we can modify and change their shapes so that they adopt themselves to the local topology of the function [103].

In more detail, at each iteration of Nelder and Mead's algorithm, a current simplex is defined by $n + 1$ vertices. We have five kinds of operation on the simplices: reflection, expansion, outside contraction, inside contraction and shrinking. There are two possible outcomes: (1) a single new point replaces the worst vertex; or (2) if a shrink is performed, the new simplex contains the best point from the previous iteration and n new points closer to the best point than the previous ones.

For the Nelder and Mead simplex algorithm, only very week convergence results and only in one or two dimensions has been established. This method is widely used because it is simple and uses only function values. It is also applicable to nonsmooth problems, and where the function is not explicitly given.

2.2.2 Derivative Free Methods

As stated above, derivative free methods are a class of optimization methods for nonlinear programming problems which construct a local approximation of the objective function and use the derivative information of this approximation. this kind of methods have been extensively considered in last few decades. The approximation model would be constructed using interpolation or regression methods.

The idea of employing a quadratic model instead of the original objective function goes back to Winfield [55, 56]. From the early days, one important problem was the geometry of sampling of points for interpolation or regression. A similar approach to Winfield's models was Powell's linear multivariate interpolation [127]. The primary difference between the two methods was in updating geometric properties of sample points.

The first proof of convergence of the methods was presented by Conn, Scheinberg and Toint [50]. Recently Conn et al [51, 137] have extensively investigated the convergence properties and geometry of the sampling set.

Trust Region Method

The concept of trust region is introduced in Levenberg [92] and Marquardt [53] for solving nonlinear least square problems. The method constructs a model $m_k(p)$ which is an approximation of the objective function $f(x)$ in a neighborhood of the current iterate x_k. Because the model is quite trustable in this neighborhood, it is called a trust region. The model is minimized to get the minimum point $X_k + s$ as a candidate for the next iterate. Then the original objective function $(f(x_k + s))$ is evaluated at the trial point. If the new point is better than previous iterations $(f(x_k + s) < f(x_k))$, it is accepted as the new iteration $x_{k+1} = x_k + s$, and the radius of the trust region will be increased. Otherwise, the trial point is rejected and the radius will be decreased.

Algorithm 2.2.3 Trust Region Method.

Step 1. Choose initial guess x_0, initial radius δ_0 and termination tolerance ε. $k = 0$

Step 2. Construct the model m_k

Step 3. Minimize it within the trust region and consider the minimum at $x_k + s$.

Step 4. If $f(x_k + s) < f(x_k)$, increase δ_k, $x_{k+1} = x_k + s$ and update the model by step 2.

Step 5. Otherwise, reduce δ_k. if $\delta_k < \varepsilon$ exit, in other case, go to step 3.

2.3 Global Optimization

Global Optimization means finding the absolute minimum of a given nonconvex function. The primary difficulty in dealing with global optimization problems using conventional methods is that they easily get trapped at a local minimum. However, because of nonconvexity there are many such local minima, and we are interested in the global one. A good reference for global optimization is [83], and especially for Lipschitz functions is [84].

Algorithms for global optimization could be classified into deterministic such as branch and bound, and metaheuristic, including simulated annealing, genetic algorithm, tabu search etc.

Metaheuristic methods are generally inspired from the real world and they use artificial intelligence. They have the capability to explore the whole search space, and

they can escape from local minima. However, they are computationally expensive because of poor convergence rates. The reason for poor convergence is the random search nature of such algorithms.

2.3.1 Branch and Bound(B&B)

Branch and Bound algorithms are nonheuristic general methods for finding optimal solution especially for discrete (integer programming) and combinatorial optimization, introduced by Dakin [147] and developed by Gupta and Ravindra [70], Borchers and Mitchell [20] and Leyffer [154]. The method works reasonably efficiently and another good aspect is that it solves linear continuous problems as subproblems.

B&B searches for the optimal solution by investigating only part of search space rather than the whole space while it is understood that upon derived bounds on the objective function there is no optimal solution on the excluded parts of space. B&B is guaranteed to find the global minimizer with desired accuracy after a predictable number of steps.

The central idea behind B&B is to branch (partition the feasible region into many simpler subsets) and then attempt to find the best solution or compute a lower bound of the objective function on each subset. We can consider each subset as a subproblem of the original problem with a smaller feasible region. Each subproblem is solved if the best feasible solution in that subset is found or, it is discovered that the subset is empty or, based on the bounds, it is proven that there exists no optimal solution in that subset. If the subproblem couldn't be ended in one of these cases, then the subset will be partitioned into smaller subsets and the process will be repeated.

B&B uses both lower bounds and upper bounds of objective function. The lower bounds can be obtained by relaxing the problem. The upper bound will be considered the best solution value found in subset. If the lower bound in a subset is worse than the upper bound already obtained in another subproblem, the first subproblem does not contain a better solution and it does not need to be explored.

In continuous global optimization, the subproblems are constructed by partitioning the feasible region into smaller parts. The most commonly used way is dividing the domain of a variable into smaller intervals. In this case the complexity of branching may increase exponentially with the dimension.

2.3.2 Metaheuristics

In the theory of algorithm, there are two basic goals in designing algorithms. They are finding algorithms with good run time and good optimal solution quality. A heuristic algorithm is one that neglects one of them. For many practical problems, heuristic algorithms are the only way because there is no satisfactory good algorithm that ends in reasonable time or with accurate solution.

There is a class of general heuristic strategies called metaheuristics which often use a concept of random search. They can be applied to a wide range of real-world problems, even though they are designed for combinatorial hard problems. The term metaheuristics was first used by Glover [60] in 1986. Metaheuristic methods are virtually used to find a good solution within an acceptable time. Usually, there is no guarantee of obtaining the global solution for them. For this reason the result of metaheuristic methods for optimization problems is called sub-optimal solution.

The growing interaction between Computer Science and Optimization has yielded new practical solvers for global optimization problems. Metaheuristics mainly employ exploration and exploitation search procedures in order to diversify the search all over the search space and intensify the search in some promising areas. Therefore, metaheuristics cannot easily be trapped in a local minimum. However, they are computationally expensive because of their slow convergence due to using randomized search methods.

Metaheuristics can be classified into two categories: population-based methods and point-to-point methods. In a point-to-point scheme, at the end of each iteration there is a new point that is not worse than the previous one. In population-based methods, they start with a set (population) of feasible points and via iterations the methods improve the set. Genetic algorithms are regarded as examples of population-based methods, and simulated annealing and tabu search as examples of point-to-point methods.

In the rest, we briefly explain simulated annealing and tabu search as two representatives of point-to-point methods, and genetic algorithms as a type of population-based algorithms.

Tabu Search (TS)

Tabu search (TS) as a member of metaheuristic local search methods was proposed by Glover [60] in 1986. The method was first designed for hard Combinatorial Optimization problems and the results were satisfactory [61, 62, 63, 82, 48]. There are a few contributions of TS in continuous optimization. The most important features of TS are its use of adaptive memory and responsive exploration.

TS is the combination of local search and anti-cycling memory-based rules. It uses a local search procedure to move from a solution x_k to the next one x_{k+1} in the neighborhood of x_k, until some stopping criteria have been met. TS uses the concept of tabu list to restrict returning to recently visited solutions. The list will act as a filter for the search procedure. Using tabu list, the search will be able to explore regions of the search space that would be left unexplored. The side result of the prevention of revisiting explored regions is escaping from local optimality. The stopping criteria could be different depending on the structure of the problem. More or less, if there is no progress after seeing some new points, the algorithm would be terminated[17]. Although the method is widely used on many problems from different areas, there is not a robust convergence proof for it.

Algorithm 2.3.1 Tabu Search (TS) Algorithm.

Step 1. *(Initialization)* Choose an initial solution x_0. Set the tabu list to be empty, and $k = 0$.

Step 2. *(Generation)* Generate neighborhood moves (trial points) list $M(x_k) = \{y : y \in N(x_k)\}$, based on tabu restrictions (considering the points should not be met), where $N(x_k)$ is a neighborhood of x_k.

Step 3. *(Update)* Set x_{k+1} equal to the best trial solution in $M(x_k)$, and update the tabu list.

Step 4. *(Stopping Criteria)* If stopping criteria are satisfied, stop. Otherwise, go to Step 2.

Simulated Annealing (SA)

Simulated annealing (SA) is a random search method for the global optimization problem to find a suboptimal solution. It was independently introduced for combinatorial optimization problems by S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi in 1983 [102], and by V. Cerny in 1985 inspired from Metropolis et. al. [125], simulation on the behavior of atoms in special temperature. The application of SA for continuous problems was suggested by Pincus [122] and developed in [69, 73, 54]. An excellent overview of SA can be found in [129]. Despite proof of convergence of the method to the global minimum, it is computationally slow [108].

By analogy with the physical process of annealing in thermodynamics, each step of the SA algorithm replaces the current solution by a random "nearby" solution, chosen with a probability that depends on the difference between the corresponding function values and on a global parameter T (called the temperature). The dependence is such that the current solution changes almost randomly when T is large, but increasingly "downhill" as T goes to zero. The allowance for "uphill" moves saves the method from becoming stuck at local minima, the problem with greedier methods.

At each step, the SA heuristic considers some neighbour $\overline{X}$ of the current state X, and probabilistically decides between moving the system to state $\overline{X}$ or staying in state X. The probabilities are chosen so that the system ultimately tends to move to states of lower energy.

The main parts of a SA are choice of neighbours , transition between states and decreasing cooling schedule (the gradually reduction of T). The neighbours of each state are specified by the user, usually in an application-specific way. The probability of making the transition from the current state X to a candidate new state $\overline{X}$ is a function $A(e, e', T)$ of the energies $e = f(X)$ and $e' = f(\overline{X})$ of the two states, and of a varying parameter T (the temperature).

One essential requirement for the transition probability function A is that it must be nonzero when $e' > e$, meaning that the system may move to the new state even when it is worse (has a higher energy) than the current one (to escape from local minima). On the other hand, when T goes to zero, the probability $A(e, e', T)$ must tend to zero if $e' > e$, and to a positive value if $e' < e$. In particular, when T becomes 0, the procedure will reduce to a greedy algorithm. In order to apply the SA method

to a specific problem, one must specify the state space, the neighbor selection method, the probability transition function, and the annealing schedule.

Algorithm 2.3.2 Simulated Annealing.

Step 1. Choose an initial solution x_0 and cooling scheme.

While $(T > T_{min})$ do

Step 2. Randomly generate a neighbor $\overline{x}$ of x_k.

Step 3. If $e(\overline{x}) < e(x_k)$, set $x_{k+1} = \overline{x}$.

Step 4. Otherwise, set $x_{k+1} = x_k$ with probability $e^{-\frac{e(\overline{x})-e(x)}{T}}$ and decrease T.

Genetic Algorithm (GA)

Genetic algorithm is a method that mimics genetic evolution in nature and environmental adaptation of consecutive generations. The mentioned adaptation is applied through genetic inheritance from parents to children and survival of the fittest. The first appearance of GA is considered the Holland's paper and book [89, 90] in 1975. Nowadays GA is one of the best known types of metaheuristics, and has multipurpose usage in all branches of Science [162, 21, 175].

Formally GA starts with a population that is a finite set of feasible solutions which are called chromosomes. Chromosomes are gained through coding process that converts feasible solutions from original problem so that the algorithm can manipulate them. Before the coding process, we consider fixed number of attributes or variables for chromosomes (which are called genes). Also, there is a fitness function derived from the objective function and used for evaluations.

GAs consist of three main operations: selection, crossover and mutation. By selection, the algorithm selects chromosomes form current population in order to apply other operators. The selection is based on the value of fitness, and the chromosomes with higher fitness value have more chance to be selected. After selection, pairs of parents (chromosomes) are chosen upon probability to crossover and produce offsprings (new chromosomes). In order to increase diversity, the mutation is used to alter one or more genes of a randomly chosen chromosome [17].

Algorithm 2.3.3 Genetic Algorithm (GA).

Step 1. *(Initialization)* Generate an initial population P_0. Set the crossover and mutation probabilities $p_c \in (0,1)$ and $p_m \in (0,1)$, respectively. Set the generation counter $t = 0$.

Step 2. *(Selection)* Evaluate the fitness function F at all chromosomes in P_t. Select an intermediate population P_t' from the current population P_t.

Step 3. *(Crossover)* Associate a random number from (0, 1) with each chromosome in P_t' and add this chromosome to the parents pool S_t^P if the associated number is less than p_c. Repeat the following Steps 3.1 and 3.2 until all parents in S_t^P are crossovered:

Step 3.1 Choose two parents p_1 and p_2 from S_t^P . Mate p_1 and p_2 to reproduce children c_1 and c_2.

Step 3.2 Update the children pool set S_t^C through $S_t^C := S_t^C \cup \{c_1, c_2\}$ and update S_t^P through $S_t^P := S_t^P - \{p_1, p_2\}$.

Step 4. *(Mutation)* Associate a random number from (0, 1) with each gene in each chromosome in P_t', mutate this gene if the associated number is less than p_m, and add the mutated chromosome only to the children pool set S_t^C.

Step 5. *(Stopping Criteria)* If stopping conditions are satisfied, then terminate. Otherwise, select the next generation P_{t+1} from $P_t \cup S_t^C$. Set S_t^C to be empty, set $t := t+1$, and go to Step 2.

Hybrid Methods

One approach that has recently drawn attention is to combine global and local search methods to design more efficient global optimization algorithms (see [11, 18, 172]). Such an approach allows one to use powerful methods of local optimization for solving global optimization problems. Local search methods are fast and more precise than the global search methods. In contrast, global search methods are time-consuming. However, local search methods can be even trapped in stationary points that are not local minimizers. Therefore various combinations of local and global search methods can be considered to design algorithms of global optimization exploiting advantages of both local and global search methods.

Since DFO methods do not explicitly rely on local models of the objective and/or constraint functions their combination with global search methods may lead to better hybrid methods than any Newton-like methods. Results of numerical experiments

presented, for example, in [11] show that unlike Newton-like methods, some DFO methods can overcome stationary points which are not local minimizers and even sometimes shallow local minimizers. Thus the use of DFO methods allows one to reduce the number of stationary points which might be met in order to reach the global minimizer. Therefore, DFO methods have got more attention to be used in hybrid methods of global optimization.

Different strategies can be used for a combination of local and global search methods. These algorithms fall roughly into the following three classes:

1. The first class contains algorithms where the global search methods are applied to improve global search properties of local search methods (see, for example, [18, 107, 171]);

2. The second class contains algorithms where global search methods are used to escape from a stationary point which has been calculated by the local search algorithm and to generate a new starting point for a local search algorithm (see, for example, [11, 15, 172]);

3. The third class contains algorithms where a global method is used to generate a set of starting points for a local search method. Then the local search method is applied starting from each these points and the best is taken as an approximation to the global solution (see e.g. [117]).

Chapter 3

Approximate Subgradient Method

In this chapter, we propose approximate subgradient method (ASM) to solve the following problem:

$$\begin{cases} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathbb{R}^n, \end{cases} \tag{3.0.1}$$

where the objective function f is locally Lipschitz.

As stated in previous chapter, the subgradient method is a very simple algorithm for minimizing a nonsmooth convex function (see, [58] and [130] for details). This method uses step-lengths that are fixed ahead of time, and it does not contain a line search procedure. The subgradient method is not a descent method. For some problems it is extremely inefficient; However, it is simple and can be applied to a far wider variety problems. These facts motivate us to develop minimization algorithms based on the subgradient methods which are still quite simple, easy to implement and on the same time are more efficient than the subgradient algorithms and applicable to a wider class of minimization problems.

The approximate subgradient method is such an algorithm. This algorithm can be applied for minimizing nonconvex, nonsmooth functions. In this algorithm, as stated in Section 3.1, descent directions are computed by solving a system of linear inequalities. The latter problem is solved using the subgradient method, which is demonstrated in subsection 3.1.1. The algorithm is presented in Section 3.2 and its convergence is proved for quasidifferentiable semismooth functions. Armijo-type line search technique is used to find step-lengths. The numerical efficiency of the algorithm is demonstrated in Section 3.3 by numerical results. The chapter will be finished by

concluding Section 3.4.

3.1 Computation of Descent Directions

In this section, we propose an algorithm for the computation of descent directions.

Let $z \in P, \lambda > 0, \alpha \in (0, 1]$, numbers $c \in (0, 1)$ and $\delta > 0$ be given.

Algorithm 3.1.1 An algorithm for the computation of the descent direction.

Step 1. Choose any $g^1 \in S_1, e \in G$, compute $i = \text{argmax}\{|g_j|,\ j = 1, \ldots, n\}$ and a discrete gradient $v^1 = \Gamma^i(x, g^1, e, z, \lambda, \alpha)$. Set $\tilde{D}_1(x) = \{v^1\}$ and $k = 1$.

Step 2. Compute the direction $g \in \mathbb{R}^n$ as a solution to the following system:

$$\langle v^i, g\rangle + \delta \leq 0,\ i = 1, \ldots, k,\quad g \in S_1. \tag{3.1.1}$$

Step 3. If the system (3.1.1) is not solvable, then stop. Otherwise compute $\bar{g}$ as a solution to this system and go to Step 4.

Step 4. If

$$f(x + \lambda\bar{g}) - f(x) \leq -c\delta\lambda, \tag{3.1.2}$$

then stop. Otherwise set $g^{k+1} = \bar{g}$ and go to Step 5.

Step 5. Compute $i = \text{argmax}\{|g_j^{k+1}| :\ j = 1, \ldots, n\}$ and a discrete gradient

$$v^{k+1} = \Gamma^i(x, g^{k+1}, e, z, \lambda, \alpha),$$

construct the set $\tilde{D}_{k+1}(x) = \text{co}\{\tilde{D}_k(x) \bigcup \{v^{k+1}\}\}$, set $k = k + 1$ and go to Step 2.

Some explanation to Algorithm 3.1.1 is necessary. In Step 1, we compute the discrete gradient with respect to an initial direction $g^1 \in S_1$. In Step 2, a solution is found to the system of linear inequalities (3.1.1) with an additional condition $g \in S_1$ (below we will discuss algorithms for solving the system (3.1.1)). If the system is not solvable, then in Step 3, we accept the point $x \in \mathbb{R}^n$ as an approximate stationary point and the algorithm stops. If the system is solvable, then we compute a new search direction $\bar{g}$, and in Step 4, we check whether this direction is a descent direction. If it is, the algorithm stops and the descent direction has been computed, otherwise

we compute another discrete gradient with respect to this direction in Step 5 and update the set $\tilde{D}_k(x)$. At each iteration k, we improve the approximation of the subdifferential of the function f.

We will show that Algorithm 3.1.1 is terminating, that is, after a finite number of steps either we find that the point x is an approximate stationary point or we compute the descent direction. First, we will prove the following propositions.

Proposition 3.1.1 *If the system (3.1.1) is not solvable, then*

$$\min_{v \in \tilde{D}_k(x)} \|v\| < \delta. \tag{3.1.3}$$

Proof: Let $\tilde{v}$ be a solution to the following problem:

$$\min \ \frac{1}{2}\|v\|^2 \quad \text{subject to} \quad v \in \tilde{D}_k(x).$$

If $\tilde{v} = 0$ then the proof is straightforward. So we assume that $\tilde{v} \neq 0$. Then it follows from the necessary condition for a minimum that

$$\langle \tilde{v}, v - \tilde{v} \rangle \geq 0, \quad \forall v \in \tilde{D}_k(x)$$

which means

$$\|\tilde{v}\|^2 \leq \langle \tilde{v}, v \rangle, \quad \forall v \in \tilde{D}_k(x). \tag{3.1.4}$$

Since the system (3.1.1) is not solvable we get

$$\max_{i=1,\ldots,k} \langle v^i, g \rangle > -\delta, \quad \forall g \in S_1.$$

Consider $g = -\frac{\tilde{v}}{\|\tilde{v}\|}$. Then there exists $i \in \{1, \ldots, k\}$ such that

$$\langle \tilde{v}, v^i \rangle < \delta \|\tilde{v}\|.$$

Then the proof follows from (3.1.4). □

Remark 3.1.1 It follows from Proposition 3.1.1 that if in Step 2 of Algorithm 3.1.1 the system (3.1.1) is not solvable, then the point $x \in \mathbb{R}^n$ can be considered as an approximate solution.

Proposition 3.1.2 *If (3.1.3) is satisfied then the system (3.1.1) is not solvable.*

Proof: Assume the contrary that is (3.1.3) takes place but the system (3.1.1) has a solution. The latter means that there exists $g \in S_1$ such that

$$\langle v^i, g \rangle + \delta \leq 0, \ i = 1, \ldots, k.$$

Let

$$\|\tilde{v}\| = \min_{v \in \tilde{D}_k(x)} \|v\|.$$

Since $\tilde{v} \in \tilde{D}_k(x)$

$$\tilde{v} = \sum_{i \in I} \alpha_i v^i, \quad \sum_{i \in I} \alpha_i = 1, \quad \alpha_i \in (0, 1], \ i \in I \subset \{1, \ldots, k\}.$$

We get

$$\langle \tilde{v}, g \rangle \leq -\delta. \tag{3.1.5}$$

On the other hand

$$|\langle \tilde{v}, g \rangle| \leq \|\tilde{v}\| \|g\| = \|\tilde{v}\| < \delta$$

which contradicts (3.1.5). □

Proposition 3.1.3 *Let f be a locally Lipschitz function defined on $\mathbb{R}^n$. Then in Algorithm 3.1.1 one of the stopping criteria will be satisfied after a finite number of steps.*

Proof: If both conditions for the termination of the algorithm are not satisfied, then a new discrete gradient $v^{k+1} \notin \tilde{D}_k(x)$ exists. Indeed, in this case

$$f(x + \lambda g^{k+1}) - f(x) > -c\delta\lambda.$$

It follows from (1.2.1) that

$$\begin{aligned} f(x + \lambda g^{k+1}) - f(x) &= \lambda \langle \Gamma^i(x, g^{k+1}, e, z, \lambda, \alpha), g^{k+1} \rangle \\ &= \lambda \langle v^{k+1}, g^{k+1} \rangle \end{aligned}$$

and therefore

$$\langle v^{k+1}, g^{k+1} \rangle > -c\delta. \tag{3.1.6}$$

Assume that $v^{k+1} \in \tilde{D}_k(x)$. Since $g^{k+1} \in S_1$ is a solution to the system (3.1.1)

$$\langle v^i, g^{k+1} \rangle + \delta \leq 0, \ i = 1, \ldots, k$$

we have

$$\langle v^{k+1}, g^{k+1} \rangle \leq -\delta$$

which contradicts (3.1.6). The latter means that $v^{k+1} \notin \tilde{D}_k(x)$.

Now we will show that Algorithm 3.1.1 is terminating. Assume the contrary. Then Algorithm 3.1.1 generates an infinite sequence $\{g^k\}$ of directions $g^k \in S_1$. It follows from (3.1.6) that

$$\langle v^k, g^k \rangle > -c\delta, \quad \forall\, k = 2, 3, \ldots. \tag{3.1.7}$$

The latter means that for any $k \in \{2, 3, \ldots\}$ the direction g^k does not satisfy the system:

$$\langle v^t, g \rangle + \delta \leq 0, \; t = 1, \ldots, i, \; i \geq k.$$

It follows from Proposition 1.2.1 that $\|v\| \leq \overline{C}$ for all $v \in \tilde{D}_k(x)$. The direction g^{k+1} is a solution to the system

$$\langle v^i, g \rangle + \delta \leq 0, \; i = 1, \ldots, k.$$

However, directions g^j, $j = 2, \ldots, k$ are not solutions to this system. Then we get

$$\|g^{k+1} - g^j\| > \frac{(1-c)\delta}{\overline{C}}, \quad \forall\, j = 2, \ldots, k. \tag{3.1.8}$$

Indeed, if there exists $j \in \{2, \ldots, k\}$ such that

$$\|g^{k+1} - g^j\| \leq \frac{(1-c)\delta}{\overline{C}}$$

then we have

$$\left| \langle v^j, g^{k+1} \rangle - \langle v^j, g^j \rangle \right| \leq (1-c)\delta.$$

The latter means that

$$\langle v^j, g^j \rangle \leq \langle v^j, g^{k+1} \rangle + (1-c)\delta \leq -c\delta$$

which contradicts (3.1.7). The inequality (3.1.8) can be rewritten as follows:

$$\min_{j=2,\ldots,k} \|g^{k+1} - g^j\| > \frac{(1-c)\delta}{\overline{C}}.$$

Thus Algorithm 3.1.1 generates a sequence $\{g^k\}$ of directions $g^k \in S_1$ such that the distance between g^k and the set of all previous directions is bounded below. Since the set S_1 is bounded the number of such directions is finite. □

3.1.1 Solving the System (3.1.1)

Step 2 is an important step in Algorithm 3.1.1, where we solve the system (3.1.1) to find search directions. Different methods have been developed to solve a system of linear inequalities (see, for example, [7, 91]). However, these methods cannot be applied directly to solve the system (3.1.1) because of the presence of the additional condition $g \in S_1$. To solve this problem, we use the nonsmooth optimization approach. To solve the system (3.1.1) we reformulate it as the following optimization problem:

$$\text{minimize } \varphi_k(g) = \max\left\{0, \langle v^j, g\rangle + \delta,\ j = 1, \ldots, k\right\} \tag{3.1.9}$$

subject to

$$g \in B_1 = \{y \in \mathbb{R}^n :\ \|y\|^2 \leq 1\}. \tag{3.1.10}$$

It is clear that if the system (3.1.1) is solvable then there exists $g \in S_1$ such that $\varphi_k(g) = 0$. If it is not solvable then $\varphi_k(g) > 0$ for all $g \in S_1$.

