University of Turku

BUNDLE METHODS IN NONSMOOTH DC OPTIMIZATION

Kaisa Joki

TURUN YLIOPISTON JULKAISUJA – ANNALES UNIVERSITATIS TURKUENSIS
Sarja – ser. AI osa – tom. 585 | Astronomica – Chemica – Physica – Mathematica | Turku 2018

Turun yliopisto
University of Turku

BUNDLE METHODS IN NONSMOOTH DC OPTIMIZATION

Kaisa Joki

TURUN YLIOPISTON JULKAISUJA – ANNALES UNIVERSITATIS TURKUENSIS
Sarja - ser. A I osa - tom. 585 | Astronomica - Chemica - Physica - Mathematica | Turku 2018

University of Turku

Faculty of Science and Engineering

Department of Mathematics and Statistics

Doctoral Programme in Mathematics and Computer Sciences

Supervised by

Professor Marko Mäkelä
Department of Mathematics and Statistics
University of Turku
Turku, Finland

Professor Adil Bagirov
Faculty of Science and Technology
Federation University Australia
Ballarat, Australia

Docent Napsu Karmitsa
Department of Mathematics and Statistics
University of Turku
Turku, Finland

Reviewed by

Professor Hermann Schichl
Fakultät für Mathematik
Universität Wien
Wien, Austria

Professor Zhiyou Wu
School of Mathematical Sciences
Chongqing Normal University
Chongqing, China

Opponent

Doctor Annabella Astorino
Istituto di Calcolo e Reti ad Alte Prestazioni
Consiglio Nazionale delle Ricerche
Rende, Italy

ISBN 978-951-29-7276-0 (PRINT)
ISBN 978-951-29-7277-7 (PDF)
ISSN 0082-7002 (Print)
ISSN 2343-3175 (Online)
Painosalama Oy – Turku, Finland 2018

Abstract

Due to the complexity of many practical applications, we encounter optimization problems with nonsmooth functions, that is, functions which are not continuously differentiable everywhere. Classical gradient-based methods are not applicable to solve such problems, since they may fail in the nonsmooth setting. Therefore, it is imperative to develop numerical methods specifically designed for nonsmooth optimization. To date, bundle methods are considered to be the most efficient and reliable general purpose solvers for this type of problems.

The idea in bundle methods is to approximate the subdifferential of the objective function by a bundle of subgradients. This information is then used to build a model for the objective. However, this model is typically convex and, due to this, it may be inaccurate and unable to adequately reflect the behaviour of the objective function in the nonconvex case. These circumstances motivate to design new bundle methods based on nonconvex models of the objective function.

In this dissertation, the main focus is on nonsmooth DC optimization that constitutes an important and broad subclass of nonconvex optimization problems. A DC function can be presented as a difference of two convex functions. Thus, we can obtain a model that utilizes explicitly both the convexity and concavity of the objective by approximating separately the convex and concave parts. This way we end up with a nonconvex DC model describing the problem more accurately than the convex one. Based on the new DC model we introduce three different bundle methods. Two of them are designed for unconstrained DC optimization and the third one is capable of solving also multiobjective and constrained DC problems. The finite convergence is proved for each method. The numerical results demonstrate the efficiency of the methods and show the benefits obtained from the utilization of the DC decomposition.

Even though the usage of the DC decomposition can improve the performance of the bundle methods, it is not always available or possible to construct. Thus, we present another bundle method for a general objective function implicitly collecting information about the DC structure. This method is developed for large-scale nonsmooth optimization and its convergence is proved for semismooth functions. The efficiency of the method is shown with numerical results.

As an application of the developed methods, we consider the clusterwise linear regression (CLR) problems. By applying the support vector machines (SVM) approach a new model for these problems is proposed. The objective in the new formulation of the CLR problem is expressed as a DC function and a method based on one of the presented bundle methods is designed to solve it. Numerical results demonstrate robustness of the new approach to outliers.

Tiivistelmä

Monissa käytännön sovelluksissa tarkastelun kohteena oleva ongelma on monimutkainen ja joudutaan näin ollen mallintamaan epäsileillä funktioilla, jotka eivät välttämättä ole jatkuvasti differentioituvia kaikkialla. Klassisia gradienttiin perustuvia optimointimenetelmiä ei voida käyttää epäsileisiin tehtäviin, sillä epäsileillä funktioilla ei ole olemassa klassista gradienttia kaikkialla. Näin ollen epäsileään optimointiin on välttämätöntä kehittää omia numeerisia ratkaisumenetelmiä. Näistä kimppumenetelmiä pidetään tällä hetkellä kaikista tehokkaimpina ja luotettavimpina yleismenetelminä kyseisten tehtävien ratkaisemiseksi.

Ideana kimppumenetelmissä on approksimoida kohdefunktion alidifferentiaalia kimpulla, joka on muodostettu keräämällä kohdefunktion aligradientteja edellisiltä iteraatiokierroksilta. Tätä tietoa hyödyntämällä voidaan muodostaa kohdefunktiolle malli, joka on alkuperäistä tehtävää helpompi ratkaista. Käytetty malli on tyypillisesti konveksi ja näin ollen se voi olla epätarkka ja kykenemätön esittämään alkuperäisen tehtävän rakennetta epäkonveksissa tapauksessa. Tästä syystä väitöskirjassa keskitytään kehittämään uusia kimppumenetelmiä, jotka mallinnusvaiheessa muodostavat kohdefunktiolle epäkonveksin mallin.

Pääpaino väitöskirjassa on epäsileissä optimointitehtävissä, joissa funktiot voidaan esittää kahden konveksin funktion erotuksena (*difference of two convex functions*). Kyseisiä funktioita kutsutaan DC-funktioiksi ja ne muodostavat tärkeän ja laajan epäkonveksien funktioiden osajoukon. Tämä valinta mahdollistaa kohdefunktion konveksisuuden ja konkaavisuuden eksplisiittisen hyödyntämisen, sillä uusi malli kohdefunktiolle muodostetaan yhdistämällä erilliset konveksille ja konkaaville osalle rakennetut mallit. Tällä tavalla päädytään epäkonveksiin DC-malliin, joka pystyy kuvaamaan ratkaistavaa tehtävää tarkemmin kuin konveksi arvio. Väitöskirjassa esitetään kolme erilaista uuden DC-mallin pohjalta kehitettyä kimppumenetelmää sekä todistetaan menetelmien konvergenssit. Kaksi näistä menetelmistä on suunniteltu rajoitteettomaan DC-optimointiin ja kolmannella voidaan ratkaista myös monitavoitteisia ja rajoitteellisia DC-optimointitehtäviä. Numeeriset tulokset havainnollistavat menetelmien tehokkuutta sekä DC-hajotelman käytöstä saatuja etuja.

Vaikka DC-hajotelman käyttö voi parantaa kimppumenetelmien suoritusta, sitä ei aina ole saatavilla tai mahdollista muodostaa. Tästä syystä väitöskirjassa esitetään myös neljäs kimppumenetelmä konvergenssitodistuksineen yleiselle kohdefunktiolle, jossa kerätään implisiittisesti tietoa kohdefunktion DC-rakenteesta. Menetelmä on kehitetty erityisesti suurille epäsileille optimointitehtäville ja sen tehokkuus osoitetaan numeerisella testauksella.

Sovelluksena väitöskirjassa tarkastellaan datalle klustereittain tehtävää lineaarista regressiota (*clusterwise linear regression*). Kyseiselle sovellukselle muodostetaan uusi malli hyödyntäen koneoppimisessa käytettyä SVM-lähestymistapaa (*support vector machines approach*) ja saatu kohdefunktio esitetään DC-funktiona. Näin ollen yhtä kehitetyistä kimppumenetelmistä sovelletaan tehtävän ratkaisemiseen. Numeeriset tulokset havainnollistavat uuden lähestymistavan robustisuutta ja tehokkuutta.

Acknowledgements

The preparation of this dissertation has been a long but at the same time really rewarding journey. During this experience, there have been a great number of people who have influenced the outcome of this work and without whom I would not have been able to finish it. Therefore, I would like to take this opportunity to bestow my gratitude for the support of those people.

First of all, I want to thank my supervisors, Professor Marko Mäkelä, Docent Napsu Karmitsa and Professor Adil Bagirov, for their excellent guidance, valuable advice, and continuous support during the preparation of this dissertation. You have given me many opportunities and ideas to improve my research and to grow as a researcher. I am very grateful to Marko for supervising my master's thesis, which served as an introduction to nonsmooth optimization and bundle methods. Since then he has guided me through my doctoral studies and patiently answered all my questions. I also want to express my deep gratitude to Napsu, who has been an excellent mentor, always giving useful feedback and helping me whenever I have needed it. I also want to thank her for all of our research visits to meet Adil in Australia and for conference trips. You have been a great travel companion and offered me so many unforgettable experiences and opportunities, which I could never have dreamed of. In addition, I want to thank Marko and Napsu for creating such a warm, open and productive atmosphere at work. I am also very grateful for Adil. He introduced the field of DC optimization to me, and without his influence this dissertation would look very different. I want to thank him for proposing many good research topics and giving me the opportunity to visit his university in Australia three times. I have really enjoyed our productive conversations and collaboration over the years. Hopefully it will continue.

I thank all my co-authors, Professor Manlio Gaudioso, Doctor Sona Taheri and fellow PhD student Outi Montonen, for our fruitful collaboration. Special thanks to Manlio and Sona for their wonderful hospitality and guidance during my research visits to their universities in Italy and Australia. I also express my gratitude for Professor Hermann Schichl and Professor Zhiyou Wu for the time spent on reviewing this dissertation. In addition, I thank Doctor Annabella Astorino. It is an honour for me that she agreed to act as my opponent.

Next, I want to acknowledge the previous and current head of the department, Professor Juhani Karhumäki and Professor Iiro Honkala, for providing excellent working facilities and environment. I am also very grateful for the financial support of the Doctoral Programme in Mathematics and Computer Sciences (MATTI) of University of Turku, Department of Mathematics and Statistics, Jenny and Antti Wihuri Foundation, the Academy of Finland (Project No. 294002), Turku

University Foundation and Magnus Ehrnrooth Foundation. Without their support this research and networking with my foreign colleagues would not have been possible.

I am also indebted to the ex-administrative staff of our department, Tuire Huuskonen, Sonja Vanto and Lasse Forss, for helping me with the bureaucracy. Special thanks to Doctor Arto Lepistö for fast technical support, especially whenever my computer broke down. I also thank the current and previous researchers in Turku Optimization Group (TOpGroup). It has always been nice to learn what the others are doing in the field of optimization.

Then I wish to thank the staff members of the Department of Mathematics and Statistics. It has been a privilege to be part of such a pleasant and warm working environment. I have especially enjoyed our daily coffee breaks with discussions about the "urgent topics" of the day. Therefore, I wish to express my gratitude for each person I have gotten to know during my stay in the department. I want to especially thank Outi Montonen for sharing many ups and downs during my undergraduate and doctoral studies. It has been nice to share the interest in nonsmooth optimization with a fellow PhD student, and our joint research went unexpectedly smoothly in our nonsmooth world. Special thanks go to our innumerable mathematical and non-mathematical discussions from which many have been irreplaceable to complete this dissertation. I also thank Maria Jaakkola, Outi Montonen, Jarkko Peltomäki and Sari Yli-Sipilä for our get-togethers, which we started to organize since our undergraduate days. Special thanks also to Anni Hakanen, Outi Montonen and Jarkko Peltomäki for daily lunch meetings, which gave me needed breaks from daily mathematics.

Finally, I want to acknowledge my other close friends and family. First, I want to thank Eeva, Hilla, Ilona, Noora, Selmi and Vilma for our long friendships, which have lasted nearly half of my life. Then I want to express my gratitude to my great aunt Hanna and grandparents Maija and Urpo for their endless support and encouragement. For the most important support, I want to thank my parents Leena and Risto and my brother Mikko. Words cannot express how grateful I am for everything you have done on my behalf. Thank you for always believing in me, even at those times when I did not, and encouraging me through this experience.

Turku, April 2018 Kaisa Joki

List of original publications

This PhD dissertation is based on the following five original publications:

I. Joki, K., Bagirov, A. M., Karmitsa, N., and Mäkelä, M. M. A proximal bundle method for nonsmooth DC optimization utilizing nonconvex cutting planes. *Journal of Global Optimization 68*, 3 (2017), 501–535.

II. Joki, K., Bagirov, A. M., Karmitsa, N., Mäkelä, M. M., and Taheri, S. Double bundle method for finding Clarke stationary points in nonsmooth DC programming. *SIAM Journal on Optimization*, to appear (2018).

III. Montonen, O., and Joki, K. Bundle-based descent method for nonsmooth multiobjective DC optimization with inequality constraints. *Journal of Global Optimization* (2018). DOI 10.1007/s10898-018-0651-0

IV. Karmitsa, N., Gaudioso, M., and Joki, K. Diagonal bundle method with convex and concave updates for large-scale nonconvex and nonsmooth optimization. *Optimization Methods and Software* (2017). DOI 10.1080/10556788.2017.1389941

V. Joki, K., Bagirov, A. M., Karmitsa, N., Mäkelä, M. M., and Taheri, S. New bundle method for clusterwise linear regression utilizing support vector machines. *TUCS Technical Report No. 1190*, Turku Centre for Computer Science, Turku (2017).

These papers are included here with the permission of the publishers.

Contents

List of symbols

$\mathbb{R}^n$	n-dimensional Euclidean space
$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$	Column vectors
$\boldsymbol{x}^T$	Transposed vector
$\boldsymbol{x}^T\boldsymbol{y}$	Inner product of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$
$\|\boldsymbol{x}\|$	Norm of vector $\boldsymbol{x}$ in $\mathbb{R}^n$
$\{\boldsymbol{x}_i\}$	Sequence of vectors
$\mathbf{0}$	Zero vector
$c, m, \varepsilon, \delta$	Scalars
$t \downarrow 0$	$t \to 0_+$
$\boldsymbol{A}, \boldsymbol{D}$	Square matrices
$B(\boldsymbol{x}; \varepsilon)$	Open ball centred at $\boldsymbol{x}$ with a radius ε
$[a, b]$	Closed interval
S, U, X	Sets
cl S	Closure of set S
conv S	Convex hull of set S
ray S	Smallest cone containing S
cone S	Smallest convex cone containing S
I, J, L	Sets of indices
$\|J\|$	Number of elements in set J
$f(\boldsymbol{x})$	Objective function value at $\boldsymbol{x}$
f^1, f^2	Convex DC components
$f^1 - f^2$	DC decomposition
$\arg\min f(\boldsymbol{x})$	Point where function f attains its minimum value
$\nabla f(\boldsymbol{x})$	Gradient of function f at $\boldsymbol{x}$
$f'(\boldsymbol{x}, \boldsymbol{d})$	Directional derivative of function f at $\boldsymbol{x}$
$\partial_c f(\boldsymbol{x})$	Subdifferential of convex function f at $\boldsymbol{x}$
$\partial f(\boldsymbol{x})$	Subdifferential of function f at $\boldsymbol{x}$
$\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$	Subgradient of function f at $\boldsymbol{x}$
$\partial_\varepsilon f(\boldsymbol{x})$	ε-subdifferential of function f at $\boldsymbol{x}$
$\partial_\varepsilon^G f(\boldsymbol{x})$	Goldstein ε-subdifferential of function f at $\boldsymbol{x}$
$\hat{f}_k(\boldsymbol{x})$	Convex cutting plane model of function f at $\boldsymbol{x}$
$\tilde{f}_k(\boldsymbol{x})$	Nonconvex DC cutting plane model of function f at $\boldsymbol{x}$
max	Maximum
min	Minimum

DC	Difference of two convex functions
NSO	Nonsmooth optimization
SVM	Support vector machines
CLR	Clusterwise linear regression
PBDC	Proximal bundle method for DC optimization
DBDC	Double bundle method for DC optimization
MDBDC	Multiobjective double bundle method for DC optimization
SMDB	Splitting metrics diagonal bundle method
DB-SVM-CLR	Double bundle method for CLR problems

Part I

Summary

Chapter 1

Introduction

In optimization, the aim is to find the best possible solution fulfilling given conditions. The general optimization problem deals with minimizing or maximizing an objective function(s) under some constraints. The goal can be, for example, to maximize the production of a company, to minimize the driving time or to design an optimal shape for the wing of an aircraft. Since these types of mathematical problems appear everywhere in real life it is important to have tools for solving them. In addition, these problems are typically modelled with nonconvex functions since, even though convexity is a preferred feature in optimization, it is often a too restrictive assumption in practical applications. However, nonconvexity adds another challenge, since verifiable characterizations of global solutions do not exist and we face the difficult problem how to distinguish global solutions from local ones. For this reason, we need efficient numerical solution algorithms to provide solutions for complex problems.

In nonsmooth optimization (NSO), functions do not need to be continuously differentiable (or smooth) everywhere. These types of functions appear naturally in many practical applications when we construct a model of the problem. There exist also several different sources for nonsmoothness (see, e.g., [93]). For example, the original phenomenon itself can contain several discontinuities or we have a technological constraint causing nonsmoothness of functions. Some solution algorithms for optimization (e.g., the exact penalty function method) can translate the original smooth problem into a nonsmooth one. There exist also so-called stiff problems, which are analytically smooth, but numerically nonsmooth due to the too rapidly varying gradient. Applications of NSO include, for example, mechanics [100], machine learning [81], data mining [20], computational chemistry [39] and control theory [30].

Most solution algorithms are designed to solve smooth optimization problems and the most efficient of these methods utilize the derivative of the objective, since the opposite of the gradient is always a descent direction for a smooth function. Unfortunately, in NSO we face the situation, where a derivative does not exist for some values of the variables. Due to this, we cannot directly apply the classical optimization theory based on smoothness and the gradient based methods since they may fail in the nonsmooth setting. Instead, we rely on the subdifferential theory developed by Rockafellar [113] and Clarke [29] and use the so-called subgradients (or generalized gradients) instead of gradients. It is also worth noting

that, if we can determine the whole subdifferential (i.e., the set of all subgradients) for a nonsmooth function at a point, then a descent direction at that point is the opposite of the subgradient yielding the minimum norm in the subdifferential. Unfortunately, in practise the calculation of the whole subdifferential is often a too demanding or impossible task and we typically need to assume that only one arbitrary subgradient is known. However, the opposite of this subgradient does not need to be a descent direction.

One possible option to solve NSO problems is to use derivative free methods such as Powel's method (see, e.g., [111, 126]) and the Nelder-Mead method (see, e.g., [101, 105]). However, these methods are quite unreliable and the larger the problem is the more inefficient these methods become. Different kinds of smoothing or regularization techniques (see, e.g., [28, 47, 106]) are another possibility but, even though they work well in some special cases, they may not be efficient or even applicable in general. Therefore, we need methods specially designed for nonsmooth problems. In addition, nonsmoothness can be seen as an extension of the smooth case, and NSO methods can be applied to solve smooth problems.

There exist several methods for NSO problems, which are typically divided into two main classes: subgradient methods (see, e.g, [115]) and bundle methods (see, e.g., [59, 72, 93, 96, 114]). The basic assumption in these methods is that the objective function is locally Lipschitz continuous. These methods require also that at each point we can compute the value of the objective function and one arbitrary subgradient. In addition, other approaches for NSO include, for example, the gradient sampling method [22, 23, 77, 78], the discrete gradient method [9, 68], the quasi-secant method [8, 89] and the quasi-Newton method [87, 118].

In subgradient methods (Kiev methods), the basic idea is to replace the gradient with an arbitrary subgradient in smooth methods. This generalization is easy to perform and due to its simple structure it is widely used. However, subgradient methods have some serious disadvantages. First, we cannot guarantee that a descent direction is always obtained. Second, we do not have any implementable subgradient based stopping condition. Finally, the convergence rate of subgradient methods is poor.

To date, bundle methods together with their variations are considered to be the most efficient and reliable general purpose methods for solving NSO problems. The initial idea of bundle methods originates from the ε-steepest descent method [86] combining the cutting plane model [70] with the conjugate subgradient method [85, 125]. This method was developed further in [72], where two different strategies were designed to bound the number of subgradients used. Especially one of them, the so-called subgradient aggregation strategy, is widely used in various bundle methods. Next improvements were the proximal bundle method [73] and the bundle trust region method [114], which both present an insightful way to combine the bundle method with the trust region concept. After this several different bundle methods have been presented for both convex [3, 21, 66, 91, 92, 98, 104] and nonconvex [2, 34, 42, 43, 51, 52, 53, 96, 102] functions.

The basic idea in bundle methods is to approximate the subdifferential of an objective function with a bundle by collecting subgradients from previous iterations. This information is used to construct a model of the objective, which is utilized to determine a solution for the original problem. Typically this approximation is a convex cutting plane model defined as a point-wise maximum of a

set of first order approximations and for a convex function this model is always an underestimate. The convex cutting plane model is often reasonably good also for a nonconvex function. An exception to this are those parts of the objective, where there exists some concavity, since they cannot be captured with a convex model. In addition, a first order approximation does not need to support from below a nonconvex function. Therefore, we may loose the interpolation property of the model and to guarantee it several authors have introduced possible downward shifting of first order approximations (see, e.g., [74, 96, 114]). Due to these facts, a convex model of a nonconvex function can be a really rough estimate of the original problem and fail to represent the structure of the objective in the most relevant parts.

In this dissertation, the aim is to design new bundle methods, which in the modelling phase of the objective take both convex and concave behaviour explicitly into account. Some bundle methods utilizing "implicitly" convexity and concavity of the objective have been studied before [41, 42, 43, 46]. However, in these methods different types of behaviours of the objective are captured at the current iteration point by dividing first order approximations into two sets, but this may fail to describe the true structure of the objective. Therefore, we concentrate on so-called DC functions, which are expressed as a difference of two convex (DC) functions. These functions constitute an important and broad subclass of nonconvex functions and in many practical applications functions can be modelled explicitly in the DC form, for example, in production-transportation planning [60], data visualization [26] and computational chemistry [39]. In addition, the DC structure enables us to use convex analysis and optimization to some extent, even though this theory is typically lost in the nonconvex setting.

In our new bundle methods, the DC structure distinguishing convexity and concavity gives us a way to form a more accurate and realistic cutting plane model than the convex one, since we can construct separate approximations for the convex and concave parts. The new model is nonconvex and DC. In addition, it does not need the somewhat arbitrary downward shifting of the first order approximations. Therefore, by designing bundle methods for DC functions we are able to utilize the real nonconvex structure of the objective and this way obtain more efficient and reliable methods. Moreover, bundle methods utilizing explicitly the DC structure form a relatively new class of bundle methods and only recently a couple of methods from this class have come to light [45, 103].

The bundle methods for nonsmooth DC optimization presented in this dissertation are local solution methods. Before, DC optimization problems have been mainly considered in global optimization, where the aim is to find a global solution. Algorithms to globally solve DC problems are introduced, for example, in [61, 62, 120]. The development of local solution methods for DC optimization has attracted less attention (see, e.g., [7, 12, 83, 117]) despite the fact that they are typically needed in global solvers. At the moment one of the most commonly used local methods is DCA [107, 108, 109] based on local optimality and duality in DC optimization. However, there still exists a need for efficient local methods specifically developed for DC problems.

The dissertation consists of two parts. The first part, Part I, summarizes the results and contributions of this work and gives an introduction to related research. The second part, Part II, contains five original publications on which

this dissertation is based. Next, we give a more detailed description about Part I.

First, in Chapter 2 we give some basic notations and definitions. Especially, we introduce some generalizations of the gradient that yield us tools for designing methods in NSO. After that in Chapter 3 we familiarize ourselves with DC functions and give definitions for single- and multiobjective DC problems. In addition, we present different kinds of optimality conditions used in DC optimization.

In Chapter 4, we concentrate on standard proximal bundle methods and give a survey of these methods both in the convex and nonconvex case. The aim of this introduction is to highlight the characteristic features of bundle methods and especially illustrate the differences in the convex and nonconvex settings.

Chapter 5 is devoted to the new bundle methods developed for nonsmooth DC optimization. We start with introducing the new nonconvex DC cutting plane model based on the explicit utilization of the DC structure and show how the model is used to determine a search direction and a new iteration point. After that we introduce three proximal bundle methods relying on the new model. The first two of these methods are designed for unconstrained DC optimization whereas the third one is also able to handle multiobjective and constrained DC optimization problems.

Another new bundle method for unconstrained NSO is presented in Chapter 6. This bundle method is designed for large-scale problems and it does not assume that the objective function is a DC function. However, we implicitly collect some local information about the DC structure by dividing the first order approximations of the objective function into two sets. This way we are able to introduce another type of approach, which utilizes the convex and concave behaviour of the objective, but at the same time is capable of solving large-scale problems.

In Chapter 7, we focus on clusterwise linear regression (CLR) problems and introduce an approach to solve them, which is based on one of the methods developed in Chapter 5. We start with the introduction of a new formulation for the CLR problem utilizing the support vector machines (SVM) technique. This model is expressed as a DC function and a bundle method utilizing this DC structure is designed to solve CLR problems. In addition, some numerical results are presented.

Finally, Chapter 8 concludes the dissertation by giving a short summary about the benefits obtained by utilizing convexity and concavity of the objective function in bundle methods. In addition, some ideas for future research are discussed.

Chapter 2

Preliminaries

In this chapter, we present some notations and basic definitions including definitions of the generalized gradients for nonsmooth functions.

2.1 Notations and definitions

The n-dimensional Euclidean space is denoted by $\mathbb{R}^n$ and we use the notation $\boldsymbol{x} \in \mathbb{R}^n$ to present the column vector in this space. The row vector $\boldsymbol{x}^T$ is obtained by transposing $\boldsymbol{x}$. The inner product $\boldsymbol{x}^T\boldsymbol{y}$ of two vector $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{R}^n$ is

$$\boldsymbol{x}^T\boldsymbol{y} = \sum_{i=1}^{n} x_i y_i$$

and $\|\cdot\|$ is the norm in the n-dimensional Euclidean space $\mathbb{R}^n$, that is, $\|\boldsymbol{x}\| = (\boldsymbol{x}^T\boldsymbol{x})^{\frac{1}{2}}$. The open ball with a center $\boldsymbol{x} \in \mathbb{R}^n$ and a radius $\varepsilon > 0$ is denoted by

$$B(\boldsymbol{x};\varepsilon) = \{\boldsymbol{y} \in \mathbb{R}^n \mid \|\boldsymbol{y} - \boldsymbol{x}\| < \varepsilon\}.$$

A square matrix is denoted by $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, where n is the number of rows and columns. If $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is a diagonal matrix then we have nonzero elements only in the main diagonal and all the other elements are zero. Moreover, a square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is called *positive definite* if $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} > 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$ such that $\boldsymbol{x} \neq \boldsymbol{0}$. Similarly, a square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is *negative definite* if $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} < 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$ such that $\boldsymbol{x} \neq \boldsymbol{0}$.

A set $S \subseteq \mathbb{R}^n$ is *compact* if it is closed and bounded. In addition, the notation cl S is used to denote the *closure* of $S \subseteq \mathbb{R}^n$. A set $S \subseteq \mathbb{R}^n$ is a *cone* if $\lambda\boldsymbol{x} \in S$ for all $\boldsymbol{x} \in S$ and $\lambda \geq 0$, and we denote by ray $S = \{\lambda\boldsymbol{x} \mid \lambda \geq 0,\, \boldsymbol{x} \in S\}$ the smallest cone containing $S \subseteq \mathbb{R}^n$. In addition, a set $S \subseteq \mathbb{R}^n$ is *convex* if

$$\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in S$$

for all $\boldsymbol{x}$ and $\boldsymbol{y}$ in S and $\lambda \in [0,1]$ meaning that any closed line-segment joining two points of the set S belongs to this set. The *convex hull* of any set $S \subseteq \mathbb{R}^n$ is the smallest convex set containing S and it is denoted by conv S.

Next, we define some properties for a function $f : \mathbb{R}^n \to \mathbb{R}$. First, a function f is *convex* if

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$$

for all $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{R}^n$ and $\lambda \in [0,1]$. This means that any line-segment joining two values of the function f is above or on the graph of f. Similarly, a function f is *concave* if $-f$ is convex. If f is not convex, then it is called *nonconvex*.

A function f is *locally Lipschitz continuous* if at each point $\boldsymbol{x} \in \mathbb{R}^n$ there exists a constant $L > 0$ and a scalar $\varepsilon > 0$ such that

$$|f(\boldsymbol{y}) - f(\boldsymbol{z})| \leq L\|\boldsymbol{y} - \boldsymbol{z}\| \quad \text{for all } \boldsymbol{y}, \boldsymbol{z} \in B(\boldsymbol{x}; \varepsilon).$$

This property guarantees a good behaviour of f in the sense that it provides a bound for the change in the function values. For example, every smooth function as well as convex functions belong to the class of locally Lipschitz continuous functions.