The function φ_k is convex and piecewise linear and the problem (3.1.9)-(3.1.10) is convex programming problem. The problem of minimization of the function φ_k without the constraint (3.1.10) can be easily reduced to a linear programming problem. However, linear programming techniques cannot be applied directly to solve the problem (3.1.9)-(3.1.10) because of the nonlinear constraint (3.1.10).

The nonsmooth optimization approach has some advantages. First of all, since the discrete gradients are computed step by step we get the sequence of minimization problems of convex piecewise linear functions over the unit ball. The functions φ_k are built step by step and one can use the solution in step j, $j < k$ as a starting point in step $(j+1)$ which allows one to reduce the computational effort. We use the subgradient method to solve Problem (3.1.9)-(3.1.10). Since the minimum value or its lower bound is 0 we can use the following version of the subgradient method [30]:

$$g^{l+1} = Proj_{B_1}\left(g^l - \frac{\varphi_k(g^l)}{\|w^l\|^2} w^l\right),$$

where $Proj_{B_1}(\cdot)$ is a projection operator onto the set B_1, $w^l \in \partial\varphi_k(g^l)$ is a subgradient of the function φ_k at the point g^l. The subgradient w^l is computed as follows. First, we compute

$$R(g^l) = \left\{i \in \{1, \ldots, k\} :\ \langle v^i, g^l\rangle + \delta = \varphi_k(g^l)\right\}.$$

Then the subdifferential of the function φ_k at the point g^l is:

$$\partial\varphi_k(g^l) = \text{co}\left\{v^i : \ i \in R(g^l)\right\}.$$

Let $l_0 = |R(g^l)|$ be the cardinality of the set $R(g^l)$. Then

$$w^l = \frac{1}{l_0} \sum_{i \in R(g^l)} v^i.$$

The convergence results for this version of the subgradient method can be found, for example, in [30].

3.2 The Method and its Convergence

In this section we describe the approximate subgradient method. Let sequences $\delta_k > 0$, $z_k \in P$, $\lambda_k > 0$, $\delta_k \to +0$, $z_k \to +0$, $\lambda_k \to +0$, $k \to +\infty$, sufficiently small number $\alpha > 0$ and numbers $c_1 \in (0,1), c_2 \in (0, c_1]$ be given.

Algorithm 3.2.1 The approximate subgradient method

Step 1. Choose any starting point $x^0 \in \mathbb{R}^n$ and set $k = 0$.

Step 2. Set $s = 0$ and $x^{k_s} = x^k$.

Step 3. Apply Algorithm 3.1.1 for computation of the descent direction at $x = x^{k_s}, \delta = \delta_k, z = z_k, \lambda = \lambda_k, c = c_1$. This algorithm terminates after a finite number of iterations $l > 0$. As a result we get the system:

$$\langle v^i, g\rangle + \delta_k \leq 0, \quad i = 1, \ldots, l, \quad g \in S_1. \tag{3.2.1}$$

Step 4. If this system is not solvable set $x^{k+1} = x^{k_s}, k = k + 1$ and go to Step 2. Otherwise we get the direction $g^{k_s} \in S_1$ which is a solution to this system and

$$f(x^{k_s} + \lambda_k g^{k_s}) - f(x^{k_s}) \leq -c_1\lambda_k\delta_k. \tag{3.2.2}$$

Step 5. Construct the following iteration $x^{k_{s+1}} = x^{k_s} + \sigma_s g^{k_s}$, where σ_s is defined as follows

$$\sigma_s = \text{argmax}\left\{\sigma \geq 0 : \ f(x^{k_s} + \sigma g^{k_s}) - f(x^{k_s}) \leq -c_2\sigma\delta_k\right\}.$$

Step 6. Set $s = s + 1$ and go to Step 3.

Remark 3.2.1 One can see that Algorithm 3.2.1 consists of two loops: inner and outer loops. The inner loop consists of Steps 3, 4, 5, 6 and parameters δ_k, z_k, λ_k are fixed in this loop. The outer loop consists of Steps 2, 3 and 5 and the parameters δ_k, z_k, λ_k are updated in this loop. The algorithm each time returns to the outer loop if further improvement of the solution is not possible with the same values of the parameters, and they have to be updated to improve the approximation of subgradients.

Remark 3.2.2 Unlike the subgradient method, the proposed algorithm may use more than one approximate subgradient at each iteration. This makes it similar to bundle-type methods. However, at the same time, it does not use polyhedral underestimators of the objective function which makes it different from them. Moreover, in this method the subgradient method is applied to find descent directions.

We assume that the function f satisfies the following assumption:

Assumption 3.2.1 *Let $x \in \mathbb{R}^n$ be a given point. For any $\varepsilon > 0$ there exist $\delta > 0$ and $\lambda_0 > 0$ such that*

$$D_0(y, \lambda) \subset \partial f(x + \bar{S}_\varepsilon) + S_\varepsilon \tag{3.2.3}$$

for all $y \in S_\delta(x)$ and $\lambda \in (0, \lambda_0)$. Here

$$\partial f(x + \bar{S}_\varepsilon) = \bigcup_{y \in \bar{S}_\varepsilon(x)} \partial f(y).$$

For the point $x^0 \in \mathbb{R}^n$, we consider the set $M(x^0) = \{x \in \mathbb{R}^n : f(x) \leq f(x^0)\}$.

Theorem 3.2.1 *Assume that $f \in \mathcal{F}$, Assumption 3.2.1 is satisfied and the set $M(x^0)$ is bounded for starting points $x^0 \in \mathbb{R}^n$. Then every accumulation point of $\{x^k\}$ belongs to the set $X^0 = \{x \in \mathbb{R}^n : 0 \in \partial f(x)\}$.*

Proof: Since the function f is locally Lipschitz and the set $M(x^0)$ is bounded,

$$f_* = \inf \{f(x) : x \in \mathbb{R}^n\} > -\infty. \tag{3.2.4}$$

First we show that the inner loop stops after finite number of steps. In other words, for any $k > 0$ there exists $s = s_k \geq 0$ such that the system (3.2.1) becomes unsolvable

at x^{k_s}. Assume the contrary that is there exists $k > 0$ such that the inner loop is infinite for this k. This implies that the system (3.2.1) is solvable and the inequality (3.2.2) is satisfied for all $s \geq 0$. Since $c_2 \in (0, c_1]$, it follows from (3.2.2) that $\sigma_s \geq \lambda_k$. Then we can write

$$\begin{aligned} f(x^{k_{s+1}}) - f(x^{k_s}) &\leq -c_2\sigma_s\delta_k \\ &< -c_2\sigma_s\|v^{k_s}\| \\ &\leq -c_2\lambda_k\|v^{k_s}\|. \end{aligned}$$

If the system (3.2.1) is solvable for any s then it follows from Propositions 3.1.1 and 3.1.2 that $\|v^{k_s}\| \geq \delta_k$ and

$$f(x^{k_{s+1}}) - f(x^{k_s}) \leq -c_2\lambda_k\delta_k$$

or

$$f(x^{k_{s+1}}) \leq f(x^{k_0}) - (s+1)c_2\lambda_k\delta_k. \tag{3.2.5}$$

Since $\lambda_k > 0$ and $\delta_k > 0$ are fixed for any $k > 0$, it follows from (3.2.5) that $f(x^{k_s}) \to -\infty$ as $s \to +\infty$. This contradicts (3.2.4), that is the inner loop stops after a finite number of steps. This implies that for any $k > 0$ there exists $s = s_k \geq 0$ such that the system (3.2.1) is not solvable at x^{k_s}. It follows from Proposition 3.1.1 that

$$\|v^{k_s}\| = \min_{v \in \tilde{D}_{k_s}(x^{k_s})} \|v\| < \delta_k.$$

At the end of k-th inner loop, we get a point $x^{k+1} = x^{k_s}$, and

$$\min_{v \in \tilde{D}_{k+1}(x^{k+1})} \|v\| < \delta_k.$$

Since $\tilde{D}_{k+1}(x^{k+1}) \subset D_0(x^{k+1}, \lambda_k)$,

$$\min_{v \in D_0(x^{k+1}, \lambda_k)} \|v\| < \delta_k.$$

Replacing $k+1$ by k, we get

$$\min_{v \in D_0(x^k, \lambda_{k-1})} \|v\| < \delta_{k-1}. \tag{3.2.6}$$

Since $\{f(x^k)\}$ is a decreasing sequence, $x^k \in M(x^0)$ for all $k > 0$. Then the sequence $\{x^k\}$ is bounded and therefore it has at least one accumulation point. Assume x^* is

any accumulation point of the sequence $\{x^k\}$ and $x^{k_i} \to x^*$ as $i \to +\infty$. Then we have from (3.2.6)

$$\min_{v \in D_0(x^{k_i}, \lambda_{k_i - 1})} \|v\| \leq \delta_{k_i - 1}. \tag{3.2.7}$$

According to Assumption 3.2.1 at the point x^* for any $\varepsilon > 0$ there exist $\beta > 0$ and $\lambda_0 > 0$ such that

$$D_0(y, \lambda) \subset \partial f(x^* + \bar{S}_\varepsilon) + S_\varepsilon \tag{3.2.8}$$

for all $y \in S_\beta(x^*)$ and $\lambda \in (0, \lambda_0)$. Since the sequence $\{x^{k_i}\}$ converges to x^* for $\beta > 0$, there exists $i_0 > 0$ such that $x^{k_i} \in S_\beta(x^*)$ for all $i \geq i_0$. On the other hand since $\delta_k, \lambda_k \to 0$ as $k \to +\infty$ there exists $k_0 > 0$ such that $\delta_k < \varepsilon$ and $\lambda_k < \lambda_0$ for all $k > k_0$. Then there exists $i_1 \geq i_0$ such that $k_i \geq k_0 + 1$ for all $i \geq i_1$. Thus it follows from (3.2.7) and (3.2.8) that

$$\min_{v \in \partial f(x^* + \bar{S}_\varepsilon)} \|v\| \leq 2\varepsilon$$

Since $\varepsilon > 0$ is arbitrary and the mapping $\partial f(x)$ is upper semicontinuous $0 \in \partial f(x^*)$. □

Remark 3.2.3 Since Algorithm 3.1.1 can compute descent directions for any values of $\lambda > 0$, we take $\lambda_0 \in (0, 1)$, some $\beta \in (0, 1)$ and update λ_k, $k \geq 1$ as follows: $\lambda_k = \beta^k \lambda_0$, $k \geq 1$.

Remark 3.2.4 It follows from (3.2.2) and $c_2 \leq c_1$ that always $\sigma_s \geq \lambda_k$; therefore, $\lambda_k > 0$ is a lower bound for σ_s. This leads to the following rule for the computation of σ_s. We define a sequence:

$$\theta_m = m\lambda_k, \quad m, \geq 1$$

and σ_s is defined as the largest θ_m satisfying the inequality in Step 5.

3.3 Numerical Experiments

We compare the proposed algorithm with the subgradient method [130]. This method is as follows:

$$x^{k+1} = x^k - \alpha_k v^k, \tag{3.3.1}$$

where $v^k \in \partial f(x^k)$ is any subgradient, and $\alpha_k > 0$ is a step-length.

Convergence of the subgradient method was proved only for convex functions [130]. However, we apply this algorithm to nonconvex problems. Two different versions of the subgradient method is considered here:

1. SUB1: in this version the step-length α_k is to some extent constant. We take $\alpha_k = 0.005$ for the first 1000 iterations, $\alpha_k = 0.001$ for the next 4000 iterations and $\alpha_k = 0.0001$ for all other iterations. Based on numerical experience, such a choice of α_k leads to better results.

2. SUB2: in this version the step-length α_k is a decreasing sequence. We take $\alpha_k = 1/k$, however after each 25000 iterations we update it. Let p_k is the largest integer smaller than or equal to $k/25000$. Then,

$$\alpha_k = \frac{1}{k - 25000p_k}.$$

   Without updating of α_k the convergence of the subgradient method is extremely poor for nonconvex functions.

Since there is no stopping criterion in the subgradient method, we use the following two stopping criteria strategies. The number of function evaluations is restricted by 10^6, and also, the algorithm stops if it cannot decrease the value of objective function in 1000 successive iterations. We compute subgradients v^k in (3.3.1) using the scheme from Section 1.2.

Numerical experiments were conducted on a Pentium 4 PC with CPU 1.83 GHz and 1GB of RAM. We used 20 random starting points for each problem, and the starting points are the same for all three algorithms.

Results of numerical experiments are presented in Tables 3.1 and 3.2. In these tables ASM stands for the approximate subgradient method. In Table 3.1, we report the average objective function value over 20 runs of the algorithms as well as the numbers n_b and n_s (refer to A for definitions) for each problem. Table 3.2 presents the average number of the objective function evaluations and the average CPU time over 20 runs.

Results presented in Table 3.1 show that ASM outperforms other two algorithms in all problems except problems P5 and P7 where SUB2 algorithm produces better results. The latter with the application of updates of the step-lengths gives better

Table 3.1: Results of numerical experiments: obtained solutions

Prob.	ASM			SUB1			SUB2		
	f_{av}	n_b	n_s	f_{av}	n_b	n_s	f_{av}	n_b	n_s
P1	1.95222	20	20	1.95236	18	18	1.95223	20	20
P2	-43.99997	20	20	-43.94407	12	12	-43.99973	20	20
P3	3.70348	20	20	1106.25628	0	0	3.72973	8	8
P4	0.00492	18	19	2.72430	0	1	3.48299	6	6
P5	0.26208	4	4	17.49696	14	17	0.08947	10	11
P6	3.59972	20	20	3.60367	19	19	3.59974	20	20
P7	12.17732	11	12	-11.53027	5	5	-29.89826	7	12
P8	0.03891	3	13	0.04810	0	3	0.04761	0	8
P9	0.04273	0	17	1.59493	0	1	0.09259	0	3
P10	115.70646	20	20	231.41148	0	0	197.90843	0	0
P11	0.00318	0	20	0.79443	0	0	0.21997	0	0
P12	0.06862	0	12	0.10392	0	3	0.06876	0	8
P13	0.10373	2	19	4.85194	0	0	1.75332	0	1
P14	0.33172	2	17	12.30531	0	1	5.94889	0	2
P15	0.04008	17	20	0.59012	0	0	0.50676	0	0
P16	0.14117	0	20	0.52659	0	0	0.47843	0	0
P17	680.63056	20	20	1283.63922	0	0	738.18848	0	0
P18	24.30702	20	20	1083.73783	0	0	242.21861	0	0
P19	93.93033	8	20	6353.92360	0	0	3597.49593	0	0
P20	0.38112	0	20	749.33916	0	0	172.41626	0	0
P21	0.34827	0	14	1.52721	0	4	0.95114	0	6
P22	2.00000	20	20	2.00009	18	18	2.00000	20	20
P23	0.40000	12	19	2.12330	0	1	2.13230	0	1
P24	0.90000	11	19	54.55650	0	0	44.29527	0	1
P25	0.00000	20	20	7.09383	0	0	1.11665	4	4
P26	0.31303	0	20	62.26375	0	0	37.87321	0	0
P27	26.19072	0	20	42.35418	0	0	40.92745	0	0

results than SUB1 algorithm. Overall, ASM produced best known solutions in 49.6 % of cases (100 % for nonsmooth convex, 40.3 % for nonconvex regular and 52.5 % for nonconvex, nonregular functions). It gives in 81.9 % of cases (100 % for nonsmooth convex, 73.3 % for nonconvex regular and 98.3 % for nonconvex, nonregular functions) best solutions among all three algorithms.

SUB1 algorithm produced best known solutions in 15.9 % of cases (50 % for nonsmooth convex, 10.6 % for nonconvex regular and 15 % for nonconvex, nonregular functions). It gives in 19.1 % of cases (50 % for nonsmooth convex, 15 % for nonconvex regular and 15.8 % for nonconvex, nonregular functions) best solutions among all three

algorithms.

SUB2 algorithm produced best known solutions in 21.3 % of cases (80 % for nonsmooth convex, 11.9 % for nonconvex regular and 20 % for nonconvex, nonregular functions). It gives in 28.0 % of cases (80 % for nonsmooth convex, 21.4 % for nonconvex regular and 21.7 % for nonconvex, nonregular functions) best solutions among all three algorithms.

Comparing these results, one can see that the approximate subgradient algorithm is more efficient than two other subgradient algorithms. Our results show that both versions of the subgradient method are inefficient for solving nonsmooth optimization problems with moderately large number of variables (more than 10 variables).

One can see from results presented in Table 3.2 that ASM requires a significantly less number of the objective function evaluations. However, this is not the case for average CPU time. In Problems P2, P4, P7, P17, P18, P19, P24, P25 it requires more CPU time than other two algorithms. This means that in this problems ASM spends the most of CPU time on solving the subproblem to find descent directions. In the same time in the most of these problems it produces significantly better results than other algorithms.

3.4 Conclusions

In this chapter, we have presented an approximate subgradient algorithm for solving unconstrained nonsmooth convex and nonconvex optimization problems. The problem of computation of descent directions ,in this algorithm, is reduced to the minimization of a convex piecewise linear function. The latter problem is solved using the subgradient method. Unlike the subgradient method, the proposed algorithm may use more than one approximate subgradient at each iteration. This makes it similar to bundle-type algorithms. But on the same time, it does not use polyhedral underestimators of the objective function which makes it different from them. Moreover, in this method the subgradient method is applied to find descent directions. This makes the proposed method easier to implement. However, it is hard to say that this algorithm is as efficient for nonconvex problems as bundle-type algorithms for convex problems. To make the proposed algorithm more efficient, better algorithms

Table 3.2: Results of numerical experiments: the number of function evaluations

Prob.	No. of function eval.			CPU time		
	ASM	SUB1	SUB2	ASM	SUB1	SUB2
P1	365	180761	69601	0.03	0.03	0.01
P2	1600	954382	369713	0.17	0.12	0.04
P3	5419	1000000	878419	1.38	16.59	14.55
P4	4165	241993	31460	0.38	0.03	0.01
P5	780	606317	577381	0.04	0.11	0.11
P6	643	756612	283865	0.08	0.09	0.03
P7	2875	819993	643470	0.24	0.11	0.07
P8	692	992131	582893	0.05	2.94	1.72
P9	1691	905838	880996	0.13	0.29	0.27
P10	1736	1000000	1000000	0.23	3.14	3.11
P11	3815	957729	934791	0.36	0.82	0.80
P12	2324	751279	638933	0.22	2.78	2.36
P13	2788	996645	930583	0.42	2.13	1.95
P14	3231	1000000	870773	0.46	4.51	3.91
P15	5662	977514	946413	0.89	11.15	10.69
P16	11197	889630	725645	1.93	12.78	10.41
P17	2930	1000000	1000000	0.56	0.20	0.15
P18	11454	1000000	1000000	2.47	0.29	0.21
P19	34949	1000000	1000000	18.63	0.70	0.34
P20	50617	1000000	1000000	28.16	61.23	55.77
P21	6053	620464	558975	1.38	15.33	13.59
P22	339	82720	41532	0.03	0.02	0.01
P23	378	955895	527285	0.04	0.07	0.04
P24	1487	967631	885073	0.22	0.07	0.06
P25	7989	1000000	960136	1.71	0.17	0.15
P26	11934	1000000	1000000	2.56	3.75	3.96
P27	6671	54272	33675	1.77	1.26	1.59

for solving subproblems should be developed. This, as well as the comparison of the proposed algorithm with the bundle method, will be a topic for the future research.

Chapter 4

Secant Method

In this chapter, we propose a secant algorithm for solving the problem

$$\begin{cases} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathbb{R}^n. \end{cases} \tag{4.0.1}$$

We introduce the notion of an r-secant for locally Lipschitz functions and design a minimization algorithm based on it. The secant is an approximation to a subgradient. We use the bundling idea to collect some information from previous iterations which makes this algorithm similar to the bundle method. However, we don't approximate the objective function which makes this method different.

After this short introduction, the notion of an r-secant is introduced and studied in Section 4.1. Necessary conditions for a minimum using r-secants are given in Section 4.2. In Section 4.3 we describe an algorithm for the computation of a descent direction. A secant method is introduced and its convergence is studied in Section 4.4. Results of numerical experiments are given in Section 4.5. Section 4.8 concludes the chapter.

4.1 An r-Secant

In this section we introduce the notion of an r-secant for locally Lipschitz functions.

Let $S_1 = \{g \in \mathbb{R}^n : \|g\| = 1\}$ be the unit sphere in $\mathbb{R}^n$. For any $g \in S_1$ we define

$$g^{max} = \max\{|g_i|, \quad i = 1, \ldots, n\}.$$

It is clear that $|g^{max}| \geq n^{-1/2}$ for any $g \in S_1$.

Definition 4.1.1 *Let $g \in S_1$ and $g_j = g^{max}$ for some $j \in \{1, \ldots, n\}$. For $r > 0$ and $g \in S_1$ we take any subgradient $v \in \partial f(x + rg)$. A vector $s = s(x, g, r) \in \mathbb{R}^n$ where*

$$s = (s_1, \ldots, s_n) : s_i = v_i, \ i = 1, \ldots, n, \ i \neq j$$

and

$$s_j = \frac{f(x + rg) - f(x) - r \sum_{i=1, i \neq j}^{n} s_i g_i}{r g_j}$$

is called an r-secant of the function f at a point x in the direction g.

Remark 4.1.1 If $g_j = g^{max}$ for more than one $j \in \{1, \ldots, n\}$ then r-secants are defined for all of them.

Remark 4.1.2 One can see that the r-secant is defined with respect to a given direction and it is not unique if the subdifferential $\partial f(x + rg)$ is not the singleton set.

Remark 4.1.3 If $n = 1$ then an r-secant of the function f at a point $x \in \mathbb{R}^1$ is defined as follows:

$$s = \frac{f(x + rg) - f(x)}{rg}$$

where $g = 1$ or $g = -1$.

Proposition 4.1.1 *Let $x \in \mathbb{R}^n$ and $g \in S_1$. Then for a given $r > 0$*

$$f(x + rg) - f(x) = r\langle s(x, g, r), g \rangle \tag{4.1.1}$$

where $s(x, g, r)$ is an r-secant of the function f at a point x in the direction g.

Proof: Proof follows immediately from Definition 4.1.1, more exactly from the definition of the j-th coordinate s_j of the r-secant $s(x, r, g)$. □

Remark 4.1.4 The equation (4.1.1) can be considered as a version of the mean value theorem for r-secants.

For any bounded subset $X \subset \mathbb{R}^n$ there exists $M > 0$ such that (see [64] for more information)

$$\sup\{\|v\| : v \in \partial f(x), \ x \in X\} \leq M. \tag{4.1.2}$$

We define the following set

$$S_r f(x) = \{s \in \mathbb{R}^n : \exists g \in S_1 : \quad s = s(x, g, r)\}$$

which is the set of all possible r-secants of the function f at the point x.

Remark 4.1.5 In general, the set $S_r f(x)$ is not a singleton set even for continuously differentiable functions.