A directional derivative gives an approximation for the change in function values. Thus, at any point it can be used to detect directions where the function value either increases, decreases or stays the same. For a function $f : \mathbb{R}^n \to \mathbb{R}$, the *directional derivative* at $\boldsymbol{x} \in \mathbb{R}^n$ in a direction $\boldsymbol{d} \in \mathbb{R}^n$ is defined as

$$f'(\boldsymbol{x}; \boldsymbol{d}) = \lim_{t \downarrow 0} \frac{f(\boldsymbol{x} + t\boldsymbol{d}) - f(\boldsymbol{x})}{t}.$$

In addition, if at a point $\boldsymbol{x} \in \mathbb{R}^n$ the directional derivative of f exists in every direction $\boldsymbol{d} \in \mathbb{R}^n$, then f is *directionally differentiable* at $\boldsymbol{x}$. Note that a general locally Lipschitz continuous function f is not necessarily directionally differentiable. However, for a convex function f this property is always guaranteed. Similarly, every difference of two convex functions can be shown to be directionally differentiable.

2.2 Subdifferentials

Next, we introduce some generalizations of gradients, so-called subdifferentials. Subdifferentials are important tools in nonsmooth analysis since for nonsmooth functions continuous gradients do not exist everywhere. In the following, we discuss the convex case and the general case separately and show relationships between different types of subdifferentials. The aim is not to give a detailed description, but to collect only those definitions and results which are needed in the following chapters. For more details about nonsmooth analysis we refer to [10, 29, 96, 113].

The *subdifferential* (or generalized gradient) of a convex function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $\boldsymbol{x} \in \mathbb{R}^n$ is defined as [113]

$$\partial_c f(\boldsymbol{x}) = \left\{ \boldsymbol{\xi} \in \mathbb{R}^n \mid f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{\xi}^T(\boldsymbol{y} - \boldsymbol{x}) \text{ for all } \boldsymbol{y} \in \mathbb{R}^n \right\} \tag{2.1}$$

and each element $\boldsymbol{\xi} \in \partial_c f(\boldsymbol{x})$ is called a *subgradient*. A subgradient can be interpreted as a slope of a linearization at a point $\boldsymbol{x}$ underestimating f. Therefore, for a convex function f subgradients can be used to construct lower approximations of f. This means that we preserve the useful property of gradients, since for a

smooth convex function the gradient can always be used to form an underestimate of f. However, in the nonsmooth case we may have several different linearizations at a single point instead of only one, and thus several subgradients. Nevertheless, if a convex function f is differentiable at $\boldsymbol{x} \in \mathbb{R}^n$, then the subdifferential $\partial_c f(\boldsymbol{x})$ contains only one subgradient, which equals the gradient $\nabla f(\boldsymbol{x})$, that is, $\partial_c f(\boldsymbol{x}) = \{\nabla f(\boldsymbol{x})\}$ [113].

To approximate the subdifferential of a convex function, we define the so-called ε-subdifferential. For $\varepsilon \geq 0$, the *ε-subdifferential* of a convex function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $\boldsymbol{x} \in \mathbb{R}^n$ is the set [113]

$$\partial_\varepsilon f(\boldsymbol{x}) = \{\boldsymbol{\xi}_\varepsilon \in \mathbb{R}^n \mid f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{\xi}^T(\boldsymbol{y} - \boldsymbol{x}) - \varepsilon \text{ for all } \boldsymbol{y} \in \mathbb{R}^n\}$$

and each $\boldsymbol{\xi}_\varepsilon \in \partial_\varepsilon f(\boldsymbol{x})$ is called an ε-subgradient. In this subdifferential, we have relaxed the condition that each ε-subgradient should construct a lower approximation of f since the deviation from the function value is tolerated with ε. Thus, this set contains subgradient information from some neighbourhood of $\boldsymbol{x}$ and, the smaller the parameter $\varepsilon > 0$ is, the better approximation of $\partial_c f(\boldsymbol{x})$ is obtained.

The previous definitions of the subdifferentials are not directly applicable for nonconvex functions and we need to generalize them. There are several possible alternatives available (see, e.g., [7, 12, 33, 99]). We present the Clarke subdifferential, which is one of the most commonly used generalizations and provides useful tools utilized, for example, in nonconvex bundle methods.

The *Clarke subdifferential* (or generalized subdifferential) for a locally Lipschitz continuous function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $\boldsymbol{x} \in \mathbb{R}^n$ is defined as [29]

$$\partial f(\boldsymbol{x}) = \operatorname{conv}\left\{\lim_{i\to\infty} \nabla f(\boldsymbol{x}_i) \,|\, \boldsymbol{x}_i \to \boldsymbol{x} \text{ and } \nabla f(\boldsymbol{x}_i) \text{ exists}\right\}$$

and, similarly to the convex case, each $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$ is called a subgradient. Contrary to the convex case, this subgradient does not necessarily provide a lower approximation of a nonconvex f. Nevertheless, it is known that $\partial f(\boldsymbol{x}) = \partial_c f(\boldsymbol{x})$ for a convex function $f : \mathbb{R}^n \to \mathbb{R}$ [29]. Thus, in the rest of the dissertation the notation $\partial f(\boldsymbol{x})$ is also used to denote the subdifferential of a convex function f.

The *Goldstein ε-subdifferential* of a locally Lipschitz continuous function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $\boldsymbol{x} \in \mathbb{R}^n$ for $\varepsilon \geq 0$ is given by the formula [96]

$$\partial_\varepsilon^G f(\boldsymbol{x}) = \operatorname{cl} \operatorname{conv}\{\partial f(\boldsymbol{y}) \,|\, \boldsymbol{y} \in B(\boldsymbol{x}; \varepsilon)\}. \tag{2.2}$$

With the selection $\varepsilon = 0$, the set $\partial_\varepsilon^G f(\boldsymbol{x})$ coincides with $\partial f(\boldsymbol{x})$. Moreover, $\partial f(\boldsymbol{x}) \subseteq \partial_\varepsilon^G f(\boldsymbol{x})$ for any $\varepsilon \geq 0$ and, therefore, the Goldstein ε-subdifferential is a generalization of the Clarke subdifferential. For this reason, the Goldstein ε-subdifferential can be used to approximate the set $\partial f(\boldsymbol{x})$ since, like for the ε-subdifferential, the decrease in the value of the parameter ε gives a more accurate approximation of $\partial f(\boldsymbol{x})$.

Chapter 3

DC optimization

In this dissertation, the interest is in nonsmooth DC optimization. Therefore, we introduce some basics about DC functions and give formal definitions for single-objective and multiobjective DC problems. In the multiobjective setting, we also introduce DC constraints providing a possibility to cover a wider and more complex set of problems. In addition, we present different types of optimality conditions used in DC optimization and discuss their usability since with these conditions we are able to detect promising candidate solutions, where the execution of an algorithm can be stopped. For more comprehensive study about DC optimization we refer to [54, 61, 82, 83, 107, 120].

3.1 DC function

A *DC function* refers to a function which can be presented as a difference of two convex (DC) functions and optimization problems having DC objectives and constraints are called *DC problems*. Typically, DC functions are nonconvex but, compared to a general nonconvex function, a DC function has a structure separating the convex and concave behaviour of a function. More specifically, a DC function is defined as follows:

Definition 3.1. *A function* $f : \mathbb{R}^n \to \mathbb{R}$ *is a* DC function *if it can be represented in the form*

$$f(\boldsymbol{x}) = f^1(\boldsymbol{x}) - f^2(\boldsymbol{x}),$$

where functions f^1, $f^2 : \mathbb{R}^n \to \mathbb{R}$ *are convex.*

The convex functions f^1 and f^2 defining a DC function f are called *DC components* and $f^1 - f^2$ is a *DC decomposition* of f. In addition, DC functions are locally Lipschitz continuous and they can be nonsmooth.

Nonconvexity of a DC function f is due to its concave part $-f^2$. However, the DC structure enables us to utilize convex analysis and optimization to some extent. This is an advantage compared to a general nonconvex function for which this useful theory is not applicable. Another nice feature is that DC functions are stable in a sense that they preserve the DC structure under simple operations frequently used in optimization. This differs from convexity which, for example, will be lost when multiplied with a negative scalar.

Proposition 3.2. *[54, 61, 120] Let $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \ldots, k$ be DC functions on $\mathbb{R}^n$. Then the following functions preserve the DC structure:*

(i) the sum $\sum_{i=1}^{k} c_i f_i(\boldsymbol{x})$ for any $c_i \in \mathbb{R}$;

(ii) the upper and lower envelopes $g(\boldsymbol{x}) = \max\{f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})\}$ and $h(\boldsymbol{x}) = \min\{f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})\}$;

(iii) the product $\prod_{i=1}^{k} f_i(\boldsymbol{x})$ and the quotient $f_1(\boldsymbol{x})/f_2(\boldsymbol{x})$ if $f_2(\boldsymbol{x}) \neq 0$ on $\mathbb{R}^n$;

(iv) the absolute value $|f_i(\boldsymbol{x})|$ for any $i = 1, \ldots, k$.

The class of DC functions is very broad and covers many frequently used functions in optimization. First, any convex or concave function has a trivial representation as a DC function. Second, every twice continuously differentiable function is a DC function [54, 57]. Moreover, Proposition 3.2 shows that with simple DC functions it is easy to generate really complex functions maintaining the DC structure. In addition, every continuous function can be approximated arbitrarily well by a DC function [120].

Even though many functions are known to be DC, it is not always trivial to obtain a DC decomposition. Thus, one important question in DC optimization is how a DC decomposition can be obtained. For a function being composed of convex and concave components, a DC decomposition is easily formed by separating the convex and concave terms. In addition, for the functions presented in Proposition 3.2 the DC decompositions can be formally written if the DC decompositions of the functions f_i for $i = 1, \ldots, k$ are known. Finally, in some applications DC representations can be explicitly constructed. These include cluster analysis [15], spherical separation [5, 6], supervised data classification problems [10], image restoration [84], finance and game theory [48], circuit design [88] and computational chemistry [39], to name a few.

It is worth to note that from one DC representation we can easily construct new ones by selecting any convex function and adding it to both DC components. Therefore, each DC function has an infinite set of different DC decompositions. The selected DC decomposition may have an influence on DC optimization and, thus, we would like to do the selection as efficiently as possible. However, the question how to select the best DC decomposition has no comprehensive answer. Even though we cannot answer this question in a general case, there exists some special cases where a solution is offered. One example is polynomials for which special algorithms are presented to determine the best DC decomposition [1, 19, 37]. In addition, a special norm minimization problem is presented in [38] to improve the DC representation mainly in the polynomial case.

3.2 Single-objective DC optimization problem

In single-objective DC optimization, we are minimizing one objective function. Thus, the unconstrained DC minimization problem is of the form

$$\begin{cases} \min & f(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{3.1}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is a DC function. A *global minimizer* (or a global optimum) is a point $\boldsymbol{x}^* \in \mathbb{R}^n$ satisfying $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^n$. This means that a global minimizer gives the smallest value for problem (3.1) on the whole space $\mathbb{R}^n$. However, problem (3.1) may have many local minimizers differing from the global ones since a DC function is often nonconvex. A point $\boldsymbol{x}^* \in \mathbb{R}^n$ is called a *local minimizer* (or a local optimum) if there exists $\varepsilon > 0$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in B(\boldsymbol{x}^*, \varepsilon)$. Therefore, a local minimizer $\boldsymbol{x}^*$ is guaranteed to yield the smallest value for problem (3.1) only in some neighbourhood of $\boldsymbol{x}^*$. Even though a global minimizer is always a local minimizer, it can be hard to distinguish from a local minimizer since local and global minimizers usually satisfy the same optimality conditions in algorithms. Therefore, we are typically content with local optimization methods finding local minimizers.

In order to design solution methods for problem (3.1), we need to define some optimality conditions which are used to detect solutions, where the execution of an algorithm can be stopped. These conditions are typically divided into two classes depending on whether they are sufficient or necessary. The satisfaction of a sufficient condition guarantees that a solution is either a global or local minimizer. Compared to this a necessary condition is weaker, since if such a condition is satisfied it does not guarantee optimality without some extra assumptions. However, both sufficient and necessary conditions can be used to exclude solutions, since if a solution does not fulfil those conditions it cannot be a local or global minimizer. Therefore, the use of necessary conditions also provides useful information and yields promising candidate solutions. Unfortunately, sufficient conditions can be hard to utilize in solution algorithms. Thus, we often need to be satisfied to use those conditions, which can be verified during the execution of the algorithm, even though they might be only necessary conditions.

Next, we introduce some optimality conditions for a DC function. This list is not comprehensive and more conditions are presented, for example, in [58, 83, 107]. We start with stating three common necessary conditions.

Theorem 3.3. *[29, 58, 83, 119] Let functions $f^1, f^2 : \mathbb{R}^n \to \mathbb{R}$ be convex. If a point $\boldsymbol{x}^* \in \mathbb{R}^n$ is a local minimizer of a DC function $f = f^1 - f^2$ then the following conditions hold*

$$\partial f^2(\boldsymbol{x}^*) \subseteq \partial f^1(\boldsymbol{x}^*), \tag{3.2}$$

$$\mathbf{0} \in \partial f(\boldsymbol{x}^*) \quad \textit{and} \tag{3.3}$$

$$\partial f^1(\boldsymbol{x}^*) \cap \partial f^2(\boldsymbol{x}^*) \neq \emptyset. \tag{3.4}$$

Points satisfying the first condition (3.2) are called *inf-stationary points*. Inf-stationarity is the strongest condition among the ones presented in Theorem 3.3, since if this condition holds then the other two conditions are also fulfilled. Furthermore, this condition guarantees local optimality if the second DC component f^2 is a polyhedral function of the form $f^2(\boldsymbol{x}) = \max_{1=1,\ldots,m}\{\boldsymbol{a}_i^T\boldsymbol{x} + b_i\}$, where $\boldsymbol{a}_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$ and $m \in \mathbb{N}$. Therefore, it would be beneficial to use inf-stationarity as a stopping condition in a solution algorithm. Unfortunately, this condition is hard to utilize in practise since the whole subdifferentials of DC components need to be known. Typically a subdifferential is not easy to calculate and sometimes it can be time-consuming to obtain even one subgradient from this set. Thus, in nonsmooth methods a typical requirement is that we know only one arbitrary

subgradient for an objective function at $\boldsymbol{x} \in \mathbb{R}^n$. In DC optimization, this assumption often appears in the form that at $\boldsymbol{x} \in \mathbb{R}^n$ one arbitrary subgradient for both DC components can be calculated and with this information we cannot validate inf-stationarity.

Condition (3.3) is widely used in solution algorithms designed for general objective functions and points satisfying it are called *Clarke stationary points*. In the convex case, this condition is also sufficient, since it guarantees global optimality. Even though this condition is often utilized in convex and nonconvex optimization, it can be hard to fulfil for a DC function if we do calculations only for DC components. This follows from the subdifferential calculus rule yielding [10]

$$\partial f(\boldsymbol{x}) \subseteq \partial f^1(\boldsymbol{x}) - \partial f^2(\boldsymbol{x}) \tag{3.5}$$

for a DC function $f = f^1 - f^2$. Thus, the difference of the subdifferentials of DC components is an estimate for the subdifferential of f. If f^1 or f^2 is differentiable then equality holds in (3.5) and the estimate coincides with the subdifferential of f. However, for an arbitrary DC decomposition this estimate may be very coarse [57]. Thus, arbitrary subgradients of DC components cannot be used to verify Clarke stationarity.

The third condition (3.4) is a relaxation of inf-stationarity. In inf-stationarity, the subdifferential of f^2 needs to be a subset of the subdifferential of f^1 and, thus, the intersection of these subdifferentials cannot be empty. Points satisfying this relaxed condition are called *critical points*. In addition, criticality is commonly used as a stopping condition in DC optimization, since it is quite easy to verify unlike conditions (3.2) and (3.3) and it typically provides good candidate solutions. However, one major drawback of a critical point is that it does not need to be a local optimum or even a saddle point. In the worst case, the algorithm may stop at a point, where the original DC function f is differentiable and the opposite of the gradient of f constructs a descent direction decreasing the value of f [65].

As already said, inf-stationarity is the strongest condition presented in Theorem 3.3 and it always implies Clarke stationarity and criticality. In addition, a Clarke stationary point always satisfies criticality. However, inverse relationships between these necessary conditions are not obtained in a general case. This means that criticality is the weakest condition. Due to the subdifferential calculus rule (3.5), criticality can imply Clarke stationarity when either f^1 or f^2 is differentiable. Moreover, to guarantee inf-stationarity with Clarke stationarity or criticality the DC component f^2 needs to be differentiable. A summary of these relationships is presented in Figure 3.1.

It is also worth noting that instead of criticality it is possible to test its generalization, the so-called *ε-criticality*, requiring that the ε-subdifferentials of the DC components intersect at the point $\boldsymbol{x}^*$ under consideration. This condition offers us more freedom in the solution process, since we can compare a little bit larger sets. In addition, the smaller the parameter $\varepsilon > 0$ is, the more accurate approximation of criticality is obtained. Naturally, ε-criticality coincides with criticality with the selection $\varepsilon = 0$.

For a DC function, it is also possible to construct the following sufficient optimality condition guaranteeing global optimality. Unfortunately, this condition is really hard to utilize in practise, since it requires that for each $\varepsilon \geq 0$ the ε-subdifferential of the DC component f^2 is a subset of the ε-subdifferential of the

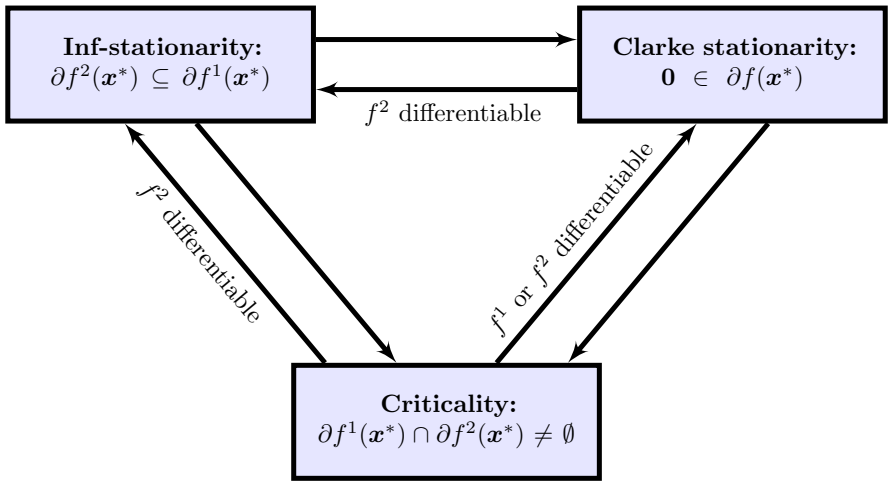


Figure 3.1: Relationships between different stationary points

DC component f^1. Thus, with each value of ε we have the same difficulties than with inf-stationarity.

Theorem 3.4. [58] *Let functions $f^1, f^2 : \mathbb{R}^n \to \mathbb{R}$ be convex. A point $\boldsymbol{x}^* \in \mathbb{R}^n$ is a global minimizer of a DC function $f = f^1 - f^2$, if and only if*

$$\partial_\varepsilon f^2(\boldsymbol{x}^*) \subseteq \partial_\varepsilon f^1(\boldsymbol{x}^*) \quad \textit{for all } \varepsilon \geq 0.$$

3.3 Multiobjective and constrained DC optimization problem

Multiobjectivity arises inherently in many optimization problems. In production planning, the interest may be, for example, to maximize quality while minimizing production and labour costs. Therefore, in multiobjective optimization we are optimizing several objective functions simultaneously instead of only one. In the nontrivial setting, these functions conflict with each other, and we need to do trade-offs between different goals to find a good enough compromise as a solution. Thus, the concept of a solution differs from the single-objective case and typically there does not exist a single solution yielding the optimal value for each objective function. The constraints, in turn, are used to give restrictions for problems. This way we are able to exclude some parts of the search space and to concentrate only on the interesting and relevant region. For example, in production planning we are able to restrict the amounts of products used.

The constrained multiobjective DC minimization problem is formally defined as

$$\begin{cases} \min & f_1(\boldsymbol{x}), \ldots, f_h(\boldsymbol{x}) \\ \text{s. t.} & \boldsymbol{x} \in X, \end{cases} \tag{3.6}$$

where each objective function $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i \in I$ is a DC function and $I = \{1, \ldots, h\}$ is a set of indices of the objectives with $h \geq 2$. The set $X \subseteq \mathbb{R}^n$ is the *feasible region* and a point $\boldsymbol{x}$ belonging to X is called a *feasible solution*. Since we consider only DC inequality constraints this set has the form $X = \{\boldsymbol{x} \in \mathbb{R}^n \,|\, g_l(\boldsymbol{x}) \leq 0,\, l \in L\}$, where $L = \{1, \ldots, m\}$ denotes the indices of the constraints and each constraint $g_l : \mathbb{R}^n \to \mathbb{R}$, $l \in L$ is a DC function. If we do not have constraints in problem (3.6), then the set $X = \mathbb{R}^n$.

To define the optimality concept in the multiobjective setting, we need to be able to measure the quality of solutions with some preference relations. If two solutions $\boldsymbol{x}$ and $\boldsymbol{y}$ on X satisfy the condition

$$f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y}) \text{ for all } i \in I \text{ and } f_i(\boldsymbol{x}) < f_i(\boldsymbol{y}) \text{ for some } i \in I$$

then $\boldsymbol{x}$ *dominates* $\boldsymbol{y}$ since one objective function has a strictly better value while the others are at least equally good. Another preference can be defined for solutions $\boldsymbol{x}$ and $\boldsymbol{y}$ on X with the condition

$$f_i(\boldsymbol{x}) < f_i(\boldsymbol{y}) \text{ for all } i \in I.$$

In this case, $\boldsymbol{x}$ *strictly dominates* $\boldsymbol{y}$ since the value of each objective function is strictly less than the one obtained with using the solution $\boldsymbol{y}$. Utilizing these two preferences we can define the optimality concept used in multiobjective optimization, the so-called *Pareto optimality*.

Definition 3.5. *For problem* (3.6)*, a solution* $\boldsymbol{x}^* \in X$ *is called*

(i) globally Pareto optimal *if there does not exist a solution* $\boldsymbol{x} \in X$ *dominating* $\boldsymbol{x}^*$*;*

(ii) globally weakly Pareto optimal *if there does not exist a solution* $\boldsymbol{x} \in X$ *strictly dominating* $\boldsymbol{x}^*$*;*

(iii) locally (weakly) Pareto optimal *if there exists* $\delta > 0$ *such that* $\boldsymbol{x}^*$ *is globally (weakly) Pareto optimal on* $X \cap B(\boldsymbol{x}^*; \delta)$.

Pareto optimality means that there does not exist any feasible solution improving one objective f_i without impairing some other objective at the same time. For problem (3.6), there usually exist several different Pareto optimal solutions. These solutions may differ a lot from each other, but mathematically they are equally good. On the other hand, weak Pareto optimality guarantees that there does not exist a feasible solution improving the value of all the objective functions f_i, $i \in I$. Therefore, the set of weakly Pareto optimal solutions always contains Pareto optimal solutions, but the inverse does not hold. In Figure 3.2, we have illustrated these concepts for a multiobjective problem with two objectives.

To formulate an optimality condition for the constrained problem (3.6), we introduce sets

$$F(\boldsymbol{x}) = \bigcup_{i \in I} \partial f_i(\boldsymbol{x}) \quad \text{and} \quad G(\boldsymbol{x}) = \bigcup_{l \in L(\boldsymbol{x})} \partial g_l(\boldsymbol{x}),$$

where $L(\boldsymbol{x}) = \{l \in L \,|\, g_l(\boldsymbol{x}) = 0\}$. Moreover, we use the contingent cone $K_X(\boldsymbol{x})$ of the set X at $\boldsymbol{x} \in X$ and the polar cone $G^{\geq}(\boldsymbol{x})$ of the set $G(\boldsymbol{x})$ (see, e.g., [10])

to define the constraint qualification

$$G^{\geq}(\boldsymbol{x}) \subseteq K_X(\boldsymbol{x}). \tag{3.7}$$

With these we can give the following necessary optimality condition.

Theorem 3.6. *[94] If a point $\boldsymbol{x}^* \in X$ is locally weakly Pareto optimal for problem* (3.6) *and the constraint qualification* (3.7) *holds at $\boldsymbol{x}^*$, then*

$$\mathbf{0} \in \operatorname{conv} F(\boldsymbol{x}^*) + \operatorname{cl} \operatorname{cone} G(\boldsymbol{x}^*), \tag{3.8}$$

where $\operatorname{cone} G(\boldsymbol{x}^*) = \operatorname{ray} \operatorname{conv} G(\boldsymbol{x}^*)$ *is the smallest convex cone containing* $G(\boldsymbol{x}^*)$.

Points satisfying condition (3.8) are called *weakly Pareto stationary*. In addition, this condition is often used to create a stopping condition in multiobjective solution algorithms and it can be seen as a generalization of Clarke stationarity.

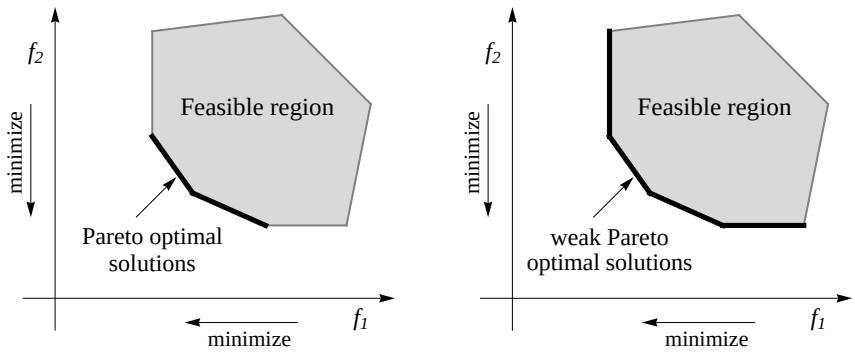


Figure 3.2: Illustrations about Pareto and weak Pareto optimal solutions

Chapter 4

Standard proximal bundle method

The main focus in this dissertation is on bundle methods for NSO. Therefore, we now present the basic idea of general bundle methods both in the convex and nonconvex cases. It is worth to note that these bundle methods do not utilize or assume the DC structure of the objective. Despite of that many of their distinctive features are later on used when we design bundle methods for DC functions.

Bundle methods are iterative solution algorithms and one of their characteristic feature is to approximate the subdifferential of the objective function with a bundle. At the current iteration, this bundle contains subgradients obtained from previous iterations together with the newest one and this way we can take advantage of the information produced during the execution of the algorithm. To determine a new iteration point, the bundle is utilized to construct an approximation for the objective. This model is typically piecewise linear and convex, but also other types of models have been designed (see, e.g., [4, 35, 36, 41, 91]). By adding a stabilizing term in the model, we obtain a direction finding problem, whose solution is a descent direction for the model and, in a good case, also for the original objective.

When the search direction is found, a line search is typically performed to obtain an auxiliary point. After this another characteristic feature of bundle methods is to decide whether to execute a serious or null step. If the value of the objective decreases significantly in the auxiliary point, then we perform a serious step and update the current iteration point with the auxiliary point. Otherwise, we need to improve the model by executing a null step and the current iteration point does not change. However, in both steps the bundle is improved with a new subgradient to build a more accurate approximation of the objective.

In this chapter, we concentrate on proximal bundle methods, where a specific parameter, the so-called proximity measure, is utilized in the stabilizing term. In the convex case, the usage of this parameter enables us to omit the line search since the stepsize can be controlled with the parameter. Next, we introduce a survey of the standard proximal bundle method and highlight the differences between the convex and nonconvex cases. For more detailed descriptions we refer to [56, 72, 73, 96, 114].

4.1 Model for a function

We start with assuming that the objective function f is locally Lipschitz continuous. To construct a model for such a function, we need some information about the objective and its subdifferential. Since in practise the whole subdifferential $\partial f(\boldsymbol{x})$ is usually hard or even impossible to determine the main requirement in bundle methods is the following:

Assumption 4.1. *At each point $\boldsymbol{x} \in \mathbb{R}^n$ we can calculate the value of the function $f(\boldsymbol{x})$ and one arbitrary subgradient $\boldsymbol{\xi} \in \partial f(\boldsymbol{x})$.*

This assumption guarantees that at each point we obtain a sufficient amount of information to build a model. Furthermore, since we assume that only one arbitrary subgradient is calculated at any point the requirement is not really restrictive and can be satisfied in many optimization problems.