Proposition 4.1.2 *Let f be locally Lipschitz function defined on $\mathbb{R}^n$. Then for any bounded subset $X \subset \mathbb{R}^n$ there exists $M_0 > 0$ such that*

$$\sup\{\|v\| : v \in S_r f(x),\ x \in X\} \leq M_0. \tag{4.1.3}$$

Proof: Take any $x \in X$ and $s \in S_r f(x)$. It follows from the definition of the set $S_r f(x)$ that there exists $g \in S_1$ such that $s = s(x, g, r)$ where $s(x, g, r)$ is an r-secant in the direction g. Since $s_i(x, g, r) = v_i,\ i \in \{1, \ldots, n\},\ i \neq k, g_k = g^{max}$ for some $v \in \partial f(x + rg)$ it follows from (4.1.2) that

$$\sum_{i=1, i\neq k}^{n} s_i^2 \leq M^2.$$

Since the function f is locally Lipschitz there exists $L > 0$ such that

$$|f(x + rg) - f(x)| \leq Lr\|g\|.$$

Then we have

$$|s_k| \leq (L + (n-1)M)n^{1/2}$$

Denote

$$M_0 = \left(M^2 + (L + (n-1)M)^2 n\right)^{1/2}.$$

Then we have

$$\|s\| \leq M_0 \quad \forall s \in S_r f(x).$$

□

Proposition 4.1.3 *The set $S_r f(x)$ is closed.*

Proof: Take any sequence $\{s^k\},\ s^k \in S_r f(x)$. For each $s^k \in S_r f(x)$ there exists $g^k \in S_1$ such that $s^k = s^k(x, g^k, r)$. Without loss of generality we assume $g^k \to g$ as $k \to +\infty$. Since S_1 is a compact set $g \in S_1$. It is clear that $g^{k,max} \to g^{max}$. Let

$$J(g) = \{j \in \{1, \ldots, n\} : g_j = g^{max}\}$$

and

$$J(g^k) = \{j \in \{1, \ldots, n\} : g_j = g^{k,max}\}.$$

Then there exists $k_0 > 0$ such that $J(g^k) \subseteq J(g)$ for all $k \geq k_0$. We take any $j_0 \in J(g)$ such that $j_0 \in J(g^k)$ for sufficiently large k. Since $g^k \to g$ the upper semicontinuity of the set-valued mapping $x \mapsto \partial f(x)$ that if $v = \lim_{k \to +\infty} v^k$ then $v \in \partial f(x + rg)$. We define an r-secant using the subgradient $v \in \partial f(x + rg)$ and the fixed $j_0 \in J(g)$. Then the continuity of the function f implies that $s_{j_0}(x, g^k, r) \to s_{j_0}(x, g, r)$. Since $v^k \to v$ and $s_i(x, g^k, r) = v_i^k, \quad s_i(x, g, r) = v_i, \ i = 1, \ldots, n, \ i \neq j_0$ we get that $s(x, g^k, r) \to s(x, g, r)$ as $k \to +\infty$. □

Corollary 4.1.1 *The set $S_r f(x), \ r > 0$ is compact.*

Proof: The proof follows immediately from Propositions 4.1.2 and 4.1.3. □

Proposition 4.1.4 *The set-valued mapping $(x, r) \mapsto S_r f(x), r > 0$ is closed.*

Proof: Take any $x^k \in \mathbb{R}^n, \ r_k > 0, \ v^k \in S_{r_k} f(x^k)$ such that $x^k \to x, r_k \to r > 0$ and $v^k \to v$ as $k \to +\infty$. Then we have to show that $v \in S_r f(x)$. If $v^k \in S_{r_k} f(x^k)$ then there exists $g^k \in S_1$ and $w^k \in \partial f(x^k + r_k g^k)$ such that

$$v_i^k = w_i^k, \ i = 1, \ldots, n, \ i \neq j, \ j \in J(g^k).$$

without loss of generality we assume that $g^k \to g \in S_1$ and $w^k \to w$ as $k \to +\infty$. Since the set-valued mapping $x \mapsto \partial f(x)$ is closed $w \in \partial f(x + rg)$. There exists $k_0 > 0$ such that $J(g^k) \subseteq J(g)$ for all $k \geq k_0$. If we define an r-secant $s(x, g, r)$ using any $j \in J(g^k), \ k \geq k_0$, the subgradient $w \in \partial f(x + rg)$ then $s(x, g, r) = \lim_{k \to +\infty} v^k$ that is $v \in S_r f(x)$. □

Proposition 4.1.5 *The set-valued mapping $(x, r) \mapsto S_r f(x), r > 0$ is upper semi-continuous.*

Proof: The upper semi-continuity of the mapping $(x, r) \mapsto S_r f(x)$ follows from the fact that it is closed and its values are compact sets. □

We introduce the following two sets:

$$S_0 f(x) = \left\{ v \in \mathbb{R}^n : \exists (g \in S_1, \ r_k \to +0, \ k \to +\infty) : \ v = \lim_{k \to +\infty} s(x, g, r_k) \right\},$$

$$S_{0,g}f(x) = \left\{ v \in \mathbb{R}^n : \exists (r_k \to +0,\ k \to +\infty) :\ v = \lim_{k \to +\infty} s(x, g, r_k) \right\}.$$

It is clear that

$$S_0 f(x) = \bigcup_{g \in S_1} S_{0,g} f(x) \tag{4.1.4}$$

Proposition 4.1.6 *Let f be directionally differentiable at $x \in \mathbb{R}^n$. Then*

$$f'(x, g) = \langle v, g \rangle, \quad \forall v \in S_{0,g} f(x).$$

Proof: The proof follows from Proposition 4.1.1 and the definition of the set $S_{0,g}f(x)$. □

Corollary 4.1.2 *Let f be directionally differentiable at $x \in \mathbb{R}^n$. Then*

$$f'(x, g) \leq \max \{ \langle v, g \rangle, \quad v \in S_0 f(x) \}.$$

The proof follows from Proposition 4.1.1 and the definition of the set $S_0 f(x)$. □

Proposition 4.1.7 *Let f be semismooth at $x \in \mathbb{R}^n$. Then*

$$S_{0,g} f(x) \subset \partial f(x).$$

Proof: For a given $g \in S_1$ at a point x consider the following set

$$Q(x, g) = \left\{ v \in \mathbb{R}^n :\ \exists (r_k \to +0,\ k \to +\infty,\ v^k \in \partial f(x + r_k g)) :\ v = \lim_{k \to +\infty} v^k \right\}.$$

Since the function f is semismooth

$$f'(x, g) = \langle v, g \rangle \ \ \forall\, v \in Q(x, g).$$

It follows from the definition of the set $S_{0,g}f(x)$ that for any $s \in S_{0,g}f(x)$ there exists $v \in Q(x, g)$ such that $s_i = v_i,\ i = 1, \ldots, n,\ i \neq j,\ j \in J(g)$. On the other hand it follows from Proposition 4.1.6

$$f'(x, g) = \langle s, g \rangle = \langle v, g \rangle.$$

Then we get $s_j = v_j$ that is $s = v$ and $s \in Q(x, g) \subset \partial f(x)$. Since $s \in S_{0,g}f(x)$ is arbitrary we get the proof. □

Corollary 4.1.3 *Let f be semismooth at $x \in \mathbb{R}^n$. Then*

$$S_0 f(x) \subset \partial f(x).$$

Proof: The proof follows immediately from (4.1.4) and Proposition 4.1.7. □

At a point $x \in \mathbb{R}^n$ consider the following two sets:

$$S_r^c f(x) = \operatorname{co} S_r f(x), \ r > 0,$$

$$S_0^c f(x) = \operatorname{co} S_0 f(x).$$

From Corollary 4.1.3 we get the following

Corollary 4.1.4 *Assume that the function f is semismooth at a point $x \in \mathbb{R}^n$. Then*

$$S_0^c f(x) \subseteq \partial f(x).$$

Proposition 4.1.8 *Assume that the function f is regular and semismooth at a point $x \in \mathbb{R}^n$. Then*

$$\partial f(x) = S_0^c f(x).$$

Proof: It follows from Corollary 4.1.4 and semismoothness of the function f that

$$S_0^c f(x) \subseteq \partial f(x).$$

Therefore we have to show that

$$\partial f(x) \subseteq S_0^c f(x).$$

Since the function f is regular it is directionally differentiable and

$$f'(x, g) = \max\{\langle v, g\rangle : v \in \partial f(x)\}.$$

Then it follows from Corollary 4.1.3 that for any $g \in S_1$

$$\begin{aligned} f'(x, g) &\leq \max\{\langle v, g\rangle : \ v \in S_0^c f(x)\} \\ &\leq \max\{\langle v, g\rangle : \ v \in \partial f(x)\} \\ &= f'(x, g). \end{aligned}$$

Therefore for any $g \in S_1$

$$\max\{\langle v, g\rangle : \ v \in S_0^c f(x)\} = \max\{\langle v, g\rangle : \ v \in \partial f(x)\}.$$

Since both sets $S_0^c f(x)$ and $\partial f(x)$ are convex and compact we have that

$$\partial f(x) = S_0^c f(x).$$

□

4.2 Necessary Conditions and Descent Directions

In this section, we introduce r-stationary points; formulate necessary conditions for a minimum; and compute descent directions using the set $S_0^c f(x)$.

Proposition 4.2.1 *Assume that $f(x + rg) \geq f(x)$ for any $g \in S_1$. Then*

$$0 \in S_r^c f(x). \tag{4.2.1}$$

Proof: It follows from (4.1.1) that for any $g \in S_1$

$$f(x + rg) - f(x) = \langle s(x, g, r), g\rangle \geq 0.$$

Since $S_r f(x) \subset S_r^c f(x)$ then we have

$$\max\{\langle v, g\rangle : \ v \in S_r^c f(x)\} \geq 0 \tag{4.2.2}$$

for all $g \in S_1$. For any $g \in \mathbb{R}^n$, $g \neq 0$ there exist $g^0 \in S_1$ and $\lambda > 0$ such that $g = \lambda g^0$, therefore the inequality (4.2.2) is true for any $g \in \mathbb{R}^n$. Furthermore, the set $S_r^c f(x)$ is convex and compact and consequently we get the inclusion (4.2.1). □

Corollary 4.2.1 *Let $x \in \mathbb{R}^n$ be a local minimizer of the function f. Then there exists $r_0 > 0$ such that $0 \in S_r^c f(x)$ for all $r \in (0, r_0]$.*

Proof: If $x \in \mathbb{R}^n$ is a local minimizer then there exists $r_0 > 0$ such that $f(x + rg) \geq f(x)$ for any $g \in S_1$ and $r \in (0, r_0]$. Then the proof follows from Proposition 4.2.1. □

Proposition 4.2.2 *Let $x \in \mathbb{R}^n$ be a local minimizer of the function f and it is directionally differentiable at x. Then*

$$0 \in S_0^c f(x). \tag{4.2.3}$$

Proof: Since x is a local minimizer $f'(x,g) \geq 0$ for all $g \in \mathbb{R}^n$.Then it follows from Corollary 4.1.2 that

$$\max\{\langle v, g\rangle : \ v \in S_0^c f(x)\} \geq 0, \quad \forall g \in \mathbb{R}^n.$$

The set $S_0^c f(x)$ is compact and convex and therefore $0 \in S_0^c f(x)$. □

Definition 4.2.1 *A point $x \in \mathbb{R}^n$ is said to be an r-stationary point for a function f on $\mathbb{R}^n$ if $0 \in S_r^c f(x)$.*

Definition 4.2.2 *A point $x \in \mathbb{R}^n$ is said to be an (r,δ)-stationary point for a function f on $\mathbb{R}^n$ if $0 \in S_r^c f(x) + B_\delta$ where*

$$B_\delta = \{v \in \mathbb{R}^n : \quad \|v\| \leq \delta\}.$$

Assume that a point $x \in \mathbb{R}^n$ is not an r-stationary point of a function f on $\mathbb{R}^n$. This means that

$$0 \notin S_r^c f(x).$$

In this case one can compute a descent direction using the set $S_r^c f(x)$, which follows from the following proposition.

Proposition 4.2.3 *Let $x \in \mathbb{R}^n$ and for a given $r > 0$*

$$\min\{\|v\| : v \in S_r^c f(x)\} = \|v^0\| > 0.$$

Then for $g^0 = -\|v^0\|^{-1} v^0$

$$f(x + r g^0) - f(x) \leq -r\|v^0\|.$$

Proof: Since $S_r^c f(x)$ is compact and convex set we have

$$\max\left\{\langle v, g^0\rangle : v \in S_r^c f(x)\right\} = -\|v^0\|.$$

Then it follows from (4.1.1) that

$$\begin{aligned} f(x+rg^0) - f(x) &= \langle s(x, g^0, r), g^0\rangle \\ &\leq r \max\left\{\langle v, g^0\rangle : v \in S_r^c f(x)\right\} \\ &= -r\|v^0\|. \end{aligned}$$

□

4.3 Computation of a Descent Direction

Proposition 4.2.3 implies that for the computation of the descent direction we have to solve the following problem:

$$\begin{cases} \text{minimize} & \|v\|^2 \\ \text{subject to} & v \in S_r^c f(x). \end{cases} \tag{4.3.1}$$

Problem (4.3.1) is difficult to solve due to difficulties to compute the set $S_r^c f(x)$. In this section we propose an algorithm which uses only a few elements of $S_r^c f(x)$.
Let the numbers $r > 0$, $c \in (0,1)$ and a small enough number $\delta > 0$ be given.

Algorithm 4.3.1 An algorithm for the computation of the descent direction.

Step 1. Choose any $g^1 \in S_1$ and compute an r-secant $s^1 = s(x, g^1, r)$ in the direction g^1. Set $\overline{W}_1(x) = \{s^1\}$ and $k = 1$.

Step 2. Compute the vector w^k which $\|w^k\|^2 = \min\{\|w\|^2 : w \in \operatorname{co}\overline{W}_k(x)\}$. If

$$\|w^k\| \leq \delta, \tag{4.3.2}$$

then stop. Otherwise go to Step 3.

Step 3. Compute the search direction by $g^{k+1} = -\|w^k\|^{-1} w^k$.

Step 4. If

$$f(x + rg^{k+1}) - f(x) \leq -cr\|w^k\|, \tag{4.3.3}$$

then stop. Otherwise go to Step 5.

Step 5. Calculate an r-secant $s^{k+1} = s(x, g^{k+1}, r)$ with respect to the direction g^{k+1}, construct the set $\overline{W}_{k+1}(x) = \text{co}\left\{\overline{W}_k(x) \bigcup \{s^{k+1}\}\right\}$, set $k = k + 1$ and go to Step 2.

Remark 4.3.1 In Step 1 we compute the first secant. In Step 2 the least distance between the convex hull of all computed secants and the origin is computed. It is a quadratic programming problem and effective algorithms exist for its solution (see, for example, [3, 136]). If the distance is less than a given tolerance $\delta > 0$ then the point x is (r, δ)-stationary point, otherwise we compute a new search direction in Step 3. If it is descent the algorithms and the descent has been found (Step 4). If it is not descent direction then in Step 5 we compute a new r-secant in this direction. It improves the approximation of the set $S_r^c f(x)$.

In the next proposition we prove that Algorithm 4.3.1 is a terminating.

Proposition 4.3.1 *Let f be a locally Lipschitz function,*

$$\max\{\|v\| : v \in S_r^c f(x)\} \leq M < +\infty$$

and $c \in (0,1), \delta \in (0, M)$. Then Algorithm 4.3.1 terminates after m steps, where

$$m \leq 2\log_2(\delta/M)/\log_2 M_1 + 2, \quad M_1 = 1 - [(1-c)(2M)^{-1}\delta]^2.$$

Proof: First, we will show that if both stopping criteria 4.3.2 and 4.3.3 are not satisfied, then a new r-secant $s^{k+1} \notin \overline{W}_k(x)$ exists, that is in this case the algorithm allows one to improve the approximation of the set $S_r^c f(x)$. Indeed, in this case $\|w^k\| > \delta$ and

$$f(x + rg^{k+1}) - f(x) > -cr\|w^k\|.$$

On the other hand it follows from (4.1.1) that

$$f(x + rg^{k+1}) - f(x) = r\langle s^{k+1}, g^{k+1}\rangle.$$

Then from the definition of the direction g^{k+1} we get

$$\langle s^{k+1}, w^k\rangle < c\|w^k\|^2. \tag{4.3.4}$$

Since $w^k = \operatorname{argmin}\{\|w\|^2 : w \in \overline{W}_k(x)\}$, the necessary condition for a minimum implies that for any $w \in \overline{W}_k(x)$

$$\langle w^k, w - w^k \rangle \geq 0$$

or

$$\langle w^k, w \rangle \geq \|w^k\|^2.$$

The latter along with (4.3.4) means that $s^{k+1} \notin \overline{W}_k(x)$.

Now we will show that the algorithm is terminating. It is sufficient to get an upper estimation for the number m when

$$\|w^m\| \leq \delta. \tag{4.3.5}$$

It is clear that $\|w^{k+1}\|^2 \leq \|tv^{k+1} + (1-t)w^k\|^2$ for all $t \in [0,1]$ or

$$\|w^{k+1}\|^2 \leq \|w^k\|^2 + 2t\langle w^k, v^{k+1} - w^k \rangle + t^2\|v^{k+1} - w^k\|^2.$$

It follows from Proposition 4.1.2 that there exists $M > 0$ such that

$$\|s^{k+1} - w^k\| \leq 2M.$$

Hence taking into account the inequality (4.3.4), we have

$$\|w^{k+1}\|^2 \leq \|w^k\|^2 - 2t(1-c)\|w^k\|^2 + 4t^2M^2.$$

If $t = (1-c)(2M)^{-2}\|w^k\|^2 \in (0,1)$ we get

$$\|w^{k+1}\|^2 \leq \left\{1 - [(1-c)(2M)^{-1}\|w^k\|]^2\right\}\|w^k\|^2. \tag{4.3.6}$$

Take any $\delta \in (0, M)$. It follows from (4.3.6) and the condition $\|w^k\| > \delta, k = 1, \ldots, m_1 - 1$ that

$$\|w^{k+1}\|^2 \leq \{1 - [(1-c)(2M)^{-1}\delta]^2\}\|w^k\|^2.$$

Let $M_1 = 1 - [(1-c)(2M)^{-1}\delta]^2$. It is clear that $M_1 \in (0,1)$. Consequently,

$$\|w^m\|^2 \leq M_1\|w^{m-1}\|^2 \leq \ldots \leq M_1^{m-1}\|w^1\|^2 \leq M_1^{m-1}M^2.$$

If $M_1^{m-1}M^2 \leq \delta^2$, then the inequality (4.3.5) is satisfied and

$$m \leq 2\log_2(\delta/M)/\log_2 M_1 + 2.$$

□

4.4 The Method and its Convergence

In this section we describe the secant method for solving problem (4.0.1). Let $r > 0$, $\delta > 0$, $c_1 \in (0,1), c_2 \in (0, c_1]$ be given numbers.

Algorithm 4.4.1 The secant method for a (r, δ)-stationary point

Step 1. Choose any starting point $x^0 \in \mathbb{R}^n$ and set $k = 0$.

Step 2. Apply Algorithm 4.3.1 for the computation of the descent direction at $x = x^k$ for given $\delta > 0, c = c_1$. This algorithm terminates after a finite number of iterations $m > 0$. As a result we get the set $\overline{W}_m(x^k)$ and an element v^k such that

$$\|v^k\|^2 = \min\left\{\|v\|^2 : \ v \in \overline{W}_m(x^k)\right\}.$$

Furthermore either $\|v^k\| \leq \delta$ or for the search direction $g^k = -\|v^k\|^{-1} v^k$

$$f(x^k + rg^k) - f(x^k) \leq -c_1 r \|v^k\|. \tag{4.4.1}$$

Step 3. If

$$\|v^k\| \leq \delta \tag{4.4.2}$$

then stop. Otherwise go to Step 4.

Step 4. Construct the following iteration $x^{k+1} = x^k + \sigma_k g^k$, where σ_k is defined as follows

$$\sigma_k = \text{argmax}\left\{\sigma \geq 0 : \ f(x^k + \sigma g^k) - f(x^k) \leq -c_2 \sigma \|v^k\|\right\}.$$

Set $k = k + 1$ and go to Step 2.

Remark 4.4.1 There are some similarities between the secant method on one side and the bundle method and the gradient sampling method on the other side. The secant method uses a bundle of r-secants which makes it similar to the bundle method. On the other hand, the secant method does not use any rule similar to the subgardient discounting rule which makes the secant method similar the gradient sampling method. However, the secant method uses r-secants instead of subgradients.

For the point $x^0 \in \mathbb{R}^n$ we consider the set $\mathcal{L}(x^0) = \{x \in \mathbb{R}^n : f(x) \leq f(x^0)\}$.

Theorem 4.4.1 *Assume that f is a locally Lipschitz function and the set $\mathcal{L}(x^0)$ is bounded for starting points $x^0 \in \mathbb{R}^n$. Then after a finite number of iterations $K > 0$ Algorithm 4.4.1 produces (r, δ)-stationary point x^K.*

Proof: Since the function f is locally Lipschitz and the set $\mathcal{L}(x^0)$ is bounded

$$f_* = \inf \{f(x) : x \in \mathbb{R}^n\} > -\infty. \tag{4.4.3}$$

Assume the contrary. Then the sequence $\{x^k\}$ is infinite and points x^k are not (r, δ)-stationary points. The latter means that

$$\min\{\|v\| : \ v \in S_r^c f(x^k)\} \geq \delta, \quad \forall x^k, k = 1, 2, \ldots.$$

Therefore Algorithm 4.3.1 will find descent directions and the inequality (4.4.1) will be satisfied at each iteration k. Since $c_2 \in (0, c_1]$ it follows from (4.4.1) that $\sigma_k \geq r$. Then we have

$$\begin{aligned} f(x^{k+1}) - f(x^k) &< -c_2 \sigma_k \|v^k\| \\ &\leq -c_2 r \|v^k\|. \end{aligned}$$

Since $\|v^k\| \geq \delta$ for all k we get

$$f(x^{k+1}) - f(x^k) \leq -c_2 r \delta.$$

Consequently

$$f(x^{k+1}) \leq f(x^0) - (k+1) c_2 r \delta.$$

We get that $f(x^k) \to -\infty$ as $k \to +\infty$ which contradicts (4.4.3), that Algorithm 4.4.1 stops after a finite number of iterations. □

Remark 4.4.2 Since $c_2 \leq c_1$ always $\sigma_k \geq r$ and therefore $r > 0$ is a lower bound for σ_k. This leads to the following rule for the computation of σ_k. We define a sequence:

$$\theta_l = lr, \quad l \geq 1$$

and σ_k is defined as the largest θ_l satisfying the inequality in Step 4.

Algorithm 4.4.1 can be modified to compute Clarke stationary points of the function f that is points x where $0 \in \partial f(x)$. We take sequences $\{r_k\}$, $\{\delta_k\}$ such that

$r_k \to +0$ and $\delta_k \to +0$ as $k \to +\infty$. Applying Algorithm 4.4.1 for each $\{r_k\}$, $\{\delta_k\}$ we get a sequence of (r_k, δ)-stationary points $\{x^k\}$. Convergence of the sequence $\{x^k\}$ to the Clarke stationary points can be proved under more restrictive assumptions.

We assume that the function f satisfies the Assumption 1.2.1 of Section 1.2.

Let $\{r_k\}$, $\{\delta_k\}$ be sequences such that $r_k \to +0$ and $\delta_k \to +0$ as $k \to +\infty$.

Algorithm 4.4.2 The secant method.

Step 1. Choose any starting point $x^0 \in \mathbb{R}^n$ and set $k = 0$.

Step 2. Apply Algorithm 4.4.1 starting from the point x^k for $r = r_k$ and $\delta = \delta_k$. This algorithm terminates after a finite number of iterations $p > 0$ and as a result the algorithm finds (r_k, δ_k)-stationary point x^{k+1}.

Step 3. Set $k = k + 1$ and go to Step 2.

Theorem 4.4.2 *Assume that the function f is locally Lipschitz which satisfies Assumption 1.2.1 at any $x \in \mathbb{R}^n$ and the set $\mathcal{L}(x^0)$ is bounded for starting points $x^0 \in \mathbb{R}^n$. Then every accumulation point of the sequence $\{x^k\}$ belongs to the set $X^0 = \{x \in \mathbb{R}^n : 0 \in \partial f(x)\}$.*

Proof: All conditions of Theorem 4.4.1 are satisfied and therefore Algorithm 4.4.1 for all $k \geq 0$ generates an (r_k, δ_k)-stationary point after the finite number of iterations. Since for any $k > 0$ the point x^{k+1} is (r_k, δ_k)-stationary it follows from the definition of the (r_k, δ_k)-stationarity that

$$\min \left\{ \|v\| : v \in S^c_{r_k} f(x^{k+1}) \right\} \leq \delta_k. \tag{4.4.4}$$

Since $x^k \in \mathcal{L}(x^0)$ for all $k \geq 0$ and the set $\mathcal{L}(x^0)$ is bounded, it follows that the sequence $\{x^k\}$ has at least one accumulation point. Let x^* be an accumulation point and $x^{k_i} \to x^*$ as $i \to +\infty$. then it follows from (4.4.4) that

$$\min \left\{ \|v\| : v \in S^c_{r_{k_i}} f(x^{k_i}) \right\} \leq \delta_{k_i - 1}. \tag{4.4.5}$$

Assumption 1.2.1 implies that at the point x^* for any $\varepsilon > 0$ there exist $r_0 > 0$ and $\eta > 0$ such that

$$S^c_r f(y) \subset \partial f(x^* + B_\varepsilon) + B_\varepsilon \tag{4.4.6}$$

for all $y \in B_\eta(x^*)$ and $r \in (0, r_0)$. Since the sequence $\{x^{k_i}\}$ converges to x^* for $\eta > 0$ there exists $i_0 > 0$ such that $x^{k_i} \in B_\eta(x^*)$ for all $i \geq i_0$. On the other hand since $\delta_k, r_k \to +0$ as $k \to +\infty$ there exists $k_0 > 0$ such that $\delta_k < \varepsilon$ and $r_k < r_0$ for all $k > k_0$. Then there exists $i_1 \geq i_0$ such that $k_i \geq k_0 + 1$ for all $i \geq i_1$. Thus it follows from (4.4.5) and (4.4.6) that

$$\min\{\|v\| : v \in \partial f(x^* + B_\varepsilon)\} \leq 2\varepsilon.$$

Since $\varepsilon > 0$ is arbitrary and the mapping $x \mapsto \partial f(x)$ is upper semicontinuous $0 \in \partial f(x^*)$. □

4.5 Results of Numerical Experiments

In our experiments we use two bundle algorithms for comparisons: PMIN - a recursive quadratic programming variable metric algorithm for minimax optimization and PBUN - a proximal bundle algorithm. The description of these algorithms can be found in [111]. PMIN is applied to minimize maximum functions and PBUN is applied to the rest of problems. In PMIN exact subgradients are used however in PBUN we approximate them using the scheme from Section 1.2.

In Algorithm, 4.4.2 $c_1 \in (0.2, 0.5)$, $c_2 = 0.001$ and subgradients are approximated using the scheme from Section 1.2.