An important feature in bundle methods is to utilize the information from the previous iterations. Therefore, this information is collected into a *bundle* providing an approximation for the subdifferential of the objective. During the iteration k, let $\boldsymbol{x}_k \in \mathbb{R}^n$ be the current iteration point. In addition, we have at our disposal auxiliary points $\boldsymbol{y}_j \in \mathbb{R}^n$ and subgradients $\boldsymbol{\xi}_j \in \partial f(\boldsymbol{y}_j)$ for $j \in J_k$, where the index set J_k is typically a nonempty subset of $\{1, \ldots, k\}$. In order to maintain this information, the bundle is defined as a set

$$\mathcal{B}_k = \{(\boldsymbol{y}_j, f(\boldsymbol{y}_j), \boldsymbol{\xi}_j)) \,|\, j \in J_k\}. \tag{4.1}$$

Note that the current iteration point is always one of the auxiliary points and, thus, it belongs to $\mathcal{B}_k$.

The bundle has a central role in the model construction, since each of its elements can be used to determine a linearization of f. For the index $j \in J_k$, the *linearization* $l_j : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$l_j(\boldsymbol{x}) = f(\boldsymbol{y}_j) + \boldsymbol{\xi}_j^T(\boldsymbol{x} - \boldsymbol{y}_j) \quad \text{for all } \boldsymbol{x} \in \mathbb{R}^n.$$

For a convex function, this linearization always supports from below the epigraph of the function since subgradients satisfy formula (2.1). This is a desired property in most bundle methods, but unfortunately it is not necessarily true for a nonconvex function. Therefore, we need to adjust linearizations in the nonconvex case to guarantee that they underestimate f at least in some neighbourhood of $\boldsymbol{x}_k$. Next, we present separately the models for convex and nonconvex functions.

4.1.1 Convex case

The aim is to construct a model underestimating the convex function f. Since we assume convexity each element of the bundle constructs a linearization being a lower approximation of f. Thus, at the point $\boldsymbol{x}_k$ the model for f is obtained by combining the linearizations and the piecewise linear approximation is given by

$$\hat{f}_k(\boldsymbol{x}) = \max_{j \in J_k} \{l_j(\boldsymbol{x})\} = \max_{j \in J_k} \left\{ f(\boldsymbol{y}_j) + \boldsymbol{\xi}_j^T(\boldsymbol{x} - \boldsymbol{y}_j) \right\} \tag{4.2}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$. This model is the classical *cutting plane model* used in convex bundle methods [64, 73, 93, 96, 114]. It has several good features which support

its usefulness. First, the cutting plane model underestimates f everywhere since the upper envelope maintains the corresponding property of linearizations. Second, $\hat{f}_k(\boldsymbol{y}_j) = f(\boldsymbol{y}_j)$ for $j \in J_k$ and, thus, the model is accurate at any auxiliary point. Furthermore, the model maintains convexity and by adding new linearizations we obtain a more accurate approximation of f. In Figure 4.1, we have illustrated these properties by presenting the cutting plane model of the convex function $f(x) = \max\{x^2 + x + 2, -x + 2\}$, when linearizations are constructed at points -3, -1 and 1.

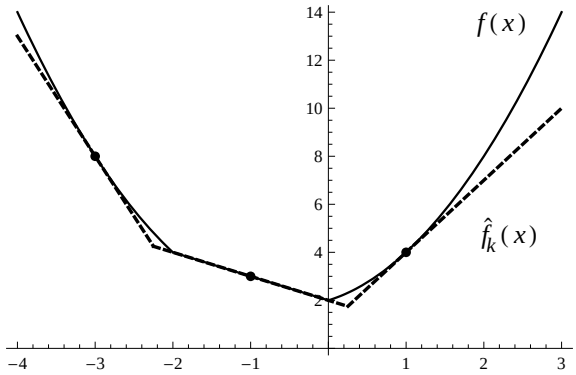


Figure 4.1: The convex cutting plane model of the convex function

Another useful feature of the model is that it can be rewritten in the form

$$\hat{f}_k(\boldsymbol{x}) = \max_{j \in J_k} \left\{ f(\boldsymbol{x}_k) + \boldsymbol{\xi}_j^T (\boldsymbol{x} - \boldsymbol{x}_k) - \alpha_j^k \right\}, \tag{4.3}$$

when we utilize the current iteration point $\boldsymbol{x}_k$ and define the *linearization error*

$$\alpha_j^k = f(\boldsymbol{x}_k) - f(\boldsymbol{y}_j) - \boldsymbol{\xi}_j^T (\boldsymbol{x}_k - \boldsymbol{y}_j) \quad \text{for all } j \in J_k. \tag{4.4}$$

The linearization error describes the difference at the current iteration point $\boldsymbol{x}_k$ between the actual value of f and the value of the linearization constructed at $\boldsymbol{y}_j$. Formula (2.1) guarantees that linearization errors are always nonnegative, that is, $\alpha_j^k \geq 0$ for $j \in J_k$ and this means that each linearization is below the function f at $\boldsymbol{x}_k$. In addition, the linearization error can be used to measure the quality of the model since the smaller the linearization error is, the better the corresponding linearization represents the function f at the point $\boldsymbol{x}_k$.

4.1.2 Nonconvex case

For a nonconvex function f, the aim is also to design a proper approximation, and in many nonconvex bundle methods (see, e.g., [42, 72, 96, 114]) the construction is done similarly to the convex case. However, since linearizations do not need to underestimate a nonconvex function we may be unable to produce even a local underestimate of f at $\boldsymbol{x}_k$ (see Figure 4.2). In model (4.3), this type of an undesirable feature occurs whenever a linearization error is negative. Moreover, a linearization error may be small even though an auxiliary point $\boldsymbol{y}_j$ is far away from

$\boldsymbol{x}_k$. Therefore, a linearization error is not a reliable tool to measure the quality of the model in nonconvex settings [93].

One option to guarantee the interpolation property at least at the current iteration point $\boldsymbol{x}_k$ is to downward shift linearizations. Several authors have used this approach (see, e.g., [72, 96, 114]) and utilized *subgradient locality measures* β_j^k instead of linearization errors α_j^k. Typically the definition of β_j^k is given in the form

$$\beta_j^k = \max\left\{|\alpha_j^k|,\, \gamma\|\boldsymbol{x}_k - \boldsymbol{y}_j\|^\omega\right\},$$

where $\gamma \geq 0$ is a distance measure parameter and $\omega \geq 1$ is a locality measure parameter. Thus, β_j^k includes the comparison of the absolute value of the linearization error with some positive distance measure. This way negative linearization errors are replaced with nonnegative values and we obtain some localizing information to the approximation, which also takes into account the distance between an auxiliary point $\boldsymbol{y}_j$ and the current iteration point $\boldsymbol{x}_k$.

Utilizing subgradient locality measures, the cutting plane model of f at $\boldsymbol{x}_k$ can be defined similarly to (4.3) with the formula

$$\hat{f}_k(\boldsymbol{x}) = \max_{j \in J_k}\left\{f(\boldsymbol{x}_k) + \boldsymbol{\xi}_j^T(\boldsymbol{x} - \boldsymbol{x}_k) - \beta_j^k\right\}. \tag{4.5}$$

Unlike in the convex case, this model is guaranteed to be only a local approximation of a nonconvex f at some neighbourhood of $\boldsymbol{x}_k$ and it does not necessarily coincide with f at each $\boldsymbol{y}_j$ for $j \in J_k$. Moreover, the model maintains convexity and, thus, it is not able to capture a concave behaviour of a nonconvex f. Another disadvantage is that the amount of shifting is more a less arbitrary and we do not know how the subgradient locality measure affects the model.

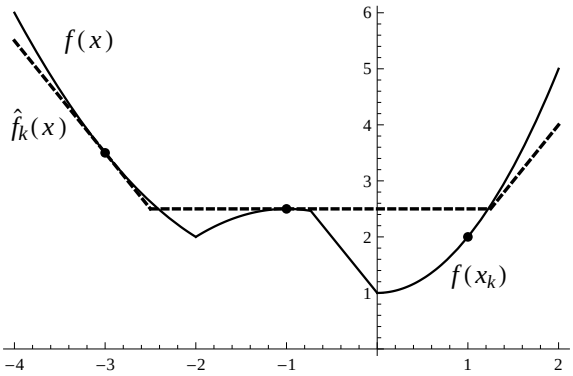


Figure 4.2: The convex model (4.3) of the nonconvex function

To illustrate the cutting plane model in the nonconvex setting, we consider the function $f(x) = \max\{x^2 + x + 2, -x + 2\} - \max\{0.5x^2, x + 1\}$ and let the current iteration point x_k be 1. Linearizations are formed at points -3, -1 and 1. The direct application of model (4.3) is presented in Figure 4.2 and this example shows the undesired and problematic feature that the function f is overestimated at $x_k = 1$. To fix this problem, the cutting plane model (4.5) shifts linearizations

with subgradient locality measures. One option is the absolute value of the linearization error, that is, $\beta_j^k = |\alpha_j^k|$ for all $j \in J_k$. With this selection we obtain the approximation presented in Figure 4.3 and it supports from below f at x_k but not, for example, at point zero.

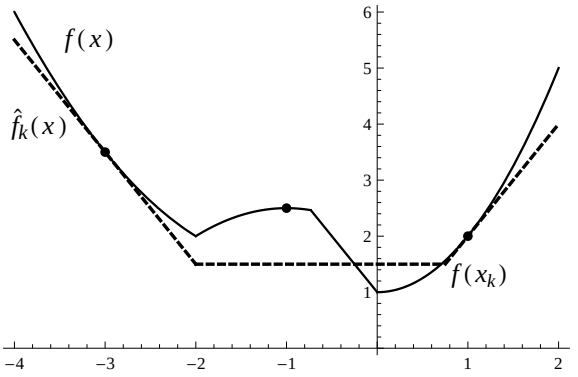


Figure 4.3: The convex cutting plane model with $\beta_j^k = |\alpha_j^k|$

4.2 Direction finding

Bundle methods are typically descent algorithms. That is, during each iteration the aim is to generate a search direction yielding a new better iteration point, which decreases the value of the objective. For this reason, the search direction has a central role in bundle methods and, next, we consider how it is obtained by utilizing the cutting plane model.

In the previous sections, we have presented the cutting plane models separately for convex and nonconvex functions. Since model (4.5) can be seen as an extension of (4.3) with the selection $\beta_j^k = \alpha_j^k$ we consider in the following only the cutting plane model (4.5). First, we denote by $\boldsymbol{d} = \boldsymbol{x} - \boldsymbol{x}_k$ the search direction at the current iteration point $\boldsymbol{x}_k$. With this notation we can rewrite model (4.5) as follows

$$\hat{f}_k(\boldsymbol{x}_k + \boldsymbol{d}) = \max_{j \in J_k} \left\{ f(\boldsymbol{x}_k) + \boldsymbol{\xi}_j^T \boldsymbol{d} - \beta_j^k \right\}. \tag{4.6}$$

It would be logical to determine the search direction $\boldsymbol{d}_k \in \mathbb{R}^n$ by minimizing the cutting plane model (4.6). Unfortunately, this model cannot be directly applied since it is piecewise linear and it does not necessarily have a finite minimizer. Therefore, we cannot guarantee the existence of a solution. Moreover, the approximation of f can be trusted only in some neighbourhood of $\boldsymbol{x}_k$. This is due to the fact that usually the farther away we are from the point $\boldsymbol{x}_k$, the more unreliable the approximation becomes. Thus, we need to insert a *stabilizing term* into the model to guarantee the existence of the solution and to keep the approximation local enough [93]. There exist several different stabilizing terms for bundle methods (see, e.g., [40, 73, 75, 93]). In proximal bundle methods, this term is of the form $(1/2t)\|\boldsymbol{d}\|^2$, where $t > 0$ is a proximity measure. The benefit of this selection is

that we can modify the stepsize by changing the proximity measure. In addition, the proximity measure is able to accumulate some second order information of the curvature of f around $\boldsymbol{x}_k$ [10].

The search direction $\boldsymbol{d}_k \in \mathbb{R}^n$ is now obtained by solving the optimization problem

$$\begin{cases} \min & \hat{f}_k(\boldsymbol{x}_k + \boldsymbol{d}) + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{s. t.} & \boldsymbol{d} \in \mathbb{R}^n, \end{cases} \tag{4.7}$$

where the proximity measure t controls the stepsize. In addition, the tuning of the proximity measure can be seen as the adjustment of the trust region, since an increase in the value of t enlarges the search space whereas a decrease in t makes the acceptable region smaller [114]. Furthermore, the adjustment of the proximity measure needs to be done appropriately and different adjustment schemes are presented in [74, 76, 114]. However, the general idea is that if our approximation is inconsistent and fails to model f, then we can decrease t in order to concentrate the search on a smaller neighbourhood of the current iteration point $\boldsymbol{x}_k$. Correspondingly, we can increase the value of t whenever the model described f correctly in order to speed up the execution of the method.

To obtain the search direction $\boldsymbol{d}_k$, we need to minimize the nonsmooth strictly convex problem (4.7). Unfortunately, in this form the optimization problem is challenging to solve due to nonsmoothness. However, the specific structure of $\hat{f}_k$ enables us to rewrite (4.7) easily as a smooth strictly convex optimization problem with a quadratic objective and linear constraints [93]. More precisely the reformulation is stated as

$$\begin{cases} \min & v + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{s. t.} & \boldsymbol{\xi}_j^T \boldsymbol{d} - \beta_j^k \leq v \quad \text{for all } j \in J_k \\ & v \in \mathbb{R},\ \boldsymbol{d} \in \mathbb{R}^n. \end{cases} \tag{4.8}$$

Therefore, we are able to utilize the powerful tools of smooth convex analysis and use a quadratic solver designed for linearly constrainted problems [90]. Instead of problem (4.8) it is also possible to solve its dual counterpart which is presented in a general form, for example, in [93]. This can ease the solution process, since the dual formulation is typically easier and less time-consuming to solve than the primal problem (4.8).

4.3 Determination of a new iteration point

After the search direction $\boldsymbol{d}_k$ is computed the next step is to decide how the new iteration point $\boldsymbol{x}_{k+1}$ is calculated. Unfortunately, a simple choice $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k$ is not enough to guarantee global convergence of the proximal bundle method. Moreover, the straightforward minimization of the function f along the direction $\boldsymbol{d}_k$ is not always sufficient either. One reason for this is that even though $\boldsymbol{d}_k$ is a descent direction for the model $\hat{f}_k$ it may be a nondescent direction for f. Second, whenever f is decreased this reduction needs to be sufficiently large since otherwise we may have infinitely many steps with only a marginal decrease which can impair the convergence [96]. Thus, we need more sophisticated ways to determine $\boldsymbol{x}_{k+1}$.