The results of numerical experiments are presented in Table 4.1. We also use present f_{av} the average objective value over 20 runs of algorithms.

Results presented in Table 4.1 show that both the secant method and the bundle method (PMIN) are effective methods for the minimization of nonsmooth convex functions. However, the bundle method is faster and more accurate than the secant method. Results for nonconvex nonsmooth, regular functions show that overall the bundle method (PMIN) produces better results than the secant method on this class of nonsmooth optimization problems. Indeed, the bundle method in 69.71 % of cases finds the best known solutions whereas for the secant method it is only 49.71 %. The bundle method, in 79.80 % of cases, produces the best solutions among two algorithms, but for the secant method is 68.82 %. On the same time we should note that the secant method performs much better than the bundle on some nonsmooth, nonconvex, regular problems (Problems P7, P14, P15).

Overall the secant method performs better than the bundle method (PBUN) on nonconvex, nonsmooth nonregular functions. The secant method in 55.83 % of cases finds the best known solutions whereas the bundle method finds only in 44.58 % of all cases. The secant method produces best solutions among two algorithms in 94.58 % of all cases, but the bundle method only in 64.17 % of all cases. However for some nonconvex, nonsmooth nonregular functions the results by both algorithms are comparable (P22, P23, P31, P32). It should be noted the objective functions in Problems P22-P33 are quasidiferentiable semismooth and their subdifferential and superdifferential are polytopes.

Table 4.2 presents the average number of objective function (n_f) and subgradient (n_{sub}) evaluations as well as the average CPU time (t) for both algorithms on each of test problems.

4.6 Secant Algorithm for Global Optimization

One can see from Algorithm 4.4.2 that, in order to apply it, we take an initial value $r_0 > 0$ of r and gradually reduce it. If $r_0 > 0$ is large then Algorithm 4.3.1 will compute long descent directions which are global descent directions. In this case, the algorithm will overcome some local minimizers. However, the minimization algorithm can only converge to Clarke stationary points. Results of numerical experiments presented in the next section show that this algorithm does not always converge to global minimizers. This means that one needs an algorithm to escape from local minimizers found by Algorithm 4.4.2. We can use Algorithm 4.3.1 with large values of r to find global descent directions from local minimizers found by Algorithm 4.4.2. In this case, we can take some initial value r_0 of r which is not too large. Then we can gradually increase r until we find a new global descent direction. We compute a new starting point for Algorithm 4.4.2 along this direction.

Algorithm 4.6.1 A secant method for global optimization.

Step 0. Choose any starting point $x^0 \in \mathbb{R}^n$ and set $k = 0$.

Step 1. Apply Algorithm 4.4.2 starting from x^k to find stationary point y. Set $x^{k+1} = y$ and set $k = k + 1$.

Step 2. Apply Algorithm 4.3.1 at the point x^k to find a global search direction. If such a descent is not found, then stop. Otherwise this algorithm find a direction g^k.

Step 3. Compute $x^k = x^k + g^k$ and go to Step 1.

It should be noted that this algorithm is still local search algorithm. However, it has very good global search properties. One should not expect that it will always converge to global minimizers. Indeed, we can prove its convergence to Clarke stationary points. Nevertheless, results of numerical experiments presented in the next section and chapter on cluster analysis demonstrate that this algorithm is fast and efficient in solving both smooth and nonsmooth global optimization problems. At the same time, we should emphasis that its efficiency can not be compared with many global optimization algorithms.

4.7 Numerical Results

The efficiency of the proposed algorithm is verified by applying it to two categories of nonsmooth problems. Table 4.3 provides information about first category. In this table n stands for the number of variables and f_{opt} is the known optimal value. Table 4.4 presents the list of second category of test functions. The latter test functions are box-constrained nonsmooth global optimization problems which are proposed by Gaviano et.al [67]. A short explanation is reported in the Section A.0.1. For these test functions, m stands for the predefined number of local minima.

We run the algorithm starting with 20 uniformly distributed initial points. The average of objective function through 20 runs is presented by f_{av}. To show the performance of the algorithm in finding optimal solution, we use the success rates n_b and n_s which are explained in Section A.0.2.

In Tables 4.3 and 4.4, we report the average objective function value over 20 runs of the algorithm as well as success rates. Table 4.3 indicates that the algorithm is

quite successful for more than 50% of test functions, and the result is moderate for 22 % of them. In Table 4.4, we present the numerical results over test functions for nonsmooth global optimization problems. The algorithm is successful regardless of dimension and number of local minima. Only for the test function $n = 10$, $m = 50$ the algorithm gets stuck in a close local optimal solution.

The tables 4.5 and 4.6 present the number of objective function evaluations and CPU time.

4.8 Conclusions

In this chapter, we developed the secant method for minimizing nonsmooth nonconvex functions. We introduced the notion of an r-secant and studied its some properties. The convex hull of all limit points of r-secant is subset of the Clarke subdifferential of semismooth functions. An effective algorithm to approximate subgradients of semismooth quasidifferentiable functions is presented. An algorithm based on r-secants for the computation of descent direction is developed and it is proved that this algorithm is terminating.

We presented results of numerical experiments. In these experiments, minimization problems with nonsmooth, nonconvex and nonregular objective functions were considered. The computational results show that the secant method outperforms the bundle method when the objective is nonsmooth, nonconvex and nonregular and it is semismooth qusidifferentiable function where both subdifferential and superdifferential are polytopes. However, in most cases the secant method requires significantly more objective function, subgradient evaluations as well as CPU time. We applied the secant method straightforward. However taking into account the special structure of the objective functions such as their piecewise partially separability (see [12]) one can significantly reduce the number of the objective function and subgradient evaluations and also CPU time.

Although the secant method is not better than the bundle method for solving nonsmooth convex problems as well as many nonconvex, nonsmooth regular problems, however, it is better than the bundle method for many nonconvex nonsmooth nonregular problems such as semismooth quasidifferentiable problems.

Also, in this chapter we present the global secant method for nonsmooth global optimization problems. This method is an extension of secant method which was presented earlier in this chapter. The numerical results are presented.

Table 4.1: Results of numerical experiments

Prob.	Secant			Bundle		
	f_{av}	n_b	n_s	f_{av}	n_b	n_s
P1	1.95222	20	20	1.95222	20	20
P2	-44	20	20	-44	20	20
P3	3.70348	20	20	3.70348	20	20
P4	0.90750	17	17	0.90750	17	17
P5	0.57592	4	4	0.00000	20	20
P6	3.59972	20	20	3.59972	20	20
P7	-44	20	20	-28.14011	12	12
P8	0.03565	7	13	0.03051	9	17
P9	0.02886	0	7	0.01520	2	16
P10	115.70644	20	20	115.70644	20	20
P11	0.00291	0	0	0.00264	20	20
P12	0.01773	0	6	0.02752	14	16
P13	0.10916	3	9	0.30582	3	15
P14	0.24037	8	18	0.32527	5	9
P15	0.03490	20	20	0.30572	12	12
P16	0.12402	0	11	0.39131	2	9
P17	680.63006	20	20	680.63006	20	20
P18	24.30621	20	20	24.30621	20	20
P19	93.90566	20	20	93.90525	20	20
P20	0.00302	0	0	0.00000	20	20
P21	0.21456	0	9	0.22057	1	14
P22	2.00000	20	20	2.00000	20	20
P23	0.10000	18	18	0.07607	17	17
P24	1.50000	7	18	2.30008	2	10
P25	0.00000	20	20	0.00000	20	20
P26	0.00000	20	20	0.00000	20	20
P27	24.72876	0	18	35.19230	0	2
P28	8.53416×10^6	10	18	10.29861×10^6	5	12
P29	5.56355×10^6	3	20	7.10874×10^6	0	9
P30	2.93562×10^6	2	19	3.16944×10^6	0	2
P31	7.27436×10^5	11	19	7.26213×10^5	12	20
P32	3.94879×10^5	10	19	3.94922×10^5	5	13
P33	1.86467×10^5	13	18	1.88295×10^5	6	9

Table 4.2: The number of function and subgradient evaluations and CPU time

Prob.	Secant			Bundle		
	n_f	n_{sub}	t	n_f	n_{sub}	t
P1	221	160	0.000	10	10	0.001
P2	1113	601	0.002	12	11	0.001
P3	974	701	0.143	65	55	0.003
P4	749	593	0.002	22	9	0.000
P5	4735	461	0.002	493	251	0.001
P6	574	309	0.001	20	16	0.000
P7	1225	579	0.003	146	56	0.001
P8	1184	342	0.009	1626	171	0.009
P9	2968	601	0.003	57891	4843	0.186
P10	1192	619	0.010	29	15	0.001
P11	1829	811	0.005	26081	1904	0.122
P12	4668	1327	0.014	372	173	0.007
P13	1310	749	0.014	61	26	0.002
P14	837	686	0.020	51	23	0.002
P15	1373	1090	0.085	4985	445	0.108
P16	3463	1989	0.302	60331	4761	1.099
P17	1270	860	0.012	58	33	0.000
P18	2234	1592	0.029	18	15	0.000
P19	4752	4001	0.362	35	26	0.003
P20	3399	2733	3.393	160	52	0.030
P21	1529	849	0.249	66280	4974	2.971
P22	289	198	0.001	32	32	0.000
P23	2278	277	0.001	22	22	0.000
P24	470	450	0.003	37	37	0.000
P25	1022	983	0.010	25	25	0.001
P26	2677	920	0.005	68	68	0.001
P27	725	707	0.039	37	37	0.002
P28	305	289	0.099	16	16	0.007
P29	417	401	0.360	21	21	0.022
P30	766	736	2.734	51	51	0.205
P31	283	257	0.254	19	19	0.024
P32	406	370	0.955	28	28	0.085
P33	703	653	7.003	67	67	0.763

Table 4.3: Results of numerical experiments for global secant method

Prob.	Global Secant				
	n	f_{opt}	f_{av}	n_b	n_s
Branin	2	0	0.1492	13	14
Ackleys	2	0	0.2580	18	19
Ackleys	10	0	1.9641	18	18
Ackleys	20	0	10.68431	11	11
Bohachevsky	2	0	0.0000	20	20
Bohachevsky	2	0	0.0000	20	20
Bohachevsky	2	0	0.0000	20	20
Camel	2	-1.0316	-1.0316	20	20
Easom	2	-1	-0.1000	2	5
Gloldestein	2	3	15.1500	17	17
Griewanks	2	0	0.0216	0	6
Griewanks	10	0	0.0385	0	5
Hansen	2	-176.5418	-176.5418	20	20
Hartman	3	-3.8627	-3.8627	20	20
Hartman	6	-3.3224	-3.3044	20	20
Michalewicz	2	-1.8013	-1.5663	12	12
Rastringins	2	0	0.6965	6	11
Rastringins	5	0	2.9351	4	8
Rastringins	10	0	7.7109	1	4

Table 4.4: Results of numerical experiments for global secant method

Prob.	Global Secant				
	n	f_{opt}	f_{av}	n_b	n_s
MDDY ($m = 10$)	2	-1	-1	20	20
MDDY ($m = 50$)	2	-1	-1	20	20
MDDY ($m = 200$)	2	-1	-1	20	20
MDDY ($m = 500$)	2	-1	-1	20	20
MDDY ($m = 10$)	5	-1	-1	20	20
MDDY ($m = 50$)	5	-1	-1	20	20
MDDY ($m = 200$)	5	-1	-1	20	20
MDDY ($m = 500$)	5	-1	-1	20	20
MDDY ($m = 10$)	10	-1	-1	20	20
MDDY ($m = 50$)	10	-1	0.6	12	12
MDDY ($m = 200$)	10	-1	-1	20	20
MDDY ($m = 500$)	10	-1	-1	20	20
MDDY ($m = 10$)	20	-1	-1	20	20
MDDY ($m = 50$)	20	-1	-1	20	20
MDDY ($m = 200$)	20	-1	-1	20	20
MDDY ($m = 500$)	20	-1	-1	20	20
MDDY ($m = 10$)	50	-1	-1	20	20
MDDY ($m = 50$)	50	-1	-1	20	20
MDDY ($m = 200$)	50	-1	-1	20	20
MDDY ($m = 500$)	50	-1	-1	20	20
MDDY ($m = 10$)	100	-1	-1	20	20
MDDY ($m = 50$)	100	-1	-1	20	20
MDDY ($m = 200$)	100	-1	-1	20	20
MDDY ($m = 500$)	100	-1	-1	20	20

Table 4.5: The number of function evaluations and CPU time for first category

Prob.	Global Secant	
	n_f	t
Branin	2251	0.003
Ackleys	2446	0.003
Ackleys	53193	0.119
Ackleys	225096	1.139
Bohachevsky	2114	0.006
Bohachevsky	999	0.003
Bohachevsky	1004	0.003
Camel	959	0.002
Easom	324	0.000
Gloldestein	1351	0.004
Griewanks	3632	0.017
Griewanks	3010	0.018
Hansen	1545	0.006
Hartman	2185	0.009
Hartman	2958	0.021
Michalewicz	1234	0.002
Rastringins	1675	0.004
Rastringins	3625	0.022
Rastringins	6123	0.055

Table 4.6: The number of function evaluations and CPU time for second category

Prob.	Global Secant	
	n_f	t
MDDY ($n = 2$, $m = 10$)	4003	0.005
MDDY ($n = 2$, $m = 50$)	2106	0.005
MDDY ($n = 2$, $m = 200$)	2109	0.008
MDDY ($n = 2$, $m = 500$)	2220	0.019
MDDY ($n = 5$, $m = 10$)	9146	0.012
MDDY ($n = 5$, $m = 50$)	5230	0.015
MDDY ($n = 5$, $m = 200$)	5719	0.033
MDDY ($n = 5$, $m = 500$)	4654	0.063
MDDY ($n = 10$, $m = 10$)	19372	0.026
MDDY ($n = 10$, $m = 50$)	16326	0.035
MDDY ($n = 10$, $m = 200$)	17794	0.137
MDDY ($n = 10$, $m = 500$)	21628	0.373
MDDY ($n = 20$, $m = 10$)	19114	0.032
MDDY ($n = 20$, $m = 50$)	62030	0.231
MDDY ($n = 20$, $m = 200$)	13172	0.176
MDDY ($n = 20$, $m = 500$)	22976	0.713
MDDY ($n = 50$, $m = 10$)	74438	0.226
MDDY ($n = 50$, $m = 50$)	241407	1.716
MDDY ($n = 50$, $m = 200$)	59927	0.713
MDDY ($n = 50$, $m = 500$)	130438	7.052
MDDY ($n = 100$, $m = 10$)	214240	2.430
MDDY ($n = 100$, $m = 50$)	160649	4.926
MDDY ($n = 100$, $m = 200$)	120837	23.12
MDDY ($n = 100$, $m = 500$)	507013	178.1

Chapter 5

Quasi Secant Method

In this chapter, we propose a new algorithm for solving the problem

$$\begin{cases} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathbb{R}^n, \end{cases} \tag{5.0.1}$$

which is efficient for the minimization of nonsmooth nonconvex functions. We also introduce the notions of a secant and a quasi secant for locally Lipschitz functions. The new minimization algorithm uses quasi secants to compute descent directions. The main difference between the notion of a secant in the quasi secant method and in the secant method introduced in Chapter 4 is in the manner of calculation of secants. In the secant method, we only used function values to calculate secants, whereas for the quasi secant method, we calculate the whole set of quasi secants for different types of nonsmooth functions.

The structure of this chapter is as follows: The notions of a secant and a quasi secant are introduced in Section 5.1. In Section 5.2, we describe an algorithm for the computation of a descent direction. A quasi secant method is introduced and its convergence is studied in Section 5.3. Results of numerical experiments are given in Section 5.4. Section 5.5 concludes the chapter.

5.1 Secants and Quasi Secants

The concept of secants is widely used in optimization. For example the secants have been used to design quasi-Newton methods. In this section we introduce the notion

of secants and quasi secants for locally Lipschitz functions. We start with univariate functions.

Consider a function $\varphi : \mathbb{R}^1 \to \mathbb{R}^1$ and assume that it is locally Lipschitz. A secant is a line passing through at least two points on the graph of the function φ. A straight line passing through points $(x, \varphi(x))$ and $(x+h, \varphi(x+h))$, where $h > 0$, is given by:

$$l(x) = cx + d$$

where

$$c = \frac{\varphi(x+h) - \varphi(x)}{h}, \quad d = \varphi(x) - cx.$$

The equation

$$\varphi(x+h) - \varphi(x) = ch \tag{5.1.1}$$

is called a secant equation (see Fig. 5.1). For smooth functions the secant line is close to the tangent line if h is sufficiently small.

We can give another representation of the number c in the secant equation using Lebourg's mean value theorem (see [64]). This theorem implies that there exist $y \in (x, x+h)$ and $u \in \partial\varphi(y)$ such that

$$\varphi(x+h) - \varphi(x) = uh,$$

and one can take $c = u$. However, it is not easy to compute the point y and consequently the subgradient u.

Now let us consider a locally Lipschitz function $f : \mathbb{R}^n \to \mathbb{R}^1$. For given $x \in \mathbb{R}^n, g \in S_1$ and $h > 0$ consider the following function $\varphi(t) = f(x+tg),\ t \in \mathbb{R}^1$. It is obvious that

$$f(x+hg) - f(x) = \varphi(h) - \varphi(0).$$

One can give the following definition of secants for the function f by generalizing the definition for the univariate function φ.

Definition 5.1.1 *A vector $u \in \mathbb{R}^n$ is called a secant of the function f at the point x in the direction $g \in S_1$ with the length $h > 0$ if*

$$f(x+hg) - f(x) = h\langle u, g\rangle. \tag{5.1.2}$$

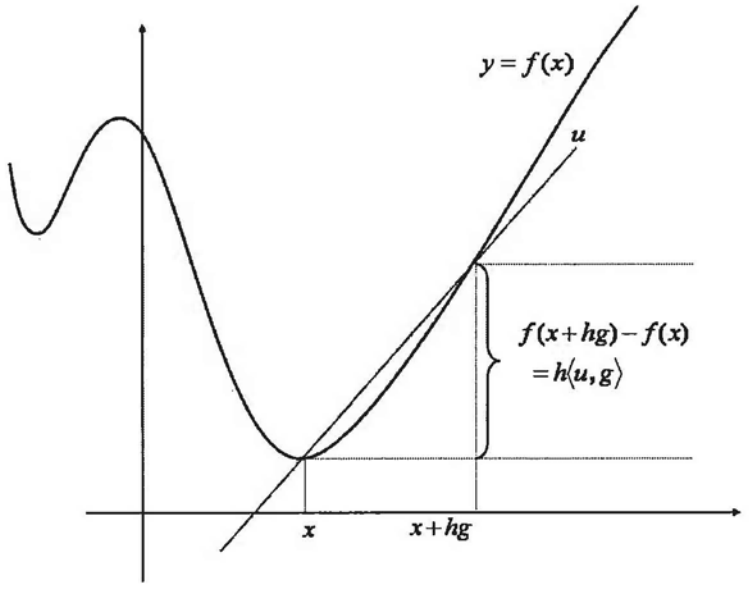

Figure 5.1: Secants for a univariate function

From now on we will use the notation $u(x, g, h)$ for any secant of the function f at a point x in the direction $g \in S_1$ with the length $h > 0$.

For a given $h > 0$, consider a set-valued mapping $x \mapsto Sec(x, h)$:

$$Sec(x, h) = \{w \in \mathbb{R}^n : \exists (g \in S_1),\ w = u(x, g, h)\}.$$

Note that a secant with respect to a given direction is not unique, and there are many vectors u satisfying the equality (5.1.2). Consequently, the mapping $x \mapsto Sec(x, h)$ can be defined in many different ways. However, only secants approximating subgradients of the function f are of interest. One such secant is given by Lebourg's mean value theorem when the function f is locally Lipschitz. Lebourg's theorem implies that there exist $y \in (x, x + hg)$ and $u \in \partial f(y)$ such that

$$f(x + hg) - f(x) = h\langle u, g\rangle.$$

Then it is clear that the subgradient u is a secant at a point x. However it is not easy to compute such subgradients.

Consider the following set at a point x:

$$SL(x) = \left\{w \in \mathbb{R}^n : \ \exists (g \in S_1, \{h_k\},\ h_k \downarrow 0 \text{ as } k \to \infty) :\ w = \lim_{k \to \infty} u(x, g, h_k)\right\}.$$

A mapping $x \mapsto Sec(x, h)$ is called a subgradient-related (SR)-secant mapping if the corresponding set $SL(x) \subseteq \partial f(x)$ for all $x \in \mathbb{R}^n$.

Computation of secants is not always an easy task. For this reason, we introduce the notion of quasi secants by replacing strict equality in the definition of secants by inequality.

Definition 5.1.2 *A vector $v \in \mathbb{R}^n$ is called a quasi secant of the function f at the point x in the direction $g \in S_1$ with the length $h > 0$ if*

$$f(x + hg) - f(x) \leq h\langle v, g\rangle.$$

Fig. 5.2 presents examples of quasi secants in univariate case.

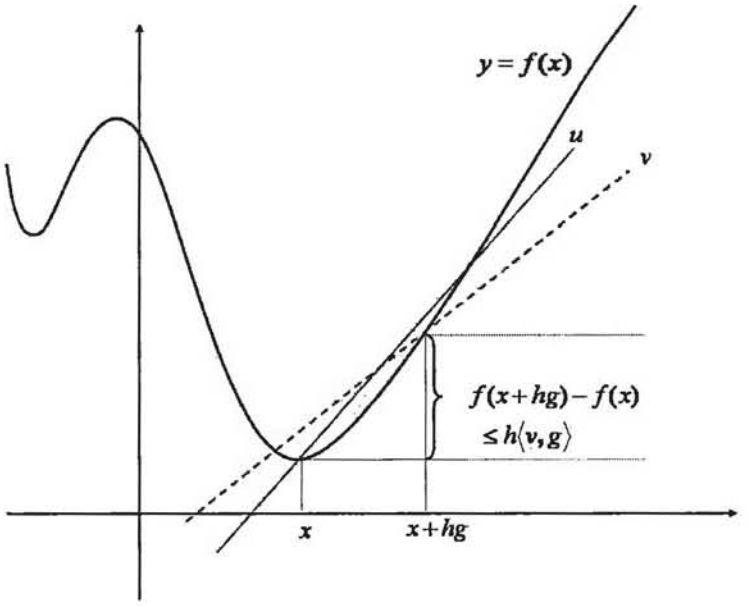

Figure 5.2: Quasi secants for a univariate function

We will use the notation $v(x, g, h)$ for any quasi secant of the function f at the point x in the direction $g \in S_1$ with the length $h > 0$. It is clear that any secant is also a quasi secant. Therefore, the computation of quasi secants must be easier than the computation of secants.

For a given $h > 0$ consider the set of quasi secants of the function f at a point x:

$$QSec(x, h) = \{w \in \mathbb{R}^n : \ \exists (g \in S_1), \ \ w = v(x, g, h)\}$$

and the set of limit points of quasi secants as $h \downarrow 0$:

$$QSL(x) = \left\{ w \in \mathbb{R}^n : \ \exists (g \in S_1, \{h_k\}, \ h_k \downarrow 0 \text{ as } k \to \infty) : \ \ w = \lim_{k \to \infty} v(x, g, h_k) \right\}.$$

A mapping $x \mapsto QSec(x, h)$ is called a subgradient-related (SR)-quasi secant mapping if the corresponding set $QSL(x) \subseteq \partial f(x)$ for all $x \in \mathbb{R}^n$. In this case elements of $QSec(x, h)$ are called SR-quasi secants. In the sequel, we will consider sets $QSec(x, h)$ which contain only SR-quasi secants. Next we will present classes of functions for which SR-quasi secants can be efficiently computed.

5.1.1 Quasi Secants of Smooth Functions

Assume that the function f is continuously differentiable. Then

$$v(x, g, h) = \nabla f(x + hg) + \alpha g, \ \ g \in S_1, \ \ h > 0$$

where

$$\alpha = \frac{f(x + hg) - f(x) - h\langle \nabla f(x + hg), g \rangle}{h}$$

is a secant (also quasi secant) at a point x with respect to the direction $g \in S_1$. Since the function f is continuously differentiable, it is clear that $v(x, g, h) \to \nabla f(x)$ as $h \downarrow 0$, which means that $v(x, g, h)$ is SR-quasi secant at the point x.

5.1.2 Quasi Secants of Convex Functions

Assume that the function f is proper convex (finite valued convex function). Since

$$f(x + hg) - f(x) \leq h\langle v, g \rangle, \ \ \forall v \in \partial f(x + hg),$$

any $v \in \partial f(x + hg)$ is a quasi secant at the point x. Then we have

$$QSec(x, h) = \bigcup_{g \in S_1} \partial f(x + hg).$$

The upper semicontinuity of the mapping $x \mapsto \partial f(x)$ implies that the set $QSL(x) \subset \partial f(x)$. This means that any $v \in \partial f(x + hg)$ is a SR-quasi secant at the point x.