This determination is started by computing a new auxiliary point $\boldsymbol{y}_{k+1}$. In the nonconvex case, we need to perform a specific line search (see, e.g., [72, 96]) along the direction $\boldsymbol{d}_k$ to generate it. Thus, this procedure yields the stepsize $t_L^k > 0$ after which we set $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + t_L^k \boldsymbol{d}_k$. In the convex case, the proximity measure eases this calculation and it is not obligatory to use a line search procedure even though it can speed up the performance of the method. Instead, we can directly select the stepsize $t_L^k = 1$ and set as a new auxiliary point $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k$.

After the auxiliary point is obtained characteristic to bundle methods is to decide whether we perform a *serious step* or a *null step*. In order to do a serious step, the decrease in the objective function needs to be significant. Since the cutting plane model provides the predicted amount of descent

$$v_k = \hat{f}_k(\boldsymbol{x}_k + \boldsymbol{d}_k) - f(\boldsymbol{x}_k) < 0,$$

we require that the real decrease of f is at least some sufficient percentage of the predicted one. This means that a serious step is done by setting $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$ if

$$f(\boldsymbol{y}_{k+1}) - f(\boldsymbol{x}_k) \leq m\, t_L^k v_k, \tag{4.9}$$

where $m \in (0, 1/2)$ is a descent parameter. After this we calculate $\boldsymbol{\xi}_{k+1} \in f(\boldsymbol{x}_{k+1})$ and add a new element $(\boldsymbol{x}_{k+1}, f(\boldsymbol{x}_{k+1}), \boldsymbol{\xi}_{k+1})$ into the bundle before starting a new iteration.

When condition (4.9) is not satisfied we perform a null step and set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$. Since the model is inconsistent we need to improve it to get a more accurate approximation. Therefore, before starting a new iteration, we insert into the bundle a new element $(\boldsymbol{y}_{k+1}, f(\boldsymbol{y}_{k+1}), \boldsymbol{\xi}_{k+1})$ calculated at the auxiliary point $\boldsymbol{y}_{k+1}$ where $\boldsymbol{\xi}_{k+1} \in \partial f(\boldsymbol{y}_{k+1})$. In the convex case, this new element always improves the model. However, without the line search a similar improvement is not always guaranteed in the nonconvex case. Due to this, the auxiliary point $\boldsymbol{y}_{k+1}$ is calculated with this specific procedure forcing a significant modification into the cutting plane model whenever a null step is done.

The basic structure of the proximal bundle method is presented in Figure 4.4. From this flowchart we see that the proximal bundle method consists of a sequence of null and serious steps and the execution of the algorithm is continued until the desired accuracy is achieved. Typically the predicted descent v_k (or its modification) is used as a stopping condition guaranteeing Clarke stationarity. That is, the execution can be stopped, for example, when $|v_k| \leq \varepsilon$ where $\varepsilon > 0$ is a final accuracy parameter supplied by the user [10].

The global convergence of the proximal bundle method is proved under the assumption of upper semi-smoothness [18] for a locally Lipschitz continuous function which is neither necessarily convex nor differentiable. In addition, for a convex f we can guarantee that a Clarke stationary point is always a global minimizer. For a nonconvex function, this kind of a result does not hold without some extra assumptions and, thus, we only know that the algorithm converges to a Clarke stationary point.

To implement the proximal bundle method, the size of $\mathcal{B}_k$ has to be bounded. The reason for this is that the larger the bundle $\mathcal{B}_k$ is, the more memory is required and the more difficult problem (4.8) becomes. One option to limit the size of $\mathcal{B}_k$ and still guarantee the global convergence is to utilize a subgradient aggregation

strategy [72], which uses an aggregated subgradient and linearization error to accumulate information from the previous iterations. Even with this option the number of stored subgradients is typically related to the dimension of the problem and, thus, the standard bundle method is developed only for small- and medium-scale problems.

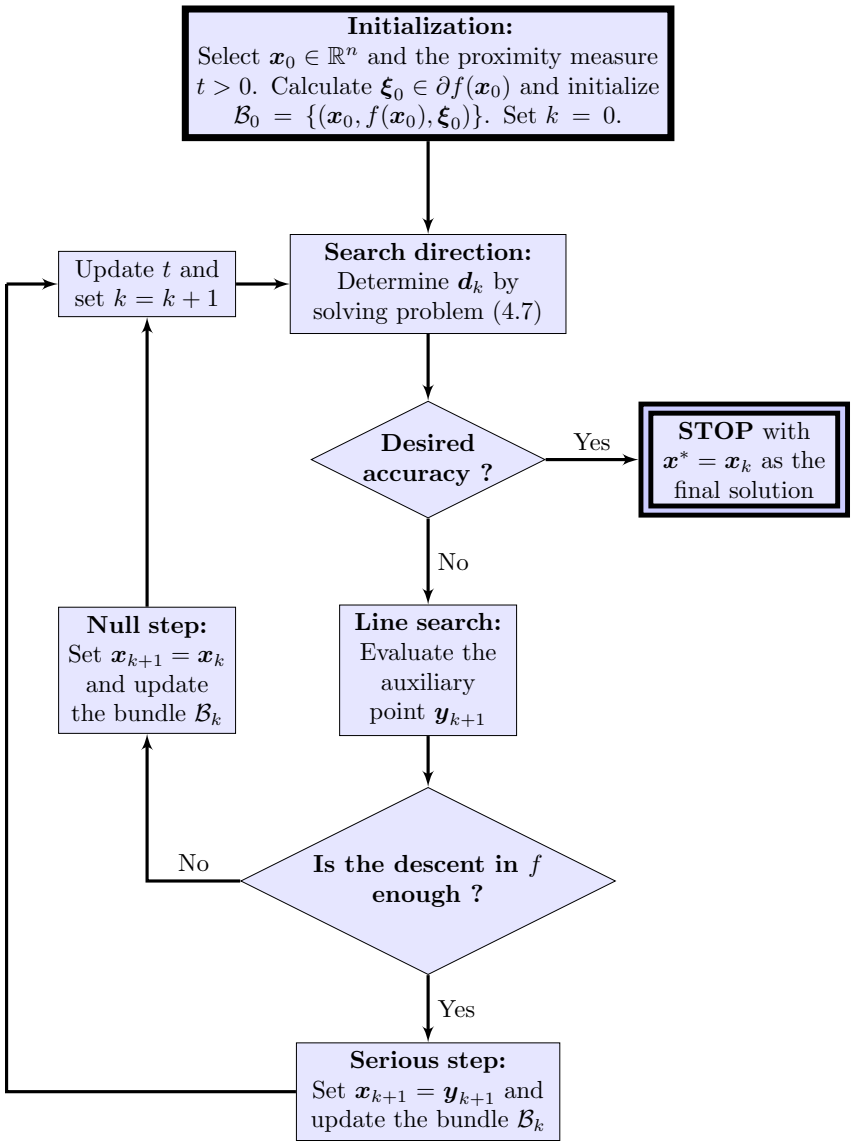


Figure 4.4: The basic structure of the proximal bundle method

Chapter 5

Usage of the DC structure in bundle methods

In this chapter, we review shortly original Publications I−III, where proximal bundle methods for nonsmooth DC optimization are designed. Thus, instead of general nonconvex objective functions we restrict our consideration on DC functions. As we have already seen, the class of DC functions is very broad and constitutes an important subclass of nonconvex functions.

The benefit of our choice enables us to utilize explicitly the DC structure of the objective in the model construction. This way we are able to build a convex cutting plane model separately for both convex DC components. In addition, by combining these separate models we obtain a new nonconvex DC cutting plane model capturing both the convex and concave behaviour of the objective. This way we obtain a more accurate approximation of a nonconvex objective since the convex cutting plane model (4.5) used in nonconvex bundle methods can be replaced with it.

In the following, we first introduce the new cutting plane model and show how it yields a search direction together with a new iteration point. After that we highlight the main aspects of three different bundle methods utilizing this new model. Publication I introduces the first method, the so-called proximal bundle method (PBDC), designed for unconstrained DC problems (3.1). This method is proved to find ε-critical points. Since critical points have some poor properties, the double bundle method (DBDC) based on a stronger optimality condition is designed in Publication II. The main difference to the PBDC is the new escaping procedure which guarantees Clarke stationarity of the obtained solution. Similar to the PBDC, the DBDC is also designed for unconstrained DC problems (3.1). Lastly, we present the multiobjective double bundle method (MDBDC) for problem (3.6) developed in Publication III. This method is the successor of the DBDC capable of handling multiobjective DC problems together with DC constraints.

5.1 Model for a DC function

We start with presenting the new cutting plane model of a DC function. Characteristic to bundle methods we assume that some information about the objective is provided at each point $\boldsymbol{x} \in \mathbb{R}^n$. However, since the DC decomposition is utilized the following requirement differs from the one presented in Assumption 4.1.

Assumption 5.1. *At each point $\boldsymbol{x} \in \mathbb{R}^n$ we can evaluate the value of the DC components $f^1(\boldsymbol{x})$ and $f^2(\boldsymbol{x})$ and two arbitrary subgradients $\boldsymbol{\xi}^1 \in \partial f^1(\boldsymbol{x})$ and $\boldsymbol{\xi}^2 \in \partial f^2(\boldsymbol{x})$.*

With this assumption we can guarantee that at each point we obtain the same amount of information about both DC components. In addition, this requirement is typically quite easy to satisfy if the DC decomposition of the objective is known. Therefore, it is important to have the exact DC decomposition, since otherwise the DC structure cannot be utilized.

The main idea is to approximate both DC components separately. Therefore, we form for each DC component its own bundle (4.1) approximating its subdifferential. This means that at the current iteration point $\boldsymbol{x}_k \in \mathbb{R}^n$ we have two separate bundles, which are denoted by

$$\mathcal{B}_k^i = \{(\boldsymbol{y}_j, f^i(\boldsymbol{y}_j), \boldsymbol{\xi}_j^i)) \,|\, j \in J_k^i\} \quad \text{for } i = 1, 2, \tag{5.1}$$

where the superscript i tells the DC component in question and $\boldsymbol{\xi}_j^i \in \partial f^i(\boldsymbol{y}_j)$. In addition, the index sets J_k^1 and J_k^2 need not to be similar. However, the current iteration point $\boldsymbol{x}_k$ is always assumed to belong to both bundles.

To utilize the DC structure in the model construction, we first form the classical convex cutting plane model (4.3) for both DC components. Therefore, the approximation of the DC component f^i is constructed with the formula

$$\hat{f}_k^i(\boldsymbol{x}) = \max_{j \in J_k^i} \left\{ f^i(\boldsymbol{x}_k) + (\boldsymbol{\xi}_j^i)^T(\boldsymbol{x} - \boldsymbol{x}_k) - \alpha_{j,i}^k \right\} \quad \text{for } i = 1, 2,$$

where $\alpha_{j,i}^k$ is the linearization error calculated for f^i and it can be obtained from (4.4) by replacing the function f with f^i and the subgradient $\boldsymbol{\xi}_j$ with $\boldsymbol{\xi}_j^i$.

The new model of the original DC function f is formed by combining the separate approximations of its DC components f^1 and f^2. This yields a *nonconvex DC cutting plane model*

$$\tilde{f}_k(\boldsymbol{x}) = \hat{f}_k^1(\boldsymbol{x}) - \hat{f}_k^2(\boldsymbol{x}), \tag{5.2}$$

which is piecewise linear and captures both the convex and concave behaviour of f. Therefore, the main advantage of the model is its ability to produce a more realistic and accurate approximation of a nonconvex objective than the convex model (4.5). Another benefit is that the new model coincides with f at the current iteration point $\boldsymbol{x}_k$. Therefore, it is enough to use the linearization errors $\alpha_{j,i}^k$ for $i = 1, 2$ and we avoid the somewhat arbitrary downward shifting of linearizations often used in nonconvex bundle methods.

To better illustrate the differences between approximations (4.5) and (5.2), we have presented one example of the new nonconvex DC cutting plane model for a DC function $f = f^1 - f^2$ with DC components $f^1(x) = \max\{x^2 + x + 2, -x + 2\}$

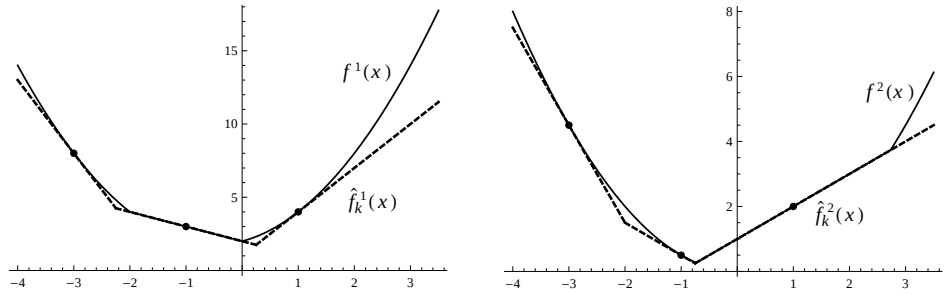


Figure 5.1: The convex cutting plane models of the DC components

and $f^2(x) = \max\{0.5x^2, x+1\}$. This nonconvex function is already used in Figure 4.3 to demonstrate model (4.5) and, thus, linearizations of both DC components are also formed at points -3, -1 and 1. With this selection we obtain the convex cutting plane models of the DC components presented in Figure 5.1. The overall approximation of f is seen in Figure 5.2 and this model is able to describe quite correctly the behaviour of the nonconvex f, even though we use only three points in the model construction. This differs from model (4.5) presented in Figure 4.3, since that model is not able to capture the real structure of f due to the convexity of the model. This supports the conclusion that it is beneficial to use the new nonconvex DC model (5.2) to approximate nonconvex DC functions.

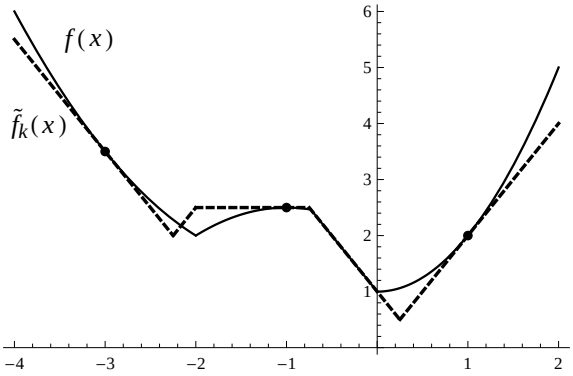


Figure 5.2: The nonconvex DC cutting plane model of the nonconvex DC function

5.2 Direction finding and a new iteration point

To determine the search direction $\boldsymbol{d}_k \in \mathbb{R}^n$, we need to globally minimize the problem

$$\begin{cases} \min & \tilde{f}_k(\boldsymbol{x}_k + \boldsymbol{d}) + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{s.t.} & \boldsymbol{d} \in \mathbb{R}^n. \end{cases} \tag{5.3}$$

Unlike (4.7) this direction finding problem is nonconvex if $|J_k^2| \geq 2$ and this is the price we have to pay for the more accurate model. Therefore, problem (5.3) is more difficult to solve than a convex one since the global minimizer needs to be distinguished from local minimizers. In general, this is not an easy task. However, problem (5.3) has a specific DC structure which makes it possible for us to solve it globally quite easily with an approach described in [82, 83, 107].

In the approach, the main observation is that the cutting plane model (5.2) can be rewritten in a form which enables us to divide problem (5.3) into separate subproblems. Thus, instead of (5.3) we can solve for each $j \in J_k^2$ a nonsmooth convex subproblem

$$\begin{cases} \min & \hat{f}^1(\boldsymbol{x}_k + \boldsymbol{d}) - f^2(\boldsymbol{x}_k) - (\boldsymbol{\xi}_j^2)^T \boldsymbol{d} + \alpha_{j,2}^k + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{s.t.} & \boldsymbol{d} \in \mathbb{R}^n \end{cases} \tag{5.4}$$

and the global minimizer $\boldsymbol{d}_k$ for (5.3) is obtained by selecting the best solution among the subproblem minimizers. In addition, for each $j \in J_k^2$ subproblem (5.4) can be reformulated as a convex quadratic programming problem

$$\begin{cases} \min & v + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{s.t.} & (\boldsymbol{\xi}_i^1 - \boldsymbol{\xi}_j^2)^T \boldsymbol{d} - (\alpha_{i,1}^k - \alpha_{j,2}^k) \leq v \quad \text{for all } i \in J_k^1 \\ & v \in \mathbb{R},\ \boldsymbol{d} \in \mathbb{R}^n \end{cases}$$

and this problem is of the same form as (4.8). Therefore, to obtain the global minimizer $\boldsymbol{d}_k$ we need to solve $|J_k^2|$ smooth subproblems or their dual counterparts. Moreover, the larger the bundle $\mathcal{B}_k^2$ is, the more time-consuming it is to determine the search direction, but at the same time we are able to improve the accuracy of the approximation of the objective f. This shows that the nonconvex DC model increases the computational burden, since for the convex model (4.5) it is enough to solve only one smooth problem (4.8). However, the only requirement for the size of the bundle $\mathcal{B}_k^2$ is that $|J_k^2| \geq 1$. Thus, we can control the amount of computation used.

After the search direction $\boldsymbol{d}_k$ is obtained, we set $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k$ as a new auxiliary point and decide whether we execute a serious step or a null step. Note that we do not need to perform a line search even though the objective f is nonconvex. Another nice feature of the new model $\tilde{f}_k$ is that it provides us the predicted amount of descent

$$\tilde{v}_k = \tilde{f}_k(\boldsymbol{x}_k + \boldsymbol{d}_k) - f(\boldsymbol{x}_k) < 0.$$

Thus, if the decrease in the objective satisfies the condition

$$f(\boldsymbol{y}_{k+1}) - f(\boldsymbol{x}_k) \leq m\tilde{v}_k < 0$$

with the descent parameter $m \in (0, 1/2)$, we perform a serious step by setting $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$ and inserting a new element calculated at $\boldsymbol{x}_{k+1}$ into both bundles $\mathcal{B}_k^1$ and $\mathcal{B}_k^2$. Otherwise a null step is executed and $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$. In this case, we improve the model either by updating the proximity measure t or by inserting a new element obtained at $\boldsymbol{y}_{k+1}$ into $\mathcal{B}_k^1$. A new element can also be inserted into $\mathcal{B}_k^2$, but this insertion is not obligatory and can be done when needed.

5.3 Proximal bundle method PBDC

In this section, we briefly introduce the proximal bundle method (PBDC) designed in Publication I. This method is developed to solve a single-objective unconstrained DC problem (3.1) and it utilizes the DC cutting plane model (5.2) together with the direction finding problem (5.3) to determine a new iteration point.

The execution of the PBDC method is stopped at a point which is either critical or ε-critical. As we have already seen criticality is a weaker optimality condition than Clarke stationarity. However, criticality can be guaranteed easily by using information about DC components and, thus, it can be seen as a natural optimality condition for the PBDC method. In practise, the simplest way to test criticality is to compare subgradients of the DC components at the current iteration point $\boldsymbol{x}_k$. If the difference between those subgradients is smaller than some small stopping tolerance $\delta > 0$ then the point $\boldsymbol{x}_k$ is critical and the algorithm can be stopped. By replacing subgradients with ε-subgradients, we obtain a similar test for ε-criticality.

The basic structure of the PBDC method is illustrated in Figure 5.3. From this flowchart we see that the method has all the characteristic steps used in the standard proximal bundle method (see Figure 4.4). However, since ε-criticality is guaranteed we use two different stopping conditions. The first condition is tested at the beginning of an iteration whenever a serious step has yielded the current iteration point $\boldsymbol{x}_k$ and it simply tests if the point $\boldsymbol{x}_k$ is critical by comparing the arbitrary subgradients of the DC components at $\boldsymbol{x}_k$. The other, so-called approximate stopping condition, is tested if the norm of the search direction $\boldsymbol{d}_k$ is small since in this case we have achieved either ε-criticality or our model is inconsistent. To find out which one of these two options occurs, we remove from the bundles $\mathcal{B}_k^1$ and $\mathcal{B}_k^2$ the elements which are not ε-subgradients at point $\boldsymbol{x}_k$. This way we can construct approximations of the ε-subdifferentials of the DC components at $\boldsymbol{x}_k$. After this we solve the quadratic minimization problem

$$\begin{cases} \min & \|\boldsymbol{\xi}^1 - \boldsymbol{\xi}^2\| \\ \text{s. t.} & \boldsymbol{\xi}^1 \in \operatorname{conv}\left\{\boldsymbol{\xi}_j^1 \,|\, j \in J_k^1\right\} \\ & \boldsymbol{\xi}^2 \in \operatorname{conv}\left\{\boldsymbol{\xi}_j^2 \,|\, j \in J_k^2\right\} \end{cases}$$

and its solution identifies the case, where the approximations of the ε-sudifferentials of the DC components intersect. If ε-criticality is not verified in this procedure, our model needs to be improved by decreasing the proximity measure t and this decrease together with other parameter updates are inspired by [41, 42, 43].

The convergence of the method to an ε-critical point after a finite number of steps is proved. This requires that the level set $\mathcal{F}_0 = \{\boldsymbol{x} \in \mathbb{R}^n \,|\, f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0)\}$ is compact for a starting point $\boldsymbol{x}_0 \in \mathbb{R}^n$ and that the overestimates of Lipschitz constants of the DC components are known. These requirements are not really restrictive since the first one holds for well-defined problems (3.1) whereas the second one is easily satisfied since any overestimate is acceptable.

Numerical experiments in Publication I show the good performance and efficiency of the PBDC method. In addition, the results are compared with three bundle methods [42, 43, 96], DCA [83] and the truncated codifferential method [12] to validate the use of the bundle method explicitly utilizing the DC structure in the model construction. The most interesting observation is that the PBDC is

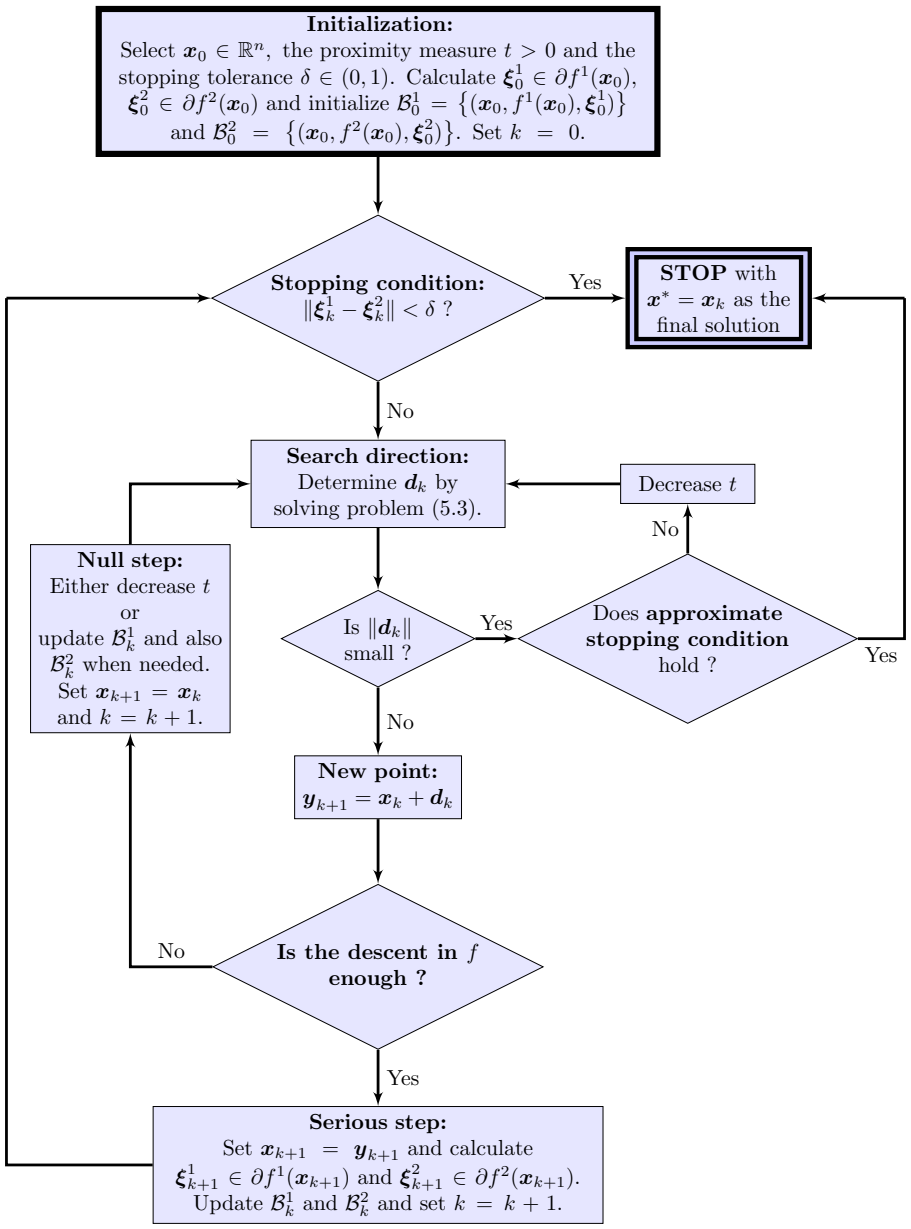


Figure 5.3: The flowchart of the proximal bundle method PBDC

nearly always able to produce the global minimizer or best known solution for test problems. In addition, it outperforms the other tested methods in this feature. Therefore, even though the PBDC is only a local method, it seems that the new DC model is able to capture some relevant information about the objective which helps to avoid local minimizers.

5.4 Double bundle method DBDC

Even though criticality can be easily tested in algorithms utilizing the DC structure, its drawback is that a solution satisfying it does not need to be a local optimum or even a saddle point. Thus, we may stop at a point having this type of an undesirable feature without knowing it and one illustrative example of this is presented in Publication II. In order to avoid this unwanted feature, the double bundle method (DBDC) is developed in Publication II to strengthen the stopping condition used in the PBDC. The DBDC is the successor of the PBDC method that presents a new escaping procedure which either guarantees Clarke stationarity at a candidate solution or generates a descent direction decreasing the value of the objective.

The sketch of the DBDC is shown in Figure 5.4 and this flowchart closely reminds of the one used in the PBDC (see Figure 5.3). The biggest difference is the new escaping procedure which is utilized whenever a promising candidate solution is encountered due to the fulfilment of the criticality condition or the small value of the norm of the search direction. With the escaping procedure we produce a point $\boldsymbol{x}^+$ and if it coincides with $\boldsymbol{x}_k$ we have obtained approximate Clarke stationarity and the DBDC algorithm can be stopped. Otherwise, we execute a serious step since $\boldsymbol{x}^+$ significantly decreases the value of the objective.

The novelty of the escaping procedure lies in its ability to ensure that the difference of subgradients of the DC components belongs to the subdifferential of the original DC function and as we have already seen this is not true in general. Therefore, the selection of subgradients of the DC components needs to be done with care and the main tool in this selection is provided by the directional derivative. First, we know that the directional derivative for a DC component f^i, $i = 1, 2$, at $\boldsymbol{x} \in \mathbb{R}^n$ into a direction $\boldsymbol{d} \in \mathbb{R}^n$ can be obtained with the formula (see, e.g., [10])

$$(f^i)'(\boldsymbol{x}; \boldsymbol{d}) = \max \left\{ \boldsymbol{\xi}^T \boldsymbol{d} \,|\, \boldsymbol{\xi} \in \partial f^i(\boldsymbol{x}) \right\}$$

giving a link to the subdifferential $\partial f^i(\boldsymbol{x})$. In addition, for any $\boldsymbol{d} \in \mathbb{R}^n$, $\boldsymbol{d} \neq \boldsymbol{0}$, we can define by

$$G_i(\boldsymbol{x}; \boldsymbol{d}) = \left\{ \boldsymbol{\xi} \in \partial f^i(\boldsymbol{x}) \,|\, \boldsymbol{\xi}^T \boldsymbol{d} = (f^i)'(\boldsymbol{x}; \boldsymbol{d}) \right\}$$

the set of the subgradients yielding the directional derivative of f^i. One of the main results in Publication II is the following useful theorem:

Theorem 5.2. *Let $f = f^1 - f^2$ be a DC function and $\boldsymbol{x} \in \mathbb{R}^n$. If $G_1(\boldsymbol{x}; \boldsymbol{d}) = \{\boldsymbol{\xi}^1\}$ and $G_2(\boldsymbol{x}; \boldsymbol{d}) = \{\boldsymbol{\xi}^2\}$ are singletons for some $\boldsymbol{d} \in \mathbb{R}^n$ then $\boldsymbol{\xi}^1 - \boldsymbol{\xi}^2 \in \partial f(\boldsymbol{x})$.*

Therefore, if there exists only one subgradient for both DC components yielding the directional derivative, the subgradient of the DC function can be obtained by calculating those specific subgradients. Nevertheless, the result in Theorem 5.2 depends on the direction and does not necessarily hold if this selection is done arbitrarily.