5.1.3 Quasi Secants of Maximum Functions

Consider the following maximum function:

$$f(x) = \max_{i=1,\ldots,m} f_i(x)$$

where the functions f_i, $i = 1, \ldots, m$ are continuously differentiable. Let $v^i \in \mathbb{R}^n$ be a SR-quasi secant of the function f_i at a point x. For any $g \in S_1$ consider the following set

$$R(x + hg) = \{i \in \{1, \ldots, m\} : \ f_i(x + hg) = f(x + hg)\}.$$

The set $QSec(x, h)$ of quasi secants at a point x is defined as

$$QSec(x, h) = \bigcup_{g \in S_1} \left\{ v^i(x, g, h), \ \ i \in R(x + hg) \right\}.$$

Since the mapping $x \mapsto \partial f(x)$ is upper semicontinuous, the set $QSL(x) \subset \partial f(x)$, and quasi secants, which are defined above, are also SR-quasi secants.

5.1.4 Quasi Secants of d.c. Functions

Consider the function

$$f(x) = f_1(x) - f_2(x)$$

where functions f_1 and f_2 are proper convex. Take any subgradients $v^1 \in \partial f_1(x + hg)$, $v^2 \in \partial f_2(x)$. Then the vector $v = v^1 - v^2$ is a quasi secant of the function f at a point x. However, such quasi secants need not be SR-quasi secants. Since d.c. functions are quasidifferentiable ([166]) and if additionally subdifferentials $\partial f_1(x)$ and $\partial f_2(x)$ are polytopes, one can use an algorithm from [24, 28] to compute subgradients v^1 and v^2 such that their difference will converge to a subgradient of the function f at the point x. Thus, we can use this algorithm to compute SR-quasi secants of the function f.

The following important nonsmooth functions are d.c. functions and their subdifferential and superdifferential are polytopes:

$$F_1(x) = \sum_{i=1}^{m} \min_{j=1,\ldots,p} f_{ij}(x),$$

$$F_2(x) = \max_{i=1,\ldots,m} \min_{j=1,\ldots,p} f_{ij}(x).$$

Here functions f_{ij} are continuously differentiable and proper convex. Both functions F_1 and F_2 can be represented as the difference of two convex functions. One can compute their SR-quasi secants using their d.c. representation. Many interesting functions belong to this class. For example, the error function in cluster analysis is of this type ([14]). Continuous piecewise linear functions can be represented as a max-min of linear functions.

Results of this section demonstrate that SR-quasi secants can be efficiently computed for a large class of nonsmooth convex and nonconvex functions. However, the computation of SR-quasi secants for d.c. functions is more costly than for other functions considered in this section.

SR-quasi secants defined in Subsections 5.1.1 - 5.1.4 satisfy the following condition: for any $\varepsilon > 0$ there exists $\delta > 0$ such that

$$QSec(y, h) \subset \partial f(x) + B_\varepsilon(0) \tag{5.1.3}$$

for all $x \in B_\delta(x)$ and $h \in (0, \delta)$. This can be easily proved taking into account upper semicontinuity of the subdifferential.

5.2 Computation of a Descent Direction

From now on, we will assume that for any bounded subset $X \subset \mathbb{R}^n$ and any $h_0 > 0$ there exists $K > 0$ such that

$$\|v\| \leq K$$

for all $v \in QSec(x, h)$, $x \in X$ and $h \in (0, h_0]$. It is obvious that this assumption is true for all functions considered in the previous section. Given $x \in \mathbb{R}^n$ and $h > 0$ we consider the following set:

$$W(x, h) = \overline{\text{co}}\, QSec(x, h).$$

where $\overline{\text{co}}$ is a closed convex hull of a set. It is clear that the set $W(x, h)$ is compact and convex.

Proposition 5.2.1 *Assume that* $0_n \notin W(x, h)$, $h > 0$ *and*

$$\|v^0\| = \min \{\|v\| : \ v \in W(x, h)\} > 0.$$

Then

$$f(x + hg^0) - f(x) \leq -h\|v^0\|$$

where $g^0 = -\|v^0\|^{-1}v^0$.

Proof: Since $W(x,h)$ is a compact and convex set we get

$$\max\left\{\langle v, g^0\rangle : \ v \in W(x,h)\right\} = -\|v^0\|.$$

Then it follows from the definition of quasi secants that

$$\begin{aligned} f(x + hg^0) - f(x) &\leq h\langle v(x, g^0, h), g^0\rangle \\ &\leq h \max\left\{\langle v, g^0\rangle : v \in W(x,h)\right\} \\ &= -h\|v^0\|. \end{aligned}$$

□

Proposition 5.2.1 implies that quasi secants can be used to find descent directions of a function f. Furthermore, this can be done for any $h > 0$. However it is not always possible to apply Proposition 5.2.1 since it assumes the entire set $W(x,h)$ to be known. However, the computation of the entire set $W(x,h)$ is not always possible. It can not even be computed for functions whose subdifferentials are polytopes at any point. Therefore, we propose the following algorithm for computation of descent directions and this algorithm uses only a few elements from $W(x,h)$.

Let the numbers $h > 0$, $c_1 \in (0,1)$ and a small enough number $\delta > 0$ be given.

Algorithm 5.2.1 Computation of the descent direction.

Step 1. Choose any $g^1 \in S_1$ and compute a quasi secant $v^1 = v(x, g^1, h)$ in the direction g^1. Set $V_1(x) = \{v^1\}$ and $k = 1$.

Step 2. Compute $\|\bar{v}^k\|^2 = \min\{\|v\|^2 : v \in \operatorname{co} V_k(x)\}$. If

$$\|\bar{v}^k\| \leq \delta, \tag{5.2.1}$$

then stop. Otherwise go to Step 3.

Step 3. Compute the search direction by $g^{k+1} = -\|\bar{v}^k\|^{-1}\bar{v}^k$.

Step 4. If

$$f(x + hg^{k+1}) - f(x) \leq -c_1 h \|\bar{v}^k\|, \tag{5.2.2}$$

then stop. Otherwise go to Step 5.

Step 5. Compute a quasi secant $v^{k+1} = v(x, g^{k+1}, h)$ in the direction g^{k+1}, construct the set $V_{k+1}(x) = \text{co}\left\{V_k(x) \bigcup \{v^{k+1}\}\right\}$, set $k = k + 1$ and go to Step 2.

Some explanations of Algorithm 5.2.1 follow. In Step 1 we select any direction $g^1 \in S_1$ and compute the initial quasi secant in this direction. The least distance between the convex hull of all computed quasi secants and the origin is found in Step 2. This is a quadratic programming problem and algorithms from [3, 136] can be applied to solve it. In numerical experiments we use the algorithm from [136]. If the least distance is less than a given tolerance $\delta > 0$, then the point x is an approximate stationary point; otherwise, we compute a new search direction in Step 3. If it is the descent direction satisfying (5.2.2) then the algorithm stops (Step 4). Otherwise, we compute a new quasi secant in the direction g^{k+1} in Step 5 which improves the approximation of the set $W(x, h)$.

It should be noted that there are some similarities between the ways descent directions are computed in the bundle-type algorithms and in Algorithm 5.2.1. The latter algorithm is close to the version of the bundle method proposed in [135]. However in the new algorithm we use quasi secants instead of subgradients.

In the next proposition, we prove that the Algorithm 5.2.1 terminates using a standard technique.

Proposition 5.2.2 *Assume that f is a locally Lipschitz function, $h > 0$ and there exists $K, 0 < K < \infty$ such that*

$$\max\{\|v\| : v \in W(x, h)\} \leq K. \tag{5.2.3}$$

If $c_1 \in (0, 1)$ and $\delta \in (0, K)$, then Algorithm 5.2.1 terminates after at most m steps, where

$$m \leq 2\log_2(\delta/K)/\log_2 K_1 + 2, \quad K_1 = 1 - [(1 - c_1)(2K)^{-1}\delta]^2.$$

Proof: In order to prove the proposition it is sufficient to estimate an upper bound for the number of steps m when the condition (5.2.1) is satisfied. If both stopping criteria (5.2.1) and (5.2.2) are not satisfied, then a new quasi secant v^{k+1} computed in Step 5 does not belong to the set $V_k(x)$: $v^{k+1} \notin V_k(x)$. Indeed, in this case $\|\bar{v}^k\| > \delta$ and

$$f(x + hg^{k+1}) - f(x) > -c_1 h \|\bar{v}^k\|.$$

It follows from the definition of the quasi secants that

$$f(x + hg^{k+1}) - f(x) \leq h\langle v^{k+1}, g^{k+1}\rangle,$$

and we have

$$\langle v^{k+1}, \bar{v}^k\rangle < c_1 \|\bar{v}^k\|^2. \tag{5.2.4}$$

Since $\bar{v}^k = \operatorname{argmin}\{\|v\|^2 : v \in V_k(x)\}$, the necessary condition for a minimum implies that

$$\langle \bar{v}^k, v\rangle \geq \|\bar{v}^k\|^2$$

for all $v \in V_k(x)$. The inequality along with (5.2.4) means that $v^{k+1} \notin V_k(x)$. Thus, if both stopping criteria are not satisfied then the algorithm allows one to improve the approximation of the set $W(x, h)$.

It is clear that $\|\bar{v}^{k+1}\|^2 \leq \|tv^{k+1} + (1-t)\bar{v}^k\|^2$ for all $t \in [0, 1]$ which means

$$\|\bar{v}^{k+1}\|^2 \leq \|\bar{v}^k\|^2 + 2t\langle \bar{v}^k, v^{k+1} - \bar{v}^k\rangle + t^2\|v^{k+1} - \bar{v}^k\|^2.$$

It follows from (5.2.3) that

$$\|v^{k+1} - \bar{v}^k\| \leq 2K.$$

Hence, taking into account the inequality (5.2.4), we have

$$\|\bar{v}^{k+1}\|^2 \leq \|\bar{v}^k\|^2 - 2t(1 - c_1)\|\bar{v}^k\|^2 + 4t^2K^2.$$

Let $t_0 = (1 - c_1)(2K)^{-2}\|\bar{v}^k\|^2$. It is clear that $t_0 \in (0, 1)$ and therefore

$$\|\bar{v}^{k+1}\|^2 \leq \left\{1 - [(1 - c_1)(2K)^{-1}\|\bar{v}^k\|]^2\right\}\|\bar{v}^k\|^2. \tag{5.2.5}$$

Since $\|\bar{v}^k\| > \delta$ for all $k = 1, \ldots, m-1$, it follows from (5.2.5) that

$$\|\bar{v}^{k+1}\|^2 \leq \{1 - [(1 - c_1)(2K)^{-1}\delta]^2\}\|\bar{v}^k\|^2.$$

Let $K_1 = 1 - [(1 - c_1)(2K)^{-1}\delta]^2$. Then $K_1 \in (0,1)$ and we have

$$\|\bar{v}^m\|^2 \leq K_1\|\bar{v}^{m-1}\|^2 \leq \ldots \leq K_1^{m-1}\|\bar{v}^1\|^2 \leq K_1^{m-1}K^2.$$

Thus, the inequality (5.2.1) is satisfied if $K_1^{m-1}K^2 \leq \delta^2$. This inequality must happen after at most m steps where

$$m \leq 2\log_2(\delta/K)/\log_2 K_1 + 2.$$

□

Definition 5.2.1 *A point $x \in \mathbb{R}^n$ is called a (h,δ)-stationary point if*

$$\min_{v \in W(x,h)} \|v\| \leq \delta$$

5.3 A Quasi Secant Method

In this section we describe the quasi secant method for solving problem (5.0.1). Let $h > 0,\ \delta > 0,\ c_1 \in (0,1), c_2 \in (0, c_1]$ be given numbers.

Algorithm 5.3.1 The quasi secant method for finding (h,δ)-stationary points.

Step 1. Choose any starting point $x^0 \in \mathbb{R}^n$ and set $k = 0$.

Step 2. Apply Algorithm 5.2.1 for the computation of the descent direction at $x = x^k$ for given $\delta > 0$ and $c_1 \in (0,1)$. This algorithm terminates after a finite number of iterations $m > 0$. As a result, we get the set $V_m(x^k)$ and an element v^k such that

$$\|v^k\|^2 = \min\left\{\|v\|^2 :\ v \in V_m(x^k)\right\}.$$

Furthermore, either $\|v^k\| \leq \delta$ or for the search direction $g^k = -\|v^k\|^{-1}v^k$,

$$f(x^k + hg^k) - f(x^k) \leq -c_1 h\|v^k\|. \tag{5.3.1}$$

Step 3. If

$$\|v^k\| \leq \delta \tag{5.3.2}$$

then stop. Otherwise go to Step 4.

Step 4. Compute $x^{k+1} = x^k + \sigma_k g^k$, where σ_k is defined as follows

$$\sigma_k = \text{argmax}\left\{\sigma \geq 0 : \ f(x^k + \sigma g^k) - f(x^k) \leq -c_2\sigma\|v^k\|\right\}.$$

Set $k = k + 1$ and go to Step 2.

Theorem 5.3.1 *Assume that the function f is bounded below*

$$f_* = \inf\{f(x) : x \in \mathbb{R}^n\} > -\infty. \tag{5.3.3}$$

Then Algorithm 5.3.1 terminates after finite many iterations $M > 0$ and produces (h, δ)-stationary point x^M where

$$M \leq M_0 \equiv \left\lfloor \frac{f(x^0) - f_*}{c_2 h \delta} \right\rfloor + 1$$

Proof: Assume the contrary. Then the sequence $\{x^k\}$ is infinite and points x^k are not (h, δ)-stationary points. This means that

$$\min\{\|v\| : \ v \in W(x^k, h)\} > \delta, \quad \forall k = 1, 2, \ldots.$$

Therefore, Algorithm 5.2.1 will find descent directions and the inequality (5.3.1) will be satisfied at each iteration k. Since $c_2 \in (0, c_1]$, it follows from (5.3.1) that $\sigma_k \geq h$. Therefore, we have

$$\begin{aligned} f(x^{k+1}) - f(x^k) &< -c_2\sigma_k\|v^k\| \\ &\leq -c_2 h\|v^k\|. \end{aligned}$$

Since $\|v^k\| > \delta$ for all $k \geq 0$, we get

$$f(x^{k+1}) - f(x^k) \leq -c_2 h\delta,$$

which implies

$$f(x^{k+1}) \leq f(x^0) - (k+1)c_2 h\delta$$

and therefore, $f(x^k) \to -\infty$ as $k \to +\infty$ which contradicts (5.3.3). It is obvious that the upper bound for the number of iterations M necessary to find the (h, δ)-stationary point is M_0. □

Remark 5.3.1 Since $c_2 \leq c_1$, $\sigma_k \geq h$, and therefore $h > 0$ is a lower bound for σ_k. This leads to the following rule for the estimation of σ_k. We define a sequence:

$$\theta_l = 2^l h, \quad l = 1, 2, \ldots,$$

and σ_k is defined as the largest θ_l satisfying the inequality in Step 4 of Algorithm 5.3.1.

Algorithm 5.3.1 can be applied to compute stationary points of the function f that is points x where $0 \in \partial f(x)$. Let $\{h_k\}$, $\{\delta_k\}$ be sequences such that $h_k \to +0$ and $\delta_k \to +0$ as $k \to \infty$.

Algorithm 5.3.2 The quasi secant method.

Step 1. Choose any starting point $x^0 \in \mathbb{R}^n$, and set $k = 0$.

Step 2. If $0 \in \partial f(x^k)$, then stop.

Step 3. Apply Algorithm 5.3.1 starting from the point x^k for $h = h_k$ and $\delta = \delta_k$. This algorithm terminates after finitely many iterations $M > 0$, and as a result, it computes (h_k, δ_k)-stationary point x^{k+1}.

Step 4. Set $k = k + 1$ and go to Step 2.

For the point $x^0 \in \mathbb{R}^n$, we consider the set $\mathcal{L}(x^0) = \{x \in \mathbb{R}^n : f(x) \leq f(x^0)\}$.

Theorem 5.3.2 *Assume that the function f is locally Lipschitz, the set $W(x, h)$ is constructed using SR-quasi secants, the condition (5.1.3) is satisfied and the set $\mathcal{L}(x^0)$ is bounded for starting points $x^0 \in \mathbb{R}^n$. Then every accumulation point of the sequence $\{x^k\}$ belongs to the set $X^0 = \{x \in \mathbb{R}^n : 0 \in \partial f(x)\}$.*

Proof: Since the function f is locally Lipschitz and the set $\mathcal{L}(x^0)$ is bounded, $f_* > -\infty$. Therefore, conditions of Theorem 5.3.1 are satisfied, and Algorithm 5.3.1 generates a sequence of (h_k, δ_k)-stationary points after the finite number of points for all $k \geq 0$. Since for any $k > 0$ the point x^{k+1} is (h_k, δ_k)-stationary, it follows from the definition of the (h_k, δ_k)-stationary points that

$$\min \left\{ \|v\| : v \in W(x^{k+1}, h_k) \right\} \leq \delta_k. \tag{5.3.4}$$

It is obvious that $x^k \in \mathcal{L}(x^0)$ for all $k \geq 0$. The boundedness of the set $\mathcal{L}(x^0)$ implies that the sequence $\{x^k\}$ has at least one accumulation point. Let x^* be an accumulation point and $x^{k_i} \to x^*$ as $i \to +\infty$. The inequality in (5.3.4) implies that

$$\min\left\{\|v\| : v \in W(x^{k_i}, h_{k_i})\right\} \leq \delta_{k_i - 1}. \tag{5.3.5}$$

The mapping $QSec(\cdot, \cdot)$ satisfies the condition (5.1.3), therefore, at the point x^* for any $\varepsilon > 0$ there exists $\eta > 0$ such that

$$W(y, h) \subset \partial f(x^*) + B_\varepsilon \tag{5.3.6}$$

for all $y \in B_\eta(x^*)$ and $h \in (0, \eta)$. Since the sequence $\{x^{k_i}\}$ converges to x^* there exists $i_0 > 0$ such that $x^{k_i} \in B_\eta(x^*)$ for all $i \geq i_0$. On the other hand since $\delta_k, h_k \to +0$ as $k \to +\infty$ there exists $k_0 > 0$ such that $\delta_k < \varepsilon$ and $h_k < \eta$ for all $k > k_0$. Then there exists $i_1 \geq i_0$ such that $k_i \geq k_0 + 1$ for all $i \geq i_1$. Thus, it follows from (5.3.5) and (5.3.6) that

$$\min\{\|v\| : \ v \in \partial f(x^*)\} \leq 2\varepsilon.$$

Since $\varepsilon > 0$ is arbitrary and the mapping $x \mapsto \partial f(x)$ is upper semicontinuous, $0 \in \partial f(x^*)$. □

5.4 Results of Numerical Experiments

The efficiency of the proposed algorithm was verified by applying it to the academic test problems with nonsmooth objective functions introduced in Appendix A. It is important to note that only in the tables for this section, P3 stands for Problem 2.22 [110] with $n = 10$, $m = 2$ and $f_{opt} = 54.598150$. Also, problems P23-P25 are problems 1-3 from [8] with optimal value $2, 0$ and 0 respectively. The dimensions are $2, 2$ and 6, and the values for m are $6, 0$ and 0 respectively.

In our experiments, we use two bundle algorithms for comparisons: PMIN - a recursive quadratic programming variable metric algorithm for minimax optimization and PBUN - a proximal bundle algorithm. The description of these algorithms can be found in [111]. PMIN is applied to minimize maximum functions and PBUN is applied to solve problems with nonregular objective functions. We compute subgradients in problems with maximum objective functions, and in problems with nonregular

objective functions, we approximate subgradients using the scheme from [24, 28]. In Algorithm 5.3.2 $c_1 = 0.2$, $c_2 = 0.05$, $\delta_k = 10^{-7}$, $h_{k+1} = 0.5h_k$, $k \geq 1$ and $h_1 = 1$.

First we applied both PMIN and Algorithm 5.3.2 for solving problems P1-P22 using starting points from [110]. Results are presented in Table 5.1. In this table we present the value of the objective function at the final point (f) and the number of function and subgradient evaluations (n_f and n_{sub}, respectively). These results demonstrate that the bundle method performs better than the secant method. The latter method uses significantly less function and subgradient evaluations. The secant method fails to find the best known solutions for problems P9 and P10 whereas the bundle method finds those solutions for all problems.

Table 5.1: Results of numerical experiments with given starting points

Prob.	Quasi secant			Bundle		
	f	n_f	n_{sub}	f	n_f	n_{sub}
P1	1.95222	288	134	1.95222	8	8
P2	-44	436	238	-44	16	12
P3	54.60361	469	137	54.59815	88	36
P4	3.70348	509	266	3.70348	18	17
P5	0	244	164	0	8	8
P6	0	511	241	0	180	94
P7	3.59972	709	208	3.59972	15	14
P8	-44	379	285	-44	21	13
P9	0.05061	2639	881	0.00420	9	9
P10	0.01983	455	254	0.00808	12	11
P11	115.70644	432	310	115.70644	11	11
P12	0.00264	3092	1439	0.00264	113	36
P13	0.00202	1633	995	0.00202	86	35
P14	0.00012	1214	892	0.00012	8	7
P15	0.02234	777	537	0.02234	57	17
P16	0.03490	868	681	0.03490	53	22
P17	0.00619	1194	960	0.00619	107	20
P18	680.63006	431	282	680.63006	45	20
P19	24.30621	836	679	24.30621	19	14
P20	93.90525	1991	1703	93.90525	30	21
P21	0.00000	15104	11149	0.00000	20	19
P22	0.04803	3187	2880	0.04803	286	68

At the next step we applied both methods to solve all problems using 20 randomly generated starting points. The results of numerical experiments are presented in

Tables 5.2 and 5.3. In Table 5.2 we present f_{av} - the average objective value over 20 runs of algorithms and also numbers n_b and n_s for each method (refer to A for definitions).

Table 5.2: Results of numerical experiments

Prob.	Quasi secant			Bundle		
	f_{av}	n_b	n_s	f_{av}	n_b	n_s
P1	1.95222	20	20	1.95222	20	20
P2	-44	20	20	-44	20	20
P3	54.59890	20	20	54.59815	20	20
P4	3.70348	20	20	3.70348	20	20
P5	1.21000	16	17	0.90750	17	18
P6	0	20	20	0	20	20
P7	3.59972	20	20	3.59972	20	20
P8	-44	20	20	-28.14011	12	12
P9	0.04646	4	7	0.03051	9	18
P10	0.01789	0	9	0.01520	2	17
P11	115.70644	20	20	115.70644	20	20
P12	0.00264	20	20	0.00264	20	20
P13	0.00627	19	19	0.02752	14	14
P14	0.10662	3	16	0.30582	3	15
P15	0.27411	7	17	0.32527	5	10
P16	0.09631	17	19	0.30572	12	13
P17	0.09312	0	10	0.39131	2	10
P18	680.63006	20	20	680.63006	20	20
P19	24.30621	20	20	24.30621	20	20
P20	93.90525	20	20	93.90525	20	20
P21	0.00000	20	20	0.00000	20	20
P22	0.24743	0	9	0.22057	1	16
P23	2	20	20	2	20	20
P24	0	20	20	0.07607	17	17
P25	1.5	8	17	2.30008	2	8

Results presented in Table 5.2 show that both the quasi secant method and the bundle method (PMIN) are effective methods for the minimization of nonsmooth convex functions. However, the bundle method is faster and more accurate than the secant method. Both algorithms perform similarly for nonconex nonsmooth, regular functions. However, results for f_{av} show that the bundle method is more sensitive to the choice of starting points than the quasi secant method. Overall the quasi secant method performs better than the bundle method (PBUN) on nonconvex, nonsmooth

nonregular functions.

Table 5.3 presents the average number of objective function (n_f) and subgradient (n_{sub}) evaluations as well as the average CPU time t (in seconds) over 20 runs of the algorithms. These results demonstrate that the bundle method uses significantly less function and subgradient evaluations to minimize convex functions. Results for nonconvex regular functions show that the bundle method is sensitive to the choice of starting points. The numbers n_f and n_{sub} are large for some starting points and therefore their average values are very large for some problems (Problems P9, P10, P12, P16, P22). The quasi secant method is much less sensitive to the choice of starting points.

Table 5.3: The number of function and subgradient evaluations and CPU time

Prob.	Quasi secant			Bundle		
	n_f	n_{sub}	t	n_f	n_{sub}	t
P1	269	132	0.00	10	10	0.00
P2	347	207	0.00	12	11	0.00
P3	304	121	0.00	66	35	0.00
P4	544	327	0.01	65	55	0.00
P5	242	146	0.00	22	9	0.00
P6	1729	771	0.00	493	251	0.00
P7	425	181	0.00	20	16	0.00
P8	603	408	0.00	146	56	0.00
P9	608	232	0.00	1626	171	0.01
P10	1060	372	0.00	57891	4843	0.19
P11	443	280	0.00	29	15	0.00
P12	3170	1487	0.01	26081	1904	0.12
P13	1807	1091	0.01	372	173	0.01
P14	1086	775	0.00	61	26	0.00
P15	785	529	0.01	51	23	0.00
P16	856	669	0.02	4985	445	0.11
P17	2460	1791	0.10	60331	4761	1.10
P18	445	272	0.00	58	33	0.00
P19	990	801	0.01	18	15	0.00
P20	2035	1745	0.03	35	26	0.00
P21	14427	11108	0.67	160	52	0.03
P22	1171	818	0.03	66280	4974	2.97
P23	185	104	0.00	32	32	0.00
P24	228	106	0.00	22	22	0.00
P25	459	251	0.00	37	37	0.00

5.5 Conclusions

In this chapter, we developed the quasi secant method for minimizing nonsmooth nonconvex functions. We introduced the notion of a secant and quasi secant and demonstrated how they can be computed for some classes of nonsmooth functions. We proposed an algorithm for the computation of descent directions using quasi secants.

We presented results of numerical experiments and compared the quasi secant method with the bundle method. The computational results show that the bundle method performs significantly better than the quasi secant method for minimizing nonsmooth convex functions whereas the quasi secant method outperforms the bundle method when the objective function is nonsmooth, nonconvex nonregular. Results for nonsmooth, nonconvex regular functions are mixed. These results show that the quasi secant method is much less sensitive to the choice of starting points than the bundle method.

Chapter 6

Application to Cluster Analysis

In this chapter, we present the mathematical formulation of clustering problem, and convert it to a nonsmooth nonconvex optimization problem. Then, we provide a review of the methods applied for solving this problem. An algorithm to solve this problem based on the nonsmooth formulation is described. In this algorithm, we use the secant method (described in Chapter 4) to solve nonsmooth nonconvex subproblems. Finally, we present numerical experiments and compare the results with some other clustering algorithms in this area and conclude the chapter.

6.1 Optimization Based Cluster Analysis

In this section, a general description of the clustering problem is presented. Besides different approaches dealing with this problem, we provide a formulation of the problem as a nonsmooth nonconvex problem.

Clustering is the *unsupervised* classification of patterns. Cluster analysis deals with the problems of classifying of a collection of patterns into groups (called clusters) based on similarity. It has found many applications, including cancer detection, information retrieval, image segmentation and etc.

Consider a set X of a finite number of points of n-dimensional space $\mathbb{R}^n$, that is

$$X = \{x^1, \ldots, x^m\}, \text{ where } x^i \in \mathbb{R}^n, \ i = 1, \ldots, m.$$

The subject of cluster analysis is the partitioning of the set X into a given number q of overlapping or disjoint subsets C_i, $i = 1, \ldots, q$ with respect to predefined criteria

so that

$$X = \bigcup_{i=1}^{q} C_i.$$

The sets $C_i,\ i = 1, \ldots, q$ 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 section, we will exclusively consider the hard unconstrained clustering problem, that is the condition

$$C_i \bigcap C_k = \emptyset, \quad \forall i, k = 1, \ldots, q,\ i \neq k$$

is additionally assumed, and no constraints are imposed on the clusters $C_i,\ i = 1, \ldots, q$. Therefore, every point $x \in X$ is contained in exactly one set C_i.

Each cluster C_i can be identified by its center (or centroid). Using the centroids, we can express the clustering problem in terms of the following optimization problem (see [78, 79, 72]):

$$\text{minimize } \varphi(C, a) = \frac{1}{m} \sum_{i=1}^{q} \sum_{x \in C_i} \|a^i - x\|^2 \tag{6.1.1}$$

$$\text{subject to } C \in \overline{C},\ a = (a^1, \ldots, a^q) \in \mathbb{R}^{n \times q}$$

where $\|\cdot\|$ denotes the Euclidean norm, $C = \{C_1, \ldots, C_q\}$ is a set of clusters, $\overline{C}$ is a set of all possible q-partitions of the set X, a^i is the center of the cluster $C_i,\ i = 1, \ldots, q$:

$$a^i = \frac{1}{|C_i|} \sum_{x \in C_i} x,$$

and $|C_i|$ is the cardinality of the set $C_i,\ \ i = 1, \ldots, q$. The problem (6.1.1) is also known as the minimum sum-of-squares clustering. The combinatorial formulation (6.1.1) of the minimum sum-of-squares clustering is not suitable for direct application of mathematical programming techniques. The problem (6.1.1) can be rewritten as the following mathematical programming problem:

$$\text{minimize } \psi(a, w) = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{q} w_{ij} \|a^j - x^i\|^2 \tag{6.1.2}$$

$$\text{subject to } \sum_{j=1}^{q} w_{ij} = 1,\ i = 1, \ldots, m,$$

and

$$w_{ij} \in \{0, 1\},\ i = 1, \ldots, m,\ j = 1, \ldots, q.$$

Here

$$a^j = \frac{\sum_{i=1}^{m} w_{ij} x^j}{\sum_{i=1}^{m} w_{ij}}, \quad j = 1, \ldots, q$$

and w_{ij} is the association weight of pattern x^i with cluster j (to be found), given by

$$w_{ij} = \begin{cases} 1 & \text{if pattern } i \text{ is allocated to cluster } j\ , \\ & \forall i = 1, \ldots, m,\ j = 1, \ldots, q, \\ 0 & \text{otherwise.} \end{cases}$$

6.1.1 The nonsmooth Optimization Approach to the Clustering

In this section we present a formulation of the clustering problem in terms of nonsmooth, nonconvex optimization.

The problems (6.1.1) and (6.1.2) can be reformulated as the following mathematical programming problem (see [25, 26, 78, 79])

$$\text{minimize } f(a^1, \ldots, a^q) \quad \text{subject to } (a^1, \ldots, a^q) \in \mathbb{R}^{n \times q}, \tag{6.1.3}$$

where

$$f(a^1, \ldots, a^q) = \frac{1}{m} \sum_{i=1}^{m} \min_{j=1,\ldots,q} \|a^j - x^i\|^2. \tag{6.1.4}$$

It is shown in [78] that problems (6.1.1), (6.1.2) and (6.1.3) are equivalent. The number of variables in problem (6.1.2) is $(m+n) \times q$ whereas in problem (6.1.3) 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. Also, in the hard clustering problems the coefficients w_{ij} are integer, that is the problem (6.1.2) contains both integers and continuous variables. In the nonsmooth optimization formulation of the clustering problem we have only continuous variables. All these circumstances can be considered as advantages of the nonsmooth optimization formulation (6.1.3).

If $q > 1$, the objective function (6.1.4) in problem (6.1.3) 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 (6.1.3). On the other hand, this approach provides a "deep" 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 cover the set X under consideration. In order to avoid this difficulty, a step-by-step calculation of clusters is implemented in the optimization algorithm discussed in the next section.

6.2 A Brief Overview of Clustering Algorithms

There exist several methods to deal with clustering including agglomerative and divisive hierarchical clustering algorithms as well as algorithms based on mathematical programming techniques. Some good references to study algorithms for clustering problem can be found in [138, 81, 72]. Also, [87] is an excellent up-to-date survey of existing methods and related literature.

Problem (6.1.2) is a global optimization problem, and different global search algorithms can be applied to solve this problem. Some review of these efforts presented in [133] including dynamic programming, branch and bound, cutting planes, k-means algorithms. Dynamic programming is efficient when number of instances $m \leq 20$, which means this approach is not suitable for real-world problems (see for example [145]). However, when $q = 1$ the minimum sum-of-squares clustering problem can be solved exactly by dynamic programming in polynomial time [72].

One step ahead, branch and bound algorithms are effective when the database contains only hundreds of records and the number of clusters is not large (less than 5) (see [65, 132, 104, 133]). These methods are absolutely incapable when $m \geq 1000$ and $q \geq 10$. Some heuristic methods like k-means can be used for solving large clustering problems. Different versions of this algorithm have been studied by many authors (see [72]). The method is a very fast algorithm and it is suitable for solving clustering problems in large data sets. The k-means Algoirthm provides good results when

there are few clusters but deteriorates when there are many [133]. This algorithm achieves a local minimum of problem (6.1.1)(see [161]); however, results of numerical experiments presented, for example, in [76] show that the best clustering found with k-means may be more than 50 % worse than the best known one.

Although the metaheuristic methods produced better results compared with previous methods, they take a lot of CPU time and so are considered ineffective for large problems [52]. In [57, 160, 159], the simulated annealing method is investigated, and tabu search method is applied to clustering problem in [105]. Furthermore, the genetic algorithm is tried in this problem in [52]. The results of numerical experiments, presented in [106] 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 databases one can expect that this difference will increase. This makes metaheuristic algorithms of global optimization ineffective for solving many clustering problems. However, these algorithms can be applied to large clustering problems if combined with decomposition (see [77]).

In [124] an interior point method for minimum sum-of-squares clustering problem is developed. The paper [77] develops variable neighborhood search algorithm and the paper [134] 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 [116]. The incremental approach is also studied in the paper [76]. Results of numerical experiments presented show the high effectiveness of these algorithms for many clustering problems.

6.3 An Optimization Clustering Algorithm

In this section we will describe an incremental algorithm for solving clustering problem.

Algorithm 6.3.1 An algorithm for solving the cluster analysis problem.

Step 1. (Initialization). Select a tolerance $\epsilon > 0$. Select a starting point $a^0 = (a_1^0, \ldots, a_n^0) \in \mathbb{R}^n$ and solve the minimization problem (6.1.3) with $q = 1$. Let $a^{1*} \in$

$\mathbb{R}^n$ be a solution to this problem and f^{1*} be the corresponding objective function value. 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:

$$\text{minimize} \quad \bar{f}^k(y) \quad \text{subject to} \quad y \in \mathbb{R}^n \tag{6.3.1}$$

where

$$\bar{f}^k(y) = \sum_{i=1}^{m} \min\left\{\|a^{1*} - x^i\|^2, \ldots, \|a^{k*} - x^i\|^2, \|y - x^i\|^2\right\}.$$

Step 3. (Refinement of all cluster centers). Let $\bar{y}^{k+1,*}$ be a solution to problem (6.3.1). Take $a^{k+1,0} = (a^{1*}, \ldots, a^{k*}, \bar{y}^{k+1,*})$ as a new starting point and solve the following minimization problem:

$$\text{minimize} \quad f^{k+1}(a) \quad \text{subject to} \quad a = (a^1, \ldots, a^{k+1}) \in \mathbb{R}^{n\times(k+1)} \tag{6.3.2}$$

where

$$f^{k+1}(a) = \sum_{i=1}^{m} \min_{j=1,\ldots,k+1} \|a^j - x^i\|^2.$$

Step 4. (Stopping criterion). Let $a^{k+1,*}$ be a solution to the problem (6.3.2) and $f^{k+1,*}$ be the corresponding value of the objective function. If

$$\frac{f^{k*} - f^{k+1,*}}{f^{1*}} < \epsilon$$

then stop, otherwise set $k = k + 1$ and go to Step 2.

In Step 1, the centers of the first q_0 clusters are calculated. In particular, one can take $q_0 = 1$. In this case the center of the entire set X will be calculated. In Step 2, we calculate a center of the next $(k+1)$-st cluster, assuming the previous k cluster centers to be known. It should be noted, the number of variables in problem (6.3.1) is n which is substantially less than if we calculate all cluster centers simultaneously. In Step 3, the refinement of all $k+1$ cluster centers is carried out. Since the starting point $a^{k+1,0}$ calculated in the previous step, Step 2 is not far from the solution to problem (6.3.2), and it takes only a moderate number of iterations to calculate it. Such an approach allows one to significantly reduce the computational time for solving problem (6.3.2).

Table 6.1: The brief description of data sets

Data sets	Number of instances	Number of attributes
German towns	59	2
Bavaria postal 1	89	3
Bavaria postal 2	89	4
Fisher's Iris Plant	150	4
Heart Disease	297	13
Liver Disorders	345	6
Ionosphere	351	34
Congressional Voting Records	435	16
Breast Cancer	683	9
Pima Indians Diabetes	768	8
TSPLIB1060	1060	2
Image Segmentation	2310	19
TSPLIB3038	3038	2
Page Blocks	5473	10

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.

To solve the nonsmooth nonconvex problem 6.3.2, we use the secant method, which is introduced in Chapter 4. As we will see in the next section, this algorithm creates better solutions for the clustering problem.

6.4 Numerical Experiments

To verify the efficiency of the proposed algorithm, numerical experiments with a number of real-world data sets have been conducted. Fourteen data sets have been used in numerical experiments. The brief description of the data sets is given in Table 6.1. The detailed description of German towns, Bavaria postal data sets can be found in [72], Fisher's Iris Plant data set in [144], the traveling salesman problems TSPLIB1060 and TSPLIB3038 in [66] and all other data sets in [1].

We computed up to 10 clusters in data sets with no more than 150 instances, up to 50 clusters in data sets with the number of instances between 150 and 1000 and up

to 100 clusters in data sets with more than 1000 instances. The multi-start k-means (MS k-means) and the global k-means algorithms (GKM) have been used in numerical experiments for comparison purpose (for description of these algorithms see [10]). To find k clusters, 100 times k starting points were randomly chosen in the MS k-means algorithm for all data sets and starting points were data points. In the GKM and secant algorithms a distance matrix $D = (d_{ij})_{i,j=1}^{m}$ of a data set was computed before the start of the algorithms. Here $d_{ij} = \|a^i - a^j\|^2$. This matrix was used by both algorithms to find starting points.

Results of numerical experiments are presented in Tables 6.2-6.8. In these tables we use the following notation:

- k is the number of clusters;
- f_{opt} is the best known value of the cluster function (6.1.4) (multiplied by m) for the corresponding number of clusters. For German towns, Bavaria Postal 1 and 2 and Iris Plant data sets, f_{opt} is the value of the cluster function at the known global minimizer (see [76]);
- E is the error in %;
- N is the number of Euclidean norm evaluations for the computation of the corresponding number of clusters. To avoid big numbers in tables we use its expression in the form $N = \alpha \times 10^l$ and present the values of α in tables. Thus $l = 4$ for German towns, Bavaria Postal 1 and 2, Iris Plant data sets, $l = 5$ for Heart Disease, Liver Disorders, Ionosphere, Congressional Voting Records data sets, $l = 6$ for Breast Cancer, Pima Indians Diabetes, TSPLIB1060, Image Segmentation data sets and $l = 7$ for TSPLIB3038, Page Blocks data sets.
- t is the CPU time (in seconds).
- In Table 6.9, $\sum E$ stands for accumulated errors for each data set in previous tables, and s.r stands for number of times the algorithm could find the global optimal solution or near solution ($0 \leq E \leq 1$).

The values of f_{opt} for German towns, Bavaria postal, Iris Plant, Image Segmentation ($k \leq 50$), TSPLIB1060 ($k \leq 50$) and TSPLIB3038 ($k \leq 50$) data sets are

available, for example, in [14, 76]. In all other cases we take as f_{opt} the best value obtained by the MS k-means, GKM and modified global k-means algorithms (see [10]).

The error E is computed as

$$E = \frac{(\bar{f} - f_{opt})}{f_{opt}} \cdot 100, \tag{6.4.1}$$

where $\bar{f}$ is the best value (multiplied by m) of the objective function (6.1.4) obtained by an algorithm. The condition $E = 0$ implies that an algorithm finds the best known solution. We say that an algorithm finds a near global (or best known) solution if $0 \leq E \leq 1$.

Table 6.2: Results for German towns and Bavaria postal 1 data sets

k	f_{opt}	MS k-means			GKM			Secant		
		E	α	t	E	α	t	E	α	t
German towns										
2	$0.12143 \cdot 10^6$	0.00	6.80	0.00	0.00	0.230	0.00	0.00	1.38	0.00
3	$0.77009 \cdot 10^5$	0.00	13.6	0.00	1.45	0.289	0.00	1.45	2.92	0.00
4	$0.49601 \cdot 10^5$	0.00	23.6	0.00	0.72	0.366	0.00	0.72	4.71	0.00
5	$0.38716 \cdot 10^5$	0.00	31.2	0.00	0.00	0.490	0.00	0.00	6.77	0.00
6	$0.30536 \cdot 10^5$	0.00	38.3	0.00	0.00	0.602	0.00	0.27	9.90	0.00
7	$0.24433 \cdot 10^5$	5.35	44.9	0.00	0.09	0.732	0.00	0.00	13.3	0.00
8	$0.21748 \cdot 10^5$	0.33	46.1	0.00	0.10	0.832	0.00	0.00	15.9	0.00
9	$0.18946 \cdot 10^5$	4.14	57.8	0.00	0.00	0.997	0.00	2.28	18.8	0.00
10	$0.16555 \cdot 10^5$	13.98	61.4	0.02	0.28	1.120	0.00	0.00	21.9	0.00
Bavaria postal 1										
2	$0.60255 \cdot 10^{12}$	0.00	11.7	0.00	7.75	0.445	0.00	0.00	5.26	0.02
3	$0.29451 \cdot 10^{12}$	0.00	30.5	0.00	0.00	0.507	0.00	0.00	8.98	0.02
4	$0.10447 \cdot 10^{12}$	0.00	43.0	0.00	0.00	0.730	0.00	0.00	14.7	0.02
5	$0.59762 \cdot 10^{11}$	0.00	67.6	0.00	0.00	1.050	0.00	0.00	19.8	0.02
6	$0.35909 \cdot 10^{11}$	27.65	76.0	0.00	0.00	1.170	0.00	0.00	24.0	0.02
7	$0.21983 \cdot 10^{11}$	0.61	107	0.02	1.50	1.550	0.00	1.50	42.0	0.03
8	$0.13385 \cdot 10^{11}$	0.00	124	0.03	0.00	1.980	0.00	0.00	55.2	0.03
9	$0.84237 \cdot 10^{10}$	35.81	135	0.03	0.00	2.150	0.00	0.00	69.5	0.05
10	$0.64465 \cdot 10^{10}$	30.67	160	0.03	0.00	2.870	0.00	0.00	90.2	0.05

The results presented in Table 6.2 show that the MS k-means algorithm is effective when the number of clusters is small. For German towns the results of the GKM and secant method are quite similar. Considering the summary Table 6.9, for a small data set like this, GKM has higher success rate, and the accumulated error is lower. The

similar scenario is repeated for MS k-means for Bavaria postal 1 data set, and it is successful only for small number of clusters. For this data set, the secant method outperforms GKM both in success rate and accumulated error.

Table 6.3: Results for Bavaria postal 2 and Iris Plant data sets

k	f_{opt}	MS k-means			GKM			Secant		
		E	α	t	E	α	t	E	α	t
Bavaria postal 2										
2	$0.19908 \cdot 10^{11}$	26.16	13.3	0.00	0.00	0.445	0.00	0.00	2.85	0.00
3	$0.17399 \cdot 10^{11}$	0.00	23.9	0.00	0.00	0.507	0.00	0.00	10.2	0.00
4	$0.75591 \cdot 10^{10}$	0.00	40.9	0.00	0.00	0.659	0.00	0.00	29.0	0.01
5	$0.53429 \cdot 10^{10}$	0.00	53.5	0.00	1.86	0.801	0.00	1.86	42.8	0.01
6	$0.32263 \cdot 10^{10}$	37.37	69.4	0.00	0.00	0.917	0.00	0.00	47.8	0.01
7	$0.22271 \cdot 10^{10}$	10.75	91.6	0.00	0.00	1.49	0.00	0.00	82.0	0.03
8	$0.17170 \cdot 10^{10}$	12.31	106	0.03	0.00	1.71	0.00	0.00	88.5	0.03
9	$0.14030 \cdot 10^{10}$	9.50	126	0.03	0.00	2.12	0.00	0.00	117	0.06
10	$0.11928 \cdot 10^{10}$	18.88	132	0.05	0.00	2.31	0.00	0.00	125	0.06
Iris Plant										
2	152.348	0.00	17.8	0.00	0.00	1.26	0.00	0.00	7.51	0.01
3	78.851	0.00	41.9	0.00	0.01	1.78	0.00	0.01	16.8	0.01
4	57.228	0.00	81.4	0.03	0.05	2.21	0.00	0.05	27.4	0.01
5	46.446	0.00	105	0.05	0.54	2.53	0.02	0.54	38.6	0.01
6	39.040	0.00	121	0.05	1.44	2.81	0.02	1.44	25.9	0.01
7	34.298	4.20	157	0.05	3.17	3.14	0.02	3.17	66.0	0.01
8	29.989	10.69	171	0.05	1.71	3.88	0.02	1.71	86.7	0.03
9	27.786	2.31	184	0.06	2.85	4.16	0.02	2.85	102	0.03
10	25.834	8.27	212	0.08	3.55	4.48	0.02	3.56	122	0.03

As one can see from Table 6.3, the GKM and secant algorithms find the same solutions for both Bavaria postal 2 and Iris Plant data sets. However, the GKM algorithm requires less computational effort than the secant algorithm. For the Iris plant data set, MS k-means is successful five times while the other two are successful 4 times; however, the accumulated error for secant and GKM is less than for MS k-means.

The results from Table 6.4 demonstrate that the MS k-means algorithm cannot locate the global solution for Heart Disease data set when $k > 5$ and for Liver Disorders data set when $k > 10$. For Heart Disease data set the GKM algorithm does the same as the secant algorithm with a success rate of seven, but the accumulated error for secant method is less. For Liver Disorder data set the secant method has better

Table 6.4: Results for Heart Disease and Liver Disorders data sets

k	f_{opt}	MS k-means			GKM			Secant		
		E	α	t	E	α	t	E	α	t
Heart Disease										
2	$0.59890 \cdot 10^6$	0.00	7.86	0.16	0.00	0.505	0.02	0.00	5.78	0.03
5	$0.32797 \cdot 10^6$	0.00	29.7	0.47	0.52	0.722	0.02	0.51	30.0	0.12
10	$0.20222 \cdot 10^6$	2.76	80.1	0.84	0.00	1.57	0.03	1.93	89.4	0.37
15	$0.14771 \cdot 10^6$	8.79	113	1.14	0.00	2.68	0.06	0.69	179	0.73
20	$0.11778 \cdot 10^6$	7.46	130	1.19	0.00	3.98	0.09	1.37	267	1.10
25	$0.10213 \cdot 10^6$	5.16	151	1.31	0.48	5.46	0.11	0.00	404	1.64
30	$0.88795 \cdot 10^5$	18.66	180	1.64	0.00	6.80	0.14	0.31	533	2.15
40	$0.68645 \cdot 10^5$	28.65	213	1.67	1.71	9.71	0.20	-0.84	868	3.51
50	$0.55894 \cdot 10^5$	33.68	250	1.88	2.06	13.2	0.27	0.20	1257	5.14
Liver Disorders										
2	$0.42398 \cdot 10^6$	0.00	6.91	0.09	93.96	0.600	0.00	0.00	3.81	0.16
5	$0.21826 \cdot 10^6$	0.00	41.7	0.42	0.08	0.990	0.03	0.08	19.9	0.06
10	$0.12768 \cdot 10^6$	0.09	87.5	0.67	0.00	2.00	0.05	0.05	63.4	0.18
15	$0.97474 \cdot 10^5$	6.53	147	0.92	1.62	3.41	0.08	0.73	119	0.34
20	$0.81820 \cdot 10^5$	9.05	184	1.11	0.29	5.12	0.11	0.93	176	0.50
25	$0.70419 \cdot 10^5$	16.64	208	1.17	0.23	6.99	0.13	0.18	244	0.71
30	$0.61143 \cdot 10^5$	24.33	229	1.31	0.21	8.75	0.16	-0.61	326	0.98
40	$0.47832 \cdot 10^5$	37.83	290	1.61	3.59	14.6	0.23	1.78	515	1.60
50	$0.39581 \cdot 10^5$	50.64	337	1.88	5.50	19.9	0.28	-0.21	761	2.42

performance with higher success rate and accumulated error 2.92 versus 105.48.

One important aspect in Table 6.4 is that the secant method could find the global solution better than the result reported [10], in one situation in Heart disease data set and two times in Liver disorder data set. This accuracy does not make a big difference in CPU time, but it needs more norm calculations.

In Ionosphere and Congressional Voting Records data sets the MS k-means algorithm again cannot find the global solution when the number of clusters $k > 5$ (see Table 6.5). For these date sets, the performance of GKM is similar to MS k-means, but the accumulation error is better for GKM. The secant method was successful nine times on Ionosphere and seven times on Congressional Voting Records. Even though, the accumulated error for Ionosphere is negative, it is small for Congressional Voting Records. Another evidence for superiority of secant method is that this method could find global solution four times in Ionosphere and three times in Congressional Voting Records data sets better than reported solution in [10].

Table 6.5: Results for Ionosphere and Congressional Voting Records data sets

k	f_{opt}	MS k-means			GKM			Secant		
		E	α	t	E	α	t	E	α	t
Ionosphere										
2	$0.24194 \cdot 10^4$	0.00	5.75	0.45	0.00	0.663	0.03	0.00	13.5	0.14
5	$0.18915 \cdot 10^4$	0.00	26.2	0.70	0.07	0.899	0.05	0.18	80.8	0.93
10	$0.15694 \cdot 10^4$	1.02	67.3	1.88	1.73	1.40	0.08	-0.33	243	2.45
15	$0.14014 \cdot 10^4$	3.72	104	2.47	4.31	1.88	0.11	0.05	419	4.14
20	$0.12714 \cdot 10^4$	2.62	136	3.05	5.73	2.53	0.13	-0.32	638	6.42
25	$0.11486 \cdot 10^4$	11.95	182	4.02	6.76	3.35	0.16	-0.26	936	9.85
30	$0.10469 \cdot 10^4$	13.99	200	4.19	7.37	4.35	0.20	-0.46	1240	14.1
40	$0.85658 \cdot 10^3$	30.35	273	5.59	7.82	7.88	0.30	0.46	2063	24.3
50	$0.70258 \cdot 10^3$	45.90	352	6.72	6.63	11.1	0.38	0.15	3069	36.7
Congressional Voting Records										
2	$0.16409 \cdot 10^4$	0.00	7.77	0.28	0.12	1.00	0.02	0.12	9.12	0.06
5	$0.13371 \cdot 10^4$	0.00	37.5	0.39	1.02	1.60	0.05	1.01	53.6	0.29
10	$0.11312 \cdot 10^4$	1.12	95.8	1.48	1.33	2.84	0.08	0.56	171	1.00
15	$0.10089 \cdot 10^4$	1.42	134	1.73	0.00	4.72	0.13	-0.36	380	2.01
20	$0.91445 \cdot 10^3$	6.11	174	2.30	1.40	6.25	0.17	0.32	627	3.18
25	$0.85032 \cdot 10^3$	5.87	209	2.38	2.03	7.55	0.22	-0.01	855	4.57
30	$0.78216 \cdot 10^3$	12.31	238	2.73	2.73	10.1	0.27	1.85	1144	5.68
40	$0.69412 \cdot 10^3$	18.36	291	3.20	3.32	15.2	0.38	0.27	1939	9.65
50	$0.62451 \cdot 10^3$	25.72	351	3.69	4.35	19.9	0.48	-0.56	2860	14.5

Results from Table 6.6 show that the MS k-means algorithm cannot find the global solution when the number of clusters $k > 5$ in Breast Cancer data set and when $k > 10$ in Pima Indians Diabetes data set. For breast cancer, secant method located better results considering both success rate and accumulated error. For Pima Indians Diabetes data set the success rates are close but the accumulated error result is better for secant method.

The MS k-means algorithm cannot find the global solution when the number of clusters $k > 10$ for TSPLIB1060 data set and $k > 2$ for Image Segmentation data set (Table 6.7). For TSPLIB1060 data set the GKM algorithm does the same as the secant algorithm two times, it does two times better and five times worse than the secant algorithm. For Image Segmentation data set the GKM algorithm does the same as the secant algorithm five times and it does two times better and two times worse than the secant algorithm. Again the GKM algorithm requires less computational efforts than two other algorithms.

Table 6.6: Results for Breast Cancer and Pima Indians Diabetes data sets

k	f_{opt}	MS k-means			GKM			Secant		
		E	α	t	E	α	t	E	α	t
Breast Cancer										
2	$0.19323 \cdot 10^5$	0.00	0.891	0.38	0.00	0.242	0.05	0.00	1.52	0.06
5	$0.13705 \cdot 10^5$	0.00	8.50	1.30	2.28	0.306	0.09	1.86	7.90	0.32
10	$0.10216 \cdot 10^5$	4.40	15.8	1.47	0.00	0.559	0.17	0.02	22.4	0.84
15	$0.87813 \cdot 10^4$	0.20	24.2	1.91	0.00	0.803	0.23	-0.17	39.8	1.45
20	$0.77855 \cdot 10^4$	5.99	34.0	2.45	1.80	1.06	0.31	-0.06	62.2	2.21
25	$0.69682 \cdot 10^4$	9.87	40.6	2.66	4.12	1.27	0.38	0.79	85.6	3.01
30	$0.64415 \cdot 10^4$	10.44	49.3	3.23	3.43	1.63	0.45	0.72	111	3.90
40	$0.56171 \cdot 10^4$	15.99	61.7	3.77	3.70	2.22	0.61	0.28	175	6.04
50	$0.49896 \cdot 10^4$	22.37	74.2	4.27	4.21	3.03	0.77	-1.95	252	8.59
Pima Indians Diabetes										
2	$0.51424 \cdot 10^7$	0.00	2.30	1.13	0.00	0.318	0.06	0.00	3.42	0.14
5	$0.17370 \cdot 10^7$	0.00	10.6	1.58	0.14	0.440	0.13	0.14	12.0	0.50
10	$0.94436 \cdot 10^6$	0.00	30.4	2.75	0.36	0.646	0.20	0.36	27.2	1.04
15	$0.69725 \cdot 10^6$	2.30	46.5	3.73	0.00	1.06	0.30	0.03	47.1	1.73
20	$0.57438 \cdot 10^6$	3.50	56.1	3.94	0.00	1.53	0.39	0.36	74.2	2.64
25	$0.49058 \cdot 10^6$	5.75	66.5	4.61	0.00	2.20	0.52	-0.13	101	3.53
30	$0.43641 \cdot 10^6$	10.65	77.2	5.28	1.84	2.53	0.59	0.09	135	4.65
40	$0.36116 \cdot 10^6$	13.77	106	6.61	0.00	4.02	0.83	0.07	207	7.12
50	$0.31439 \cdot 10^6$	20.16	120	7.09	0.24	5.31	1.06	-0.34	287	9.84

For TSPLIB1060 data set the MS k-means algorithm finds the best known (or near best known) solutions two times, the GKM algorithm five times and the secant algorithm six times. For Image Segmentation data set the MS k-means algorithm finds such solutions only once, the GKM algorithm six times and the secant algorithm five times. In summary for these two data sets, secant method is almost better from success rate and accumulated error point of view.

The MS k-means algorithm again cannot find the global solution when the number of clusters $k > 10$ for TSPLIB3038 data set and $k \geq 2$ for Page Blocks data set (Table 6.8). For TSPLIB3038 data set the GKM algorithm was successful in four situations where the secant method in seven situations. Also, the accumulated error for the secant method is better. For page blocks data set, the results of GKM and secant methods are close; however, the accumulated error for the secant method is better.

The Table 6.9 represents the summary of information in the previous seven tables. It contains the success rates and accumulated errors. It is important to note that, the

Table 6.7: Results for TSPLIB1060 and Image Segmentation data sets

k	f_{opt}	MS k-means			GKM			Secant		
		E	α	t	E	α	t	E	α	t
TSPLIB1060										
2	$0.98319 \cdot 10^{10}$	0.00	1.71	0.75	0.00	0.580	0.08	0.00	2.61	0.06
10	$0.17548 \cdot 10^{10}$	0.05	53.1	2.36	0.23	1.45	0.36	0.23	30.3	0.71
20	$0.79179 \cdot 10^{9}$	8.74	93.9	2.78	1.88	2.96	0.69	0.41	74.5	1.54
30	$0.48125 \cdot 10^{9}$	4.91	123	3.14	3.34	4.70	1.03	0.34	125	2.71
40	$0.35312 \cdot 10^{9}$	8.23	141	3.48	1.14	6.45	1.38	0.00	185	4.28
50	$0.25551 \cdot 10^{9}$	21.17	167	3.95	3.10	8.92	1.73	2.53	244	5.54
60	$0.20443 \cdot 10^{9}$	22.11	199	4.58	0.72	11.3	2.08	0.00	321	6.84
80	$0.13535 \cdot 10^{9}$	33.51	251	5.47	0.00	17.2	2.80	0.06	465	10.4
100	$0.10041 \cdot 10^{9}$	52.12	281	5.94	0.10	22.7	3.53	-0.02	648	14.0
Image Segmentation										
2	$0.35606 \cdot 10^{8}$	0.00	6.49	11.59	0.00	2.71	1.06	0.00	44.7	0.03
10	$0.97952 \cdot 10^{7}$	2.25	80.3	15.95	1.76	3.67	3.97	1.76	803	52.5
20	$0.51283 \cdot 10^{7}$	14.06	188	20.58	0.09	6.36	7.58	1.49	2404	151
30	$0.35076 \cdot 10^{7}$	14.52	270	23.83	0.06	12.5	11.36	0.06	4892	302
40	$0.27398 \cdot 10^{7}$	21.56	339	26.59	1.25	17.1	16.67	1.26	6879	419
50	$0.22249 \cdot 10^{7}$	27.33	423	30.55	2.41	22.8	18.73	2.47	8558	521
60	$0.19095 \cdot 10^{7}$	35.21	493	33.33	0.00	29.7	22.50	0.82	10142	619
80	$0.14440 \cdot 10^{7}$	45.87	659	39.47	0.93	45.9	30.19	0.14	13108	795
100	$0.11512 \cdot 10^{7}$	50.03	805	45.17	0.92	63.8	38.00	-0.17	15897	955

secant method could improve the global solution in 18 occasions, as reported in [10].

The MS k-means algorithm is efficient to find global solutions to minimum sum-of-squares clustering problems when the number of clusters $k \leq 5$. The solutions obtained by the MS k-means algorithm differ significantly from the global solution as the number of clusters increases and it is not an efficient algorithm to solve clustering problems with even relatively large number of clusters

Three algorithms, considered in this Chapter, are different versions of the k-means algorithm. Their main difference is in the way they compute starting points. In the MS k-means algorithm starting points are chosen randomly, however in two other algorithms special schemes are applied to find them. In the clustering algorithm based on the secant method, we solved the nonsmooth problem of finding the next cluster center by applying the secant method. Results of numerical experiments show that the secant algorithm is more effective than two other algorithms at finding good starting points. The GKM algorithm achieves considerably better results than the

Table 6.8: Results for TSPLIB3038 and Page Blocks data sets

k	f_{opt}	MS k-means			GKM			Secant		
		E	α	t	E	α	t	E	α	t
TSPLIB3038										
2	$0.31688 \cdot 10^{10}$	0.00	0.860	12.97	0.00	0.469	1.38	0.00	2.00	0.42
10	$0.56025 \cdot 10^{9}$	0.00	14.2	11.52	2.78	0.857	8.41	0.58	19.9	3.92
20	$0.26681 \cdot 10^{9}$	0.42	37.1	14.53	2.00	1.60	16.63	0.48	45.8	9.03
30	$0.17557 \cdot 10^{9}$	1.16	57.8	19.09	1.45	2.97	25.00	0.64	75.7	13.7
40	$0.12548 \cdot 10^{9}$	2.24	74.6	22.28	1.35	3.98	33.23	1.04	107	19.3
50	$0.98400 \cdot 10^{8}$	2.60	84.5	23.55	1.19	5.26	41.52	2.58	144	25.9
60	$0.82006 \cdot 10^{8}$	5.56	103	27.64	0.00	6.39	49.75	-0.50	180	34.1
80	$0.61217 \cdot 10^{8}$	4.84	119	30.02	0.00	9.56	66.42	0.38	264	49.0
100	$0.48912 \cdot 10^{8}$	5.99	138	33.59	0.59	12.9	83.16	0.29	365	69.3
Page Blocks										
2	$0.57937 \cdot 10^{11}$	0.24	1.82	577.05	0.24	1.50	8.19	0.00	8.03	3.98
10	$0.45662 \cdot 10^{10}$	206.38	42.3	168.45	0.80	1.66	49.62	0.00	107	54.8
20	$0.17139 \cdot 10^{10}$	70.44	259	367.39	0.00	2.30	92.30	0.19	258	121
30	$0.94106 \cdot 10^{9}$	399.77	452	417.28	0.75	3.15	132.41	0.00	551	330
40	$0.62570 \cdot 10^{9}$	485.89	641	477.88	0.17	4.22	172.13	0.00	841	440
50	$0.42937 \cdot 10^{9}$	725.19	760	503.03	0.04	5.86	212.27	0.00	1127	610
60	$0.31185 \cdot 10^{9}$	1057.99	920	571.77	0.00	10.1	254.88	0.33	1706	859
80	$0.20576 \cdot 10^{9}$	1647.96	889	513.25	1.46	14.2	334.36	0.51	2892	1299
100	$0.14545 \cdot 10^{9}$	998.80	796	443.64	0.00	20.5	415.19	0.10	4090	2024

MS k-means algorithm as the number of clusters increases.

There is no significant difference between the results of the GKM and the secant algorithms on small data sets. However, the GKM requires significantly less computational efforts. Additionally, the secant algorithm works better than the GKM algorithm for large data sets and for large number of clusters ($k \geq 25$). The secant algorithm is especially effective for data sets such as Ionosphere, Congressional Voting Records, Liver Disorders data sets, which do not have well separated clusters.

The secant algorithm outperforms considerably the MS k-means algorithm for all data sets and the number of clusters $k > 5$. The algorithm produces, as a rule, better solutions than the GKM algorithm in many cases. For Ionosphere data set ($k \geq 15$), Breast Cancer data set ($k \geq 25$), Congressional Voting Records data set ($k \geq 30$) and Liver Disorders data set ($k \geq 40$), solutions obtained by the secant algorithm are significantly better than those obtained by the GKM algorithm. However, the GKM algorithm takes less CPU time and uses less computational efforts than the other two

Table 6.9: The summary of numerical results

Data sets	MS k-means		GKM		Secant	
	$\sum E$	s.r	$\sum E$	s.r	$\sum E$	s.r
German towns	23.8	6	2.64	8	4.72	7
Bavaria postal 1	99.88	6	9.25	7	1.51	8
Bavaria postal 2	114.97	3	1.86	8	1.86	8
Fisher's Iris Plant	25.47	5	13.32	4	13.32	4
Heart Disease	105.16	2	4.77	7	4.16	7
Liver Disorders	145.11	3	105.48	5	2.92	8
Ionosphere	109.55	2	40.42	2	-0.54	9
Congressional Voting Records	70.91	2	16.3	2	3.22	7
Breast Cancer	69.26	3	19.54	3	1.52	8
Pima Indians Diabetes	56.13	3	2.58	8	0.59	9
TSPLIB1060	150.79	2	10.51	5	6.54	8
Image Segmentation	210.83	1	7.42	6	7.82	5
TSPLIB3038	22.81	3	9.36	4	5.52	8
Page Blocks	5592.66	1	3.42	8	1.13	9

algorithms.

6.5 Conclusion

In this chapter, the cluster analysis problem, related literature review and the results of different kind of algorithms have been presented. We formulated it as a nonsmooth nonconvex optimization problem. To minimize the nonsmooth objective function, we applied secant method which is introduced in Chapter 4. To show the numerical competence, we presented the results of the secant method and compared with the results of MS k-means algorithm and global k-means algorithm which was reported in [10]. The numerical results confirmed superiority of the secant method over other two algorithms.

Conclusion and Future Research

This thesis investigates the optimization of nonsmooth nonconvex functions. To avoid having special assumptions about the objective function such as convexity or smoothness, we have been concerned with locally Lipshitz functions. Despite existence of efficient methods for nonsmooth convex problems such as bundle methods and its variants, there is no effective method for nonsmooth and nonconvex problems. Although, we encountered some metaheuristic and hybrid methods to solve such problems, they demand huge amount of time and computation to locate a suboptimal solution.

The significance of the study originates from the frequency of appearance of nonsmooth nonconvex problems in many areas of research and application. Firstly, many practical applications exist where the differentiability assumption is no longer valid. This fact makes it impossible to apply traditional gradient based optimization methods on nonsmooth problems. However, even for many problems with differentiability assumption, the existence of errors in calculation may lead to a false solution. These considerations insist the role of methods which are able to find the optimal point without any dependence on the derivative information. Secondly, in some problems, the global solution for the problem is in interest. For these problems, a local solution would neglect some important aspects of the design or arrangement. Nevertheless, for many traditional methods, they are only able to find the first local solution; they get stuck there, and could not jump from the trap to calculate better solutions. The problem of finding the global optimal point versus local optimal points pertains to the nonconvexity nature of the problem under consideration.

In this thesis, we have introduced four new methods for nonsmooth nonconvex problems. In addition to the theoretical proof of their convergence, they have been tested on standard test problems. The numerical experiments confirm their efficiency in finding optimal solutions.

Contribution of the Study

As stated above, the design of derivative free methods for nonsmooth optimization which are applicable for nonconvex problems is important. Toward this aim, we designed some derivative free methods, and using numerical experiments, we showed their efficiency for nonsmooth nonconvex problems. Also, we applied them to solve some cluster analysis problems, which indicates their ability in dealing with large scale problems. Moreover, using one of these methods, we developed a new incremental algorithm for cluster analysis. Here is the list of designed algorithms with short explanations:

1. Approximate subgradient method

   The approximate subgradient method is a new algorithm for minimizing locally Lipschitz functions. This algorithm is as simple to implement as the subgradient method, and at the same time it is numerically efficient. Descent directions in this algorithm are computed by solving a system of linear inequalities. The convergence of the algorithm is proved for quasidifferentiable semismooth functions.

2. Secant method

   The secant method is a new algorithm to locally minimize nonsmooth, nonconvex functions. The notion of secants for nonsmooth functions is introduced. The secants are applied to find descent directions of locally Lipschitz functions. Then, the latter is used to design a minimization algorithm. It is proved that the iterates of this algorithm converges to Clarke stationary points. Also, we extended the method to a global search method and called it the global secant method.

3. Quasi secant method

   The notions of quasi secants is introduced and a minimization method is designed based on them. The quasi secants have the ability to exploit more global properties and information of the function than the subgradients.

4. New algorithm for the clustering

   Using the secant method, we developed a new incremental algorithm to solve

the clustering problem. In this method, at each step, the nonsmooth nonconvex problem of finding a starting points for cluster centers is solved using the secant method.

Future Study

The numerical results show that the proposed methods are efficient in finding optimal solution in comparison with some existing methods. However, we could improve performance of the proposed methods by doing more research in the following directions:

- Finding descent direction

  Generally in most algorithms for nonsmooth optimization, the descent direction is found by solving a quadratic problem which is expensive. This component, which is frequently called by the algorithm, plays important role in the CPU time and number of function evaluations. In approximate subgradient method, we tried to find descent direction solving a system of linear inequalities. However, more research is necessary to do in this direction to design more efficient methods.

- Large scale problems

  We successfully applied the methods on some large scale nonsmooth problems. However, according to the rapid development in science and technology, larger problems are emerging which require especial attention in developing new methods.

- Constrained optimization

  Our focus in the thesis was about unconstrained optimization problems. Although, we could convert the constrained optimization problems in unconstrained problems, this needs setting some parameters which make the situation difficult. In this regard, it would be convenient to do further research to extent the designed methods into constrained problems as well.

Bibliography

[1] Uci repository of machine learning databases. http://www.ics.uci.edu/mlearn/MLRepository.html.

[2] Cheolhong A. and Nguyen T.Q. Iterative rate-distortion optimization of h.264 with constant bit rate constraint. *IEEE Transactions on Image Processing*, 17(9):1605–1615, 2008.

[3] Frangioni A. Solving semidefinite quadratic problems within nonsmooth optimization algorithms. *Computers & Operations Research*, 23(11):1099–1118, 1996.

[4] Frangioni A. Generalized bundle methods. *SIAM Journal on Optimization*, 13(1):117–156, 2002.

[5] Ruszczynski A. A merit function approach to the subgradient method with averaging. *Optimization Methods Software*, 23(1):161–172, 2008.

[6] Eberhard A.C. and Wenczel R. A study of some optimality conditions in nonsmooth analysis. ICOTA, University of Ballarat, page 91, 2004.

[7] Golikov A.I. and Evtushenko Yu.G. A new method for solving systems of linear equalities and inequalities. *Doklady Mathematics*, 64(3):370–373, 2001.

[8] Bagirov A.M. A method for minimization of quasidifferentiable functions. *Optimization Methods and Software*, 17(1):31–60, 2002.

[9] Bagirov A.M. Continuous subdifferential approximations and their applications. *Journal of Mathematical Sciences*, 115(5):2567–2609, 2003.

[10] Bagirov A.M. Modified global k-means algorithm for minimum sum-of-squares clustering problems. *Pattern Recogn.*, 41(10):3192–3199, 2008.

[11] Bagirov A.M. and Rubinov A.M. Cutting angle method and a local search. *Journal of Global Optimization*, 27(2-3):193–213, 2003.

[12] Bagirov A.M. and Ugon J. Piecewise partially separable functions and a derivative-free algorithm for large scale nonsmooth optimization. *Journal of Global Optimization*, 35(2):163–195, 2006.

[13] Bagirov A.M. and Yearwood J. A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *European Journal of Operational Research*, 170(2):578–596, 2006.

[14] Bagirov A.M. and Yearwood J. A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *European Journal of Operational Research*, 170(2):578–596, 2006.

[15] Rubinov A.M. *Abstract Convexity and global Optimization*, volume 44 of *Nonconvex Optimization and its Application.* Kluwer Academic Publishers, Dordrecht, 2000.

[16] Conn A.R. and Toint Ph.L. *An Algorithm using Quadratic Interpolation for Unconstrained Derivative Free Optimization*, chapter Nonlinear Optimization and Applications, pages 27–47. Plenum Publishing, New York, 1996.

[17] Hedar A.R. *Studies on Metaheuristics for Continuous Global Optimization Problems.* PhD thesis, Kyoto University, Japan, 2004.

[18] Hedar A.R. and Fukushima M. Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization. *ptimization Methods and Software*, 17(1):891–912, 2002.

[19] Lewis A.S. *Local structure and algorithms in nonsmooth optimization*, chapter Optimization and Applications, page 104106. Mathematisches Forschungsinstitut Oberwolfach, Oberwolfach, Germany, 2005.

[20] Borchers B. and Mitchell. J.E. An improved branch and bound algorithm for mixed integer nonlinear programs. *Computers and Operation research*, 21(4):359–367, 1994.

[21] Fogel D.B. Baeck T. and Michalewicz Z., editors. *Evolutionary Computation 2: Advanced Algorithms and Operators.* Institute of Physics Publishing, 2000.

[22] A. M. Bagirov. Minimization methods for one class of nonsmooth functions and calculation of semi-equilibrium prices. *in: Progress in optimization: contributions from Australasia.*

[23] A. M. Bagirov and A. A. Gasanov. A method of approximating a quasidifferential. *Comput. Math. Math. Phys.*, 35(4):403–409, 1995.

[24] Ghosh M. Bagirov A.M. and Webb D. A derivative free method for linearly constrained nonsmooth optimization. *Journal of Industrial and Management Optimization*, 2(3):319–338, 2006.

[25] Rubinov A. Bagirov A.M. and Yearwood J. Using global optimization to improve classification for medical diagnosis and prognosis. *Topics in Health Information Management*, 22(1):65–74, 2001.

[26] Rubinov A.M. Bagirov A.M. and John Yearwood. A global optimization approach to classification. *Optimization and Engineering*, 3(2):129–155, 2002.

[27] Soukhoroukova N. Bagirov A.M., Rubinov A.M. and Yearwood J. Unsupervised and supervised data classification via nonsmooth and global optimization. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 11:1–93, 2003.

[28] Karasozen B. Bagirov M. and Sezer M. Discrete gradient method: Derivative-free method for nonsmooth optimization. *Journal of Optimization Theory and Applications*, 137(2):317–334, 2008.

[29] Polyak B.T. A general method for solving extremum problems. *Soviet Mathematical Doklady*, 174(3):33–36, 1967.

[30] Polyak B.T. *Introduction to Optimization.* Translations Series in Mathematics and Engineering. Optimization Software, New York, 1987.

[31] Lewis A.S. Burke J.V. and Overton M.L. Approximating subdifferentials by random sampling of gradients. *Mathematics of Operations Research*, 27(3):567–584, 2002.

[32] Lewis A.S. Burke J.V. and Overton M.L. Two numerical methods for optimizing matrix stability. *Linear Algebra and its Applications*, 351-352(15):117–145, 2002.

[33] Lewis A.S. Burke J.V. and Overton M.L. Pseudospectral components and the distance to uncontrollability. *SIAM Journal on Matrix Analysis and Applications*, 26(2):350–361, 2005.

[34] Lewis A.S. Burke J.V. and Overton M.L. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005.

[35] Lewis A.S. Burke J.V., Henrion D. and M.L. Overton. Stabilization via nonsmooth, nonconvex optimization. *Automatic Control, IEEE Transactions*, 51(11):1760–1769, 2006.

[36] Gurfil P. Butnariu D., Censor Y. and Hadar E. On the behavior of subgradient projections methods for convex feasibility problems in euclidean spaces. *SIAM Journal on Optimization*, 19(2):786–807, 2008.

[37] Audet C. and Dennis J. E. Analysis of generalized pattern searches. *SIAM J. on Optimization*, 13(3):889–903, 2002.

[38] Audet C. and Dennis J. E. A pattern search filter method for nonlinear programming without derivatives. *SIAM J. on Optimization*, 14(4):980–1010, 2004.

[39] Audet C. and Dennis J. E. Mesh adaptive direct search algorithms for constrained optimization. *SIAM J. on Optimization*, 17(1):188–217, 2006.

[40] Davis C. Theory of positive linear dependence. *American Journal of Mathematics*, 76(4):733–746, October 1954.

[41] Lemaréchal C. *Bundle Methods in Nonsmooth Optimization*, volume 3 of *IIASA Proceedings*. Pergamon Press, 1977.

[42] Lemaréchal C. *Nonsmooth Optimization and descent methods.* International Institute for Applied Systems Analysis, 1978.

[43] Lemaréchal C. *undle Methods in Nonsmooth Optimization*, chapter Nonsmooth Optimization, pages 78–102. Pergamon Press, Oxford, 1978.

[44] Lemaréchal C. *Nondifferentiable Optimization*, chapter Nonlinear Optimization, pages 149–199. Birkhauser, Boston, 1980.

[45] Lemaréchal C. *Nondifferentiable Optimization*, volume I of *Optimization*, chapter Handbooks in Oprations Research and Management Sciences. North-Holland, 1989.

[46] Lemaréchal C. Lagrangian decomposition and nonsmooth optimization: Bundle algorithm, prox iteration, augmented lagrangian. In *Nonsmooth Optimization: Methods and Applications.* Gordon and Breach Science, 1991.

[47] Lemaréchal C. and Sagastizábal C. Practical aspects of the moreau–yosida regularization: Theoretical preliminaries. *SIAM Journal on Optimization*, 7(2):367–385, 1997.

[48] Ribeiro C. *Essays and Surveys in Metaheuristics.* Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[49] Sagastizábal C. and Solodov M. An infeasible bundle method for nonsmooth convex constrained optimization without a penalty function or a filter. *SIAM Journal on Optimization*, 16(1):146–169, 2005.

[50] Scheinberg K. Conn A.R. and Toint Ph.L. n the convergence of derivative free methods for unconstrained optimization. *Approximation theory and optimization: Tributes to M.J.D. Powell*, pages 83–108, 1997.

[51] Scheinberg K. Conn A.R. and Toint Ph.L. *Trust-Region Methods (MPS-SIAM Series on Optimization).* SIAM, Philadelphia, 2000.

[52] Reeves C.R., editor. *Modern heuristic techniques for combinatorial problems.* John Wiley & Sons, Inc., New York, NY, USA, 1993.

[53] Marquardt D. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[54] Vanderbilt D. and Louie S.G. A monte carlo simulated annealing approach to optimization over continuous variables. *Journal of Computational Physics*, 56(2):259–271, 1984.

[55] Winfield D. *Function and functional optimization by interpolation in data tables.* PhD thesis, Harvard University, Cambridge , MA, 1969.

[56] Winfield D. Function minimization by interpolation in a data table. *IMA Journal of Applied Mathematics*, 12(3):339–347, 1973.

[57] Brown D.E. and Huntley C.L. A practical application of simulated annealing to clustering. *Pattern Recognition*, 25(4):401–412, 1992.

[58] Bertsekas D.P. *Nonlinear programming.* Athena Scientific, 2 edition, 1999.

[59] POLAK E. and ROYSET J.O. Algorithms for finite and semi-infinite min-max-min problems using adaptive smoothing techniques. *Journal of optimization theory and applications*, 119(3):421–457, 2003.

[60] Glover F. Future paths for integer programming and links to artificial intelligence. *Computers and operations Research*, 13(5):533–549, 1986.

[61] Glover F. Tabu search part i. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[62] Glover F. Tabu search part ii. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[63] Glover F. and Laguna M. *Tabu Search.* Kluwer Academic Press, MA, USA, 1997.

[64] Clarke F.H. *OPtimization and nonsmooth analysis.* Wiley-Interscience, New York, 1983.

[65] Diehr G. Evaluation of a branch and bound algorithm for clustering. *SIAM Journal Scientific and Statistical Computing*, 6(2):268–284, 1985.

[66] Reinelt G. Tsp-lib-a traveling salesman library. *ORSA Journal of Computing*, 3:319–350, 1991.

[67] Lera D. Gaviano M., Kvasov D.E. and Sergeyev Y.D. Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. *ACM Transactions on Mathematical Software (TOMS)*, 29(4):469–480, 2003.

[68] Box G.E.P. Evolutionary operation: A method for increasing industrial productivity. *Applied Statistics*, 6(2):81–101, 1957.

[69] Bilbro G.L. and Snyder W.E. Optimization of functions with many minima. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(4):840–849, 1991.

[70] Ravindran A. Gupta K.O. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, 31(12):1533–1546, 1985.

[71] Schramm H. and Zowe J. A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM Journal on Optimization*, 2(1):121–152, 1992.

[72] Spath H. *Cluster Analysis Algorithms*. Ellis Horwood Limited, Chichester, 1980.

[73] Szu H and Hartley R. Fast simulated annealing. In *AIP Conference Proceedings 151 on Neural Networks for Computing*, pages 420–425, Woodbury, NY, USA, 1987. American Institute of Physics Inc.

[74] Miettinen K. Haarala N. and Mäkelä M.M. New limited memory bundle method for large-scale nonsmooth optimization. *Optimization Methods & Software*, 19(6):973–692, 2004.

[75] Miettinen K. Haarala N. and Mäkelä M.M. Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Mathematical Programming*, 109(1):181–205, 2007.

[76] Cheung B.K. Hansen P., Ngai E. and Mladenovic N. Analysis of global k-means, an incremental heuristic for minimum sum-of-squares clustering. *Journal of Classification*, 2.

[77] Mladenović N. Hansen P. and Perez-Britos D. Variable neighborhood decomposition search. *Journal of Heuristics*, 7(4):335–350, 2001.

[78] Bock H.H. *Automatische Klassifikation.* Vandenhoeck & Ruprecht, Gottingen, 1974.

[79] Bock H.H. Clustering and neural networks. *Advances in Data Science and Classification*, pages 265–277, 1998.

[80] J.B. Hiriart-Urruty and Lemarechal C. *Convex Analysis and Minimization Algorithms I: Fundamentals (Grundlehren Der Mathematischen Wissenschaften)*, volume I and II. Springer, Berlin and N.Y., October 1993.

[81] Muller M.W. Houkins D.M. and Ten Krooden J.A. *Cluster analysis.* Topics in Applied Multivariate Analysis. Cambridge University press, Cambridge, 1982.

[82] Osman I.H. and Kelly J.P. *Meta-Heuristics: Theory and Applications.* Kluwer Academic Publishers, Boston, MA, 1996.

[83] Pinter J. *Global Optimization in Action, Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications*, volume 6 of *Nonconvex Optimization and Its Applications.* Kluwer Academic Publishers, Dordrecht, 1996.

[84] Pinter J. *Handbook of Global Optimization.* Kluwer Academic Publishers, Dordrecht, 2003.

[85] Zowe J. Nondifferentiable optimization: A motivation and a short introduction into the subgradient and the bundle concept. *Computational Mathematical Programming*, 15:323–356, 1985.

[86] Nelder J.A. and Mead R. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[87] Murty M.N. Jain A.K. and Flynn P.J. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[88] Dennis J.E. and V. Torczon. Direct search methods on parallel machines. *SIAM Journal on Optimization*, 1(4):448–474, 1991.

[89] Holland J.H. Outline for a logical theory of adaptive systems. *Journal of the Association of Computing Machnery*, 9(3):297–314, 1962.

[90] Holland J.H. *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA, USA, 1992.

[91] Hiebert K. Solving systems of linear equations and inequalities. *SIAM Journal on Numerical Analysis*, 17(3):447–464, 1980.

[92] Levenberg K. A method for the solution of certain non-linear problems in least squares. *The Quarterly of Applied Mathematics*, 2:164168, 1944.

[93] Ortiz M. Kane C., Repetto E.A. and Marsden J.E. Finite element analysis of nonsmooth contact. *Computer Methods in Applied Mechanics and Engineering*, 180(1):1–26, 1999.

[94] Makela M. M. Karmitsa N. M. S. and Ali M. M. Limited memory interior point bundle method for large inequality constrained nonsmooth minimization. *Applied Mathematics and Computation*, 198(1):382–400, 2008.

[95] Kiwiel K.C. *Methods of Descent for Nondifferentiable Optimization*, volume 1133 of *Lecture Notes in Mathematics.* Springer-Verlag, Berlin, 1985.

[96] Kiwiel K.C. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, 46(1-3):105–122, 1990.

[97] Kiwiel K.C. Proximal level bundle methods for convex nondifferentiable optimization, saddle-point problems and variational inequalities. *Mathematical Programming*, 69(1-3):111–147, 1995.

[98] Kiwiel K.C. A proximal bundle method with approximate subgradient linearizations. *SIAM Journal on Optimization*, 16(4):1007–1034, 2006.

[99] Kiwiel K.C. A proximal-projection bundle method for lagrangian relaxation, including semidefinite programming. *SIAM Journal on Optimization*, 17(4):1015–1034, 2006.

[100] Kiwiel K.C. Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM Journal on Optimization*, 18(2):379–388, 2007.

[101] J. E. Kelley. The cutting plane method for solving convex programming. *Journal of the Society for Industrial and Applied Mathematics*, 8:703–712, 1960.

[102] Gelatt Jr. C.D. Kirkpatrick S. and Vecchi M.P. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[103] Tamara G. Kolda, Robert Michael Lewis, and Virginia Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45:385–482, 2003.

[104] Narendra P.M. Koontz W.L.G. and Fukunaga K. A branch and bound clustering algorithm. *IEEE Transactions on Computers*, 24(9):908–915, 1975.

[105] Al-Sultan K.S. A tabu search approach to the clustering problem. *Pattern Recognition*, 28(9):1443–1451, September 1995.

[106] Al-Sultan K.S. and Khan M.M. Computational experience on four algorithms for the hard clustering problem. *Pattern Recogn. Lett.*, 17(3):295–308, 1996.

[107] Vladimr Kvasnika and Ji Pospchal. A hybrid of simplex method and simulated annealing. *Chemometrics and Intelligent Laboratory Systems*, 39(2):161 – 173, 1997.

[108] Ingber L. Simulated annealing: Practice versus theory. *Mathematical Computer Modelling*, 18(12):29–57, 1993.

[109] Lukšan L. and Vlček J. A bundle-newton method for nonsmooth unconstrained minimization. *Mathematical Programming*, 83(3):373–391, 1998.

[110] Lukšan L. and Vlček J. Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical Report 78, Academy of Sciences of the Czech Republic, 2000.

[111] Lukšan L. and Vlček J. Algorithm 811: Nda: algorithms for nondifferentiable optimization. *ACM Transaction on Mathematical Software*, 27(2):193–213, 2001.

[112] C. Lemaréchal. *An extension of Davidon Methods to Nondifferentiable problems*, chapter Nondifferentiable Optimization, Mathematical Programming Study 3, pages 95–109. 1975.

[113] C. Lemaréchal. Combining kelley's and conjugate gradient methods. In *Abstract of IX International symposium on Mathematical Programming*, Budapest, Hungary, 1976.

[114] C. Lemaréchal. *Nondifferentiable optimization*, chapter Optimization. North Holland, Amsterdam, 1989.

[115] Nemirovski A. Lemaréchal C. and Nesterov Y. New version of bundle methods. *Mathematical programming*, 69:111–147, 1995.

[116] Vlassis M. Likas A. and Verbeek J. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461, 2003.

[117] Beliakov G. Lim K.F. and Batten L.M. Predicting molecular structures: application of the cutting angle method. *hysical Chemistry Chemical Physics*, 5:3884–3890, 2003.

[118] Akgul M. *Topics in Relaxation and Ellipsoidal Methods*, volume 97 of *Research Notes in Mathematics*. Pitman Advanced Pub. Program, 1984.

[119] Fukushima M. and Qi Liqun. A globally and superlinearly convergent algorithm for nonsmooth convex minimization. *SIAM Journal on Optimization*, 6(4):1106–1120, 1996.

[120] Gaudioso M. and Monaco M.F. A bundle type approach to the unconstrained minimization of convex nonsmooth functions. *Mathematical Programming*, 23(1):216–226, 1982.

[121] Minoux M. *Mathematical Programming: Theory and application*. John Wiley, New York, 1986.

[122] Pincus M. A monte carlo method for the approximate solution of certain types of constrained optimization problems. *OPERATIONS RESEARCH*, 18(6):1225–1228, 1970.

[123] Abramson M.A. and Audet C. Convergence of mesh adaptive direct search to second-order stationary points. *SIAM J. on Optimization*, 17(2):606–619, 2006.

[124] Jaumard B. Merle O. du, Hansen P. and Mladenovic N. An interior point algorithm for minimum sum-of-squares clustering. *SIAM Journal on Scientific Computing*, 21(4):1485–1505, 2000.

[125] Rosenbluth M. Teller A. Metropolis M., Rosenbluth A. and Teller E. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[126] Sun D. Mifflin R. and Qi L. Quasi-newton bundle-type methods for nondifferentiable convex optimization. *SIAM Journal on Optimization*, 8(2):583–603, 1998.

[127] Powell M.J.D. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and Numerical Analysis, Proceedings of the 6th workshop on Optimization and Numerical Analysis*, volume 275, pages 51–67, Oaxaca, Mexico, 1994. Kluwer Academic Publishers.

[128] Makela M.M. and Neittaanmaki P. *Nonsmooth Optimization- Analysis and Algorithms with Applications to Optimal Control.* World Scientific, Singapore, 1992.

[129] Tosset M.W. What is simulated annealing? *Optimization and Engineering*, 2(2):201–213, 2002.

[130] Shor N.Z. *Minimization Methods for nondifferentiable Functions.* Springer-Verlag, Berlin, 1985.

[131] Shor N.Z. *Nondifferentiable Optimization and Polynomial Functions.* Kluwer Academic Publishers, 1998.

[132] Hanjoul P. and Peeters D. A comparison of two dual-based procedures for solving the p-median problem. *European Journal of Operational Research*, 20(3):387–396, 1985.

[133] Hansen P. and Jaumard B. Cluster analysis and mathematical programming. *Mathematical Programming: Series A and B*, 79(1-3):191–215, 1997.

[134] Hansen P. and Mladenovic N. j-means: a new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, 34(2):405–413, 2001.

[135] Wolfe P.H. A method of conjugate subgradients for minimizing nondifferentiable functions. *Mtahematical programming study*, 3:145–173, 1975.

[136] Wolfe P.H. Finding the nearest point in a polytope. *Mathematical Programming*, 11(2):128–149, 1976.

[137] Conn A. R., Scheinberg K., and Vicente L.N. Geometry of interpolation sets in derivative free optimization. *Math programming*, 111(1):141–172, 2007.

[138] Dubes R. and Jain A.K. Clustering techniques: The user's dilemma. *Pattern Recognition*, 8(4):247 – 260, 1976.

[139] Fletcher R. *Practical Methods of Optimization.* John Wiley and Sons, Chichester, 2 edition, 1987.

[140] Hooke R. and Jeeves T.A. " direct search" solution of numerical and statistical problems. *Journal of the Association of Computing Machnery*, 8(2):212–229, 1961.

[141] Mifflin R. An algorithm for constrained optimization with semismooth functions. *Mathematics of Operations Research*, 2(2):191–207, 1977.

[142] Mifflin R. Semismooth and semiconvex functions in constrained optimization. *SIAM Journal Control and Optimization*, 15(6):959–972, 1977.

[143] Mifflin R. A quasi-second-order proximal bundle algorithm. *Mathematical Programming*, 73(1):51–72, 1996.

[144] Fisher R.A. The use of multiple measurements in taxonomic problems. *Annals Eugen.*, 7:179–188, 1936.

[145] Jensen R.E. A dynamic programming algorithm for cluster analysis. *Operations Research*, 17(6):1034–1057, 1969.

[146] Perez-Guerrero R.E. and Heydt G.T. Distribution system restoration via subgradient-based lagrangian relaxation. *IEEE Transactions on Power Systems*, 23(3):1162–1169, 2008.

[147] Dakin R.J. A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8(3):250–255, 1965.

[148] Cavalcante R.L.G. and Yamada I. Multiaccess interference suppression in orthogonal spacetime block coded mimo systems by adaptive projected subgradient method. *IEEE Transactions on Signal Processing*, 56(3):1028 – 1042, 2008.

[149] Lewis R.M. and Torczon V. Rank ordering and positive basis in pattern search algorithms. Technical report, Langley Research Center, 1996.

[150] Lewis R.M. and Torczon V. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, 9(4):1082–1099, 1999.

[151] Lewis R.M. and Torczon V. Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization*, 10(3):917–941, 1999.

[152] Lewis R.M. and Torczon V. A globally convergent augmented lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, 12(4):1075–1089, 2002.

[153] Rockaffellar R.T. Monotone operators and the proximal point algorithm. *SIAM journal on optimal Control and optimization*, 14(5):877–898, 1976.

[154] Leyffer S. Integrating sqp and branch-and-bound for mixed integer nonlinear programming. *Computational Optimization and Applications*, 18(3):295–309, 2001.

[155] Hext G.R. Spendley W. and Himsworth F.R. Sequential application of simplex designs in optimization and evolutionary operation. *Technometrics*, 4(441-461), 1962.

[156] Chiou Suh-Wen. An efficient search algorithm for road network optimization. *Applied Mathematics and Computation*, 201(1-2):128 – 137, 2008.

[157] Chiou Suh-Wen. A non-smooth model for signalized road network design problems. *Applied Mathematical Modelling*, 32(7):1179 – 1190, 2008.

[158] Chiou Suh-Wen. Optimization of congestion pricing road network with variable demands. *Applied Mathematics and Computation*, 195(2):382 – 391, 2008.

[159] Song X.H. Wang J.H. Sun L.X., Xie Y.L. and Yu R.Q. Cluster analysis by simulated annealing. *Computers and Chemistry*, 18:103–108, 1994.

[160] Selim S.Z. and Alsultan K. A simulated annealing algorithm for the clustering problem. *Pattern Recognition*, 24(10):1003–1008, 1991.

[161] Selim S.Z. and Ismail M.A. k-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):81–87, 1984.

[162] Baeck T., Fogel D.B., and Michalewicz Z. *Evolutionary Computation 1: Basic Algorithms and Operators (Evolutionary Computation)*. TF-TAYLOR, Institute of Physics Publishing, 2000.

[163] Chang T.S. Comments on surrogate gradient algorithm for lagrangian relaxation. *Journal of Optimization Theory and Applications*, 137(3):691–697, 2008.

[164] Torczon V. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997.

[165] Torczon V. and Trosset M.W. From evolutionary operation to parallel direct search: Pattern search algorithms for numerical optimization. *Computing Science and Statistics*, 29:396–401, 1988.

[166] Demyanov V.F. and Rubinov A.M. *Constructive Nonsmooth Analysis*, volume 7 of *Approximation & optimization*. Peter Lang (Frankfurt am Main, New York), 1995.

[167] Demyanov V.F. and Vasilev L.V. *Nondifferentiable Optimization*. Optimization Software, Incorporated, 1985.

[168] Ongsakul W. and Petcharaks N. Fast lagrangian relaxation for constrained generation scheduling in a centralized electricity market. *International Journal of Electrical Power & Energy Systems*, 30(1):46–59, 2008.

[169] Sun W. and Yuan Y. *Optimization Theory and Methods, Nonlinear Programming.* Springer, 2006.

[170] Zhao L. Wang Q., Qiao J. and Zou C. Acoustic feedback cancellation based on adaptive projection subgradient method in hearing aids. *Bioinformatics and Biomedical Engineering*, pages 2008 – 2011, 16-18 May 2008. The 2nd International Conference on ICBBE 2008.

[171] Bogju L. Yen J., Liao J.C. and Randolph D. A hybrid approach to modeling metabolic systems using a geneticalgorithm and simplex method. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(2):173–191, 1998.

[172] Liu Y. Yiu K.F.C. and Teo K.L. A hybrid descent method for global optimization. *Journal of Global Optimization*, 28(2):229–238, 2004.

[173] Ermoliev Yu.M. Methods for solving nonlinear extremal problems. *Kibernetika*, 4:1–17, 1966.

[174] Ermoliev Yu.M. *Methods of nondifferentiable and stochastic optimization and their aplications*, chapter Progress in nondifferentiable optimization, pages 5–27. International Institute for Aplied Systems Analysis, Lexenburg, Austra, 1982.

[175] Michalewicz Z. *Genetic algorithms + data structures = evolution programs.* Springer-Verlag, London, UK, 1996.

Appendix A

Test Functions

The efficiency of the proposed nonsmooth optimization algorithms was verified by applying it to some academic test problems with nonsmooth objective functions. We consider three types of problems:

1. Problems with nonsmooth convex objective functions;

2. Problems with nonsmooth nonconvex regular objective functions;

3. Problems with nonsmooth, nonconvex and nonregular objective functions.

Test Problems 2.1-7, 2.9-12, 2.14-16, 2.18-21 and 2.23-25 from [110] and Problems 1-3, 5 and 7 from [8] have been used in numerical experiments. We also include the following problem with nonsmooth, nonconvex and nonregular objective function.

Problem 1

$$\text{minimize} \quad f(x) = \sum_{i=1}^{p} \min_{j=1,\ldots,k} \|x^j - a^i\|^2$$

Here $p = 20,\ k = 5,\ x = (x^1, \ldots, x^5) \in \mathbb{R}^{15}$ and the vectors $a^i \in R^3,\ i = 1, \ldots, 20$ are as follows:

a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}
1.1	0.8	0.1	0.6	-1.2	0.9	0.2	-0.3	-0.8	0.0
1.0	-1.6	-1.0	0.2	1.0	1.9	0.2	-0.2	0.6	-0.4
-0.1	0.3	-0.3	0.2	1.4	-0.8	0.0	0.8	-0.2	0.6
a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}	a^{19}	a^{20}
1.0	0.0	0.0	2.1	0.2	-2.1	-1.0	0.3	1.1	3.1
0.0	1.0	0.0	-1.4	-1.0	0.0	0.5	-2.0	1.2	-1.5
0.0	0.0	1.0	1.0	1.0	-1.0	1.5	0.9	1.0	2.1

This is well known clustering function (see [27, 14]). We apply it to solve clustering problem on two real-world data sets: TSPLIB1060 and TSPLIB3038. The description of these data sets can be found in [66]. The first data set contains 1060 2-dimensional points and the second data set contains 3038 2-dimensional points. We compute 3, 5 and 10 clusters for each data set.

The brief description of test problems are given in Table A.1, where the following notation is used:

- n - number of variables;
- n_m - the total number of functions under maximum and minimum (if the function contains maximum and minimum functions);
- f_{opt} - optimum value.

The objective functions in Problems P27-P33 are the clustering function. In Problem P27 the number of clusters $k = 5$, the number of data points $p = 20$ and they are given in Table 1. In Problems P28-P30 data points are from the data set TSPLIB1060 and the number of clusters in Problem P28 is 3, in Problem P29 it is 5 and in Problem P30 10. In Problems P31-P33 data points are from the data set TSPLIB3038 and the number of clusters in Problem P31 is 3, in Problem P32 is 5 and in Problem P33 it is 10.

A.0.1 Global Optimization Test Functions

A recent procedure for generating test functions for multiextremal multidimensional box-constrained global optimization is proposed by Gaviano et.al [67]. This package has capability to generate continiously differentiable, differentiable and nondifferentiable test functions. Test functions are generated by defining a convex quadratic function systematically distorted by polynomials in order to introduce local minima. The user defines the following parameters along with the type of class: (i) problem dimension, (ii) number of local minima, (iii) value of the global minimum, (iv) radius of the attraction region of the global minimizer, (v) distance from the global minimizer to the vertex of the quadratic function. The most important information about these type of test functions are the dimension of the problem and number of local

minimas. We refer to these test functions as MDDY with specifying the dimension n and multimodality m.

A.0.2 Performance of Algorithms

To compare the performance of the algorithms, we use two indicators: n_b - the number of successful runs considering the best known solution and n_s - the number of successful runs considering the best found solution by these three algorithms. For Problems P3 and P19 algorithms found better solutions than those reported in [110]. We take these new solutions as the best solutions. Assume that f_{opt} and $\bar{f}$ are the values of the objective function at the best known solution and at the best found solution, respectively. Then, we say that an algorithm finds the best solution with respect to a tolerance $\varepsilon > 0$ if

$$\frac{f_* - f_0}{1 + |f_*|} \leq \varepsilon,$$

where f_* is equal either to f_{opt} (for n_b) or to $\bar{f}$ (for n_s), and f_0 is the optimal value of the objective function found by an algorithm. In our experiments $\varepsilon = 10^{-4}$.

Table A.1: The description of test problems

Function type	Problems	n	n_m	f_{opt}
Nonsmooth convex	P1 (Problem 2.1 [110])	2	3	1.9522245
	P2 (Problem 2.5 [110])	4	4	-44
	P3 (Problem 2.23 [110])	11	10	261.08258
Nonsmooth nonconvex regular	P4 (Problem 2.2 [110])	2	3	0
	P5 (Problem 2.3 [110])	2	2	0
	P6 (Problem 2.4 [110])	3	6	3.5997193
	P7 (Problem 2.6 [110])	4	4	-44
	P8 (Problem 2.7 [110])	3	21	0.0042021
	P9 (Problem 2.9 [110])	4	11	0.0080844
	P10 (Problem 2.10 [110])	4	20	115.70644
	P11 (Problem 2.11 [110])	4	21	0.0026360
	P12 (Problem 2.12 [110])	4	21	0.0020161
	P13 (Problem 2.14 [110])	5	21	0.0001224
	P14 (Problem 2.15 [110])	5	30	0.0223405
	P15 (Problem 2.16 [110])	6	51	0.0349049
	P16 (Problem 2.18 [110])	9	41	0.0061853
	P17 (Problem 2.19 [110])	7	5	680.63006
	P18 (Problem 2.20 [110])	10	9	24.306209
	P19 (Problem 2.21 [110])	20	18	133.72828
	P20 (Problem 2.24 [110])	20	31	0
	P21 (Problem 2.25 [110])	11	65	0.0480274
Nonsmooth nonconvex nonregular	P22 (Problem 1 [8])	2	6	2
	P23 (Problem 2 [8])	2	-	0
	P24 (Problem 3 [8])	4	-	0
	P25 (Problem 5 [8])	5	-	0
	P26 (Problem 7 [8])	5	-	0
	P27 (Problem 1)	15	100	13.311214
	P28 (Problem 1)	6	3180	6.32621×10^6
	P29 (Problem 1)	10	5300	3.57642×10^6
	P30 (Problem 1)	20	10600	2.13505×10^6
	P31 (Problem 1)	6	9114	7.16372×10^5
	P32 (Problem 1)	10	15190	3.94402×10^5
	P33 (Problem 1)	20	30380	1.84415×10^5