In the escaping procedure presented in Figure 5.5, the aim is to detect whether a candidate solution $\boldsymbol{x}_k$ is Clarke stationary or not. Thus, during the execution of this algorithm we build an approximation U of the Goldstein ε-subdifferential

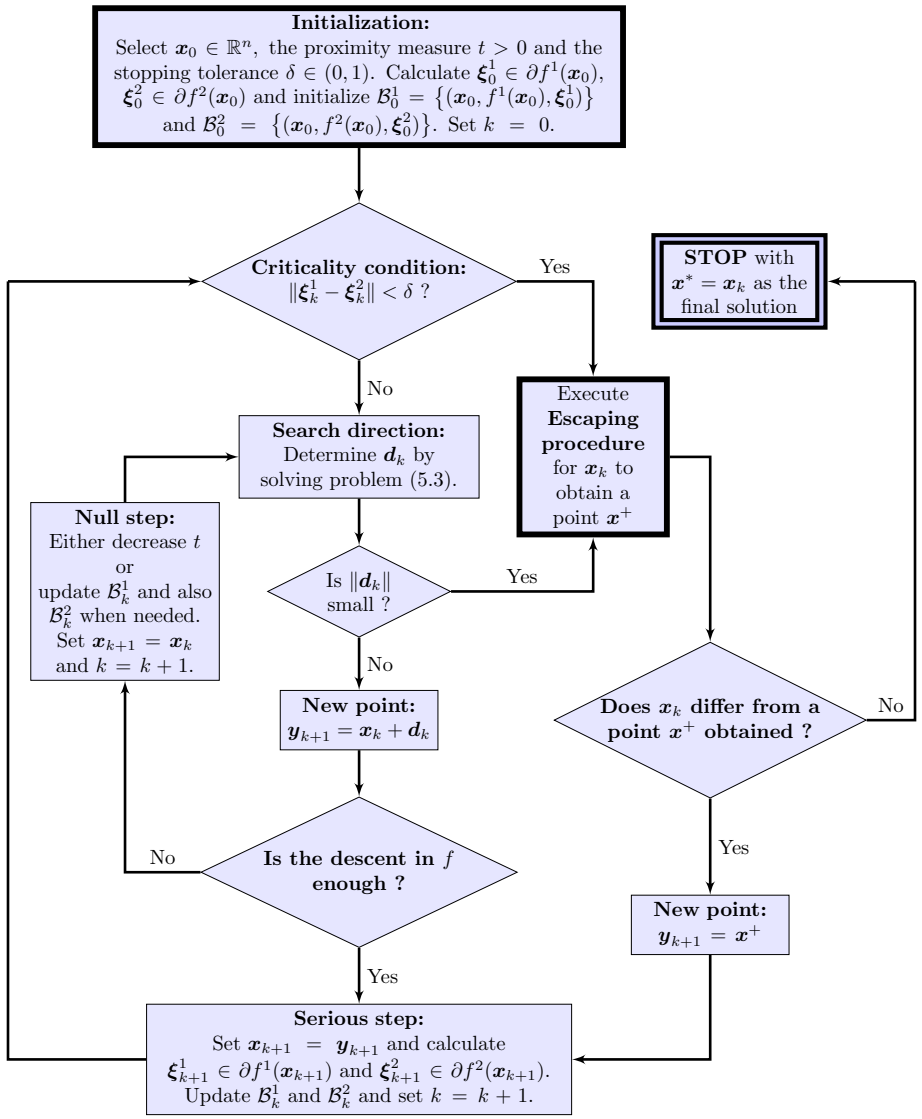


Figure 5.4: The flowchart of the double bundle method DBDC

$\partial_\varepsilon^G f(\boldsymbol{x}_k)$ (see definition (2.2)) of the DC function f for some small $\varepsilon > 0$ such that $U \subseteq \partial_\varepsilon^G f(\boldsymbol{x}_k)$. By solving the norm minimization problem

$$\begin{cases} \min & \frac{1}{2}\|\boldsymbol{u}\|^2 \\ \text{s. t.} & \boldsymbol{u} \in U, \end{cases}$$

we obtain the solution $\boldsymbol{u}^*$ yielding the minimum norm in the approximation of $\partial_\varepsilon^G f(\boldsymbol{x}_k)$. If this minimum norm is smaller than the stopping tolerance $\delta > 0$, then the execution of the escaping procedure can be stopped since $\boldsymbol{u}^*$ is close

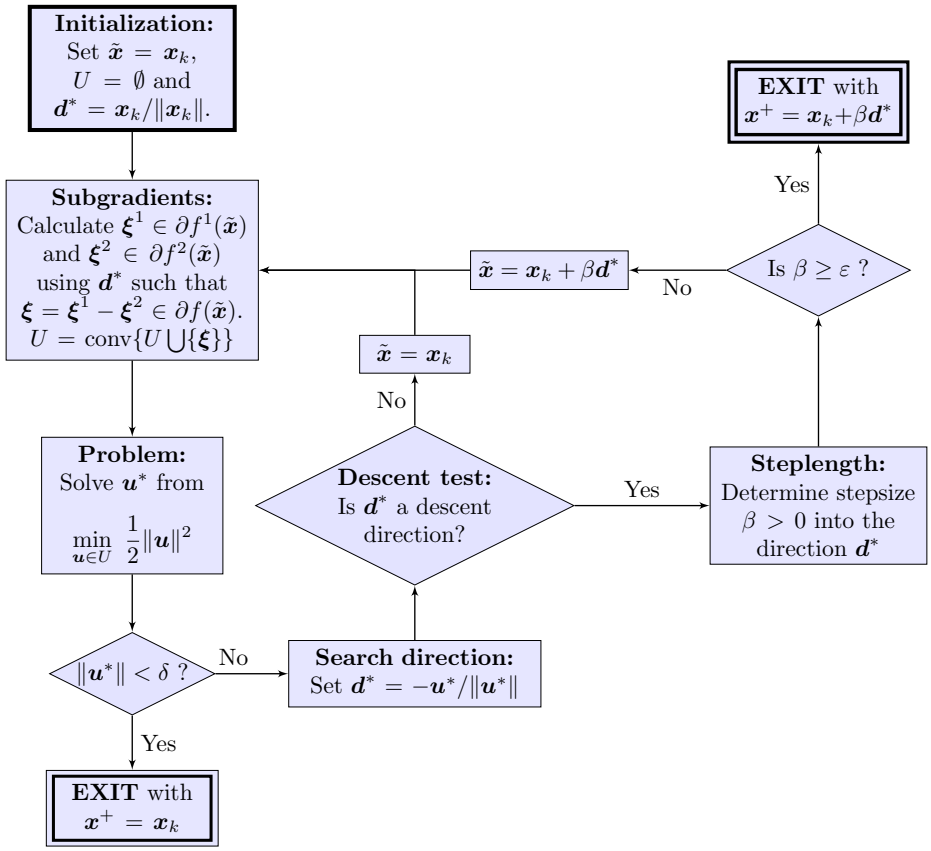


Figure 5.5: The flowchart of the escaping procedure

to the zero vector and yields approximate Clarke stationarity of $\boldsymbol{x}_k$. Otherwise a search direction $\boldsymbol{d}^*$ is formed by utilizing $\boldsymbol{u}^*$. If this new direction gives a sufficient descent in the value of the objective, then the point $\boldsymbol{x}_k$ is not Clarke stationary and we can exit from the procedure with a new better iteration point. Otherwise, the approximation of the Goldstein ε-subdifferential needs to be improved with a new subgradient generated at a point $\tilde{\boldsymbol{x}} \in B(\boldsymbol{x}_k; \varepsilon)$ by using the direction $\boldsymbol{d}^*$ and the DC components. Since the result in Theorem 5.2 may not hold for a fixed direction we show in Publication II that a small controlled change in $\boldsymbol{d}^*$ is always permitted and yields a direction $\bar{\boldsymbol{d}}$ such that the sets $G_1(\tilde{\boldsymbol{x}}; \bar{\boldsymbol{d}})$ and $G_2(\tilde{\boldsymbol{x}}; \bar{\boldsymbol{d}})$ are singletons and $G_i(\tilde{\boldsymbol{x}}; \bar{\boldsymbol{d}}) \subseteq G_i(\tilde{\boldsymbol{x}}; \boldsymbol{d}^*) \subseteq \partial f^i(\tilde{\boldsymbol{x}})$ for $i = 1, 2$. Thus, the subgradient of the objective f can be computed by defining the subgradients of the DC components which give the directional derivatives into the direction $\bar{\boldsymbol{d}}$.

The finite convergence of the DBDC method to an approximative Clarke stationary point is ensured when two assumptions are fulfilled. The first one is the same as in the PBDC requiring that the level set $\mathcal{F}_0 = \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0)\}$ is compact for a starting point $\boldsymbol{x}_0 \in \mathbb{R}^n$. The second one assumes that the subdifferentials of the DC components are polytopes at each $\boldsymbol{x} \in \mathbb{R}^n$ and it is required

when we show that the escaping procedure stops after a finite number of steps. Therefore, the second assumption differs from the one used in the PBDC, but it is not very restrictive since it nearly always holds in practical applications.

The performance of the DBDC is tested in numerical experiments. The comparison with the PBDC shows that the DBDC maintains the good ability of the PBDC to find global minimizers and it even succeeds in finding a better solution than the PBDC method in a couple of test problems. In addition, the DBDC is efficient to solve the test problems, but it sometimes requires a little bit more computational effort than the PBDC. This is mainly due to the new escaping procedure. However, it cannot be seen as a real disadvantage since we are able to guarantee a stronger optimality condition.

5.5 Multiobjective double bundle method MDBDC

In the two previous sections, we have presented methods for single-objective and unconstrained DC optimization. Therefore, to cover a wider set of problems, the multiobjective double bundle method (MDBDC) is designed in Publication III to extend the DBDC also to multiobjective and constrained DC problems (3.6). It is worth to note that this extension can be used to solve also single-objective DC problems with DC inequality constraints.

To solve problem (3.6), we need to handle several objectives and inequality constraints. A useful strategy for this is to utilize the improvement function [71, 123] which is employed, for example, in the multiobjective proximal bundle method (MPB) [95, 97] inspiring the framework used in the MDBDC. For problem (3.6), the *improvement function* $H : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is defined by

$$H(\boldsymbol{x}, \boldsymbol{y}) = \max\{f_i(\boldsymbol{x}) - f_i(\boldsymbol{y}), g_l(\boldsymbol{x}) \,|\, i \in I,\, l \in L\}.$$

This function has three important elementary properties giving a justification for its use.

Theorem 5.3. *[95, 123] The improvement function $H(\cdot, \boldsymbol{y})$ has the following properties:*

(i) If the solution $\boldsymbol{x}^* \in X$ is globally weakly Pareto optimal for problem (3.6), then

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \mathbb{R}^n} H(\boldsymbol{x}, \boldsymbol{x}^*).$$

(ii) If $\mathbf{0} \in \partial H(\boldsymbol{x}^*, \boldsymbol{x}^*)$ then the solution $\boldsymbol{x}^* \in X$ is weakly Pareto stationary for problem (3.6).

(iii) If $H(\boldsymbol{x}, \boldsymbol{y}) < H(\boldsymbol{y}, \boldsymbol{y})$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in X$ then $f_i(\boldsymbol{x}) < f_i(\boldsymbol{y})$ for all $i \in I$ and $g_l(\boldsymbol{x}) < 0$ for all $l \in L$.

The properties (i) and (ii) of Theorem 5.3 give a link between the original problem (3.6) and the improvement function. Therefore, the aim is to modify the single-objective DBDC method to find a Clarke stationary solution $\boldsymbol{x}^* \in X$ for the improvement function $H(\cdot, \boldsymbol{x}^*)$ (i.e. $\mathbf{0} \in \partial H(\boldsymbol{x}^*, \boldsymbol{x}^*)$) since such a solution is

weakly Pareto stationary for the original multiobjective problem (3.6). For this reason, we determine the search direction $\boldsymbol{d}_k \in \mathbb{R}^n$ at the current iteration point $\boldsymbol{x}_k$ by solving the single-objective unconstrained problem

$$\begin{cases} \min & H(\boldsymbol{x}_k + \boldsymbol{d}, \boldsymbol{x}_k) \\ \text{s.t.} & \boldsymbol{d} \in \mathbb{R}^n. \end{cases} \tag{5.5}$$

Additionally, the last property of Theorem 5.3 guarantees that a descent direction for the improvement function is also a descent direction for the original problem (3.6) in a sense that it decreases the values of all the objectives simultaneously and also maintains feasibility. Thus, the benefit of the improvement function springs from the ability to state the multiobjective and constrained problem as an unconstrained single-objective one and at the same time maintain a useful link between the problem and its transformation.

In order to solve problem (5.5), we need to build a model approximating the nonsmooth objective. First, we notice that the improvement function $H(\cdot, \boldsymbol{y})$ is a DC function by using the case (ii) of Proposition 3.2 and for such a function the DC decompostion is easily formulated as in [57]. To write it, we let the DC decompositions of f_i and g_l be $f_i = f_i^1 - f_i^2$ for $i \in I$ and $g_l = g_l^1 - g_l^2$ for $l \in L$. With this selection the DC decomposition of $H(\cdot, \boldsymbol{y})$ can be presented in the form

$$H(\boldsymbol{x}, \boldsymbol{y}) = H^1(\boldsymbol{x}, \boldsymbol{y}) - H^2(\boldsymbol{x})$$

with DC components

$$\begin{aligned} H^1(\boldsymbol{x}, \boldsymbol{y}) &= \max\{A_i(\boldsymbol{x}, \boldsymbol{y}), B_l(\boldsymbol{x}) \,|\, i \in I,\, l \in L\} \quad \text{and} \\ H^2(\boldsymbol{x}) &= \sum_{i \in I} f_i^2(\boldsymbol{x}) + \sum_{l \in L} g_l^2(\boldsymbol{x}), \end{aligned}$$

where

$$\begin{aligned} A_i(\boldsymbol{x}, \boldsymbol{y}) &= f_i^1(\boldsymbol{x}) + \sum_{\substack{j \in I \\ j \neq i}} f_j^2(\boldsymbol{x}) + \sum_{t \in L} g_t^2(\boldsymbol{x}) - f_i(\boldsymbol{y}) \quad \text{and} \\ B_l(\boldsymbol{x}) &= g_l^1(\boldsymbol{x}) + \sum_{\substack{t \in L \\ t \neq l}} g_t^2(\boldsymbol{x}) + \sum_{i \in I} f_j^2(\boldsymbol{x}). \end{aligned}$$

Therefore, we can utilize a DC cutting plane model resembling (5.2) since both DC components $H^1(\cdot, \boldsymbol{y})$ and H^2 are convex with respect to $\boldsymbol{x}$ and the vector $\boldsymbol{y}$ is treated as a constant.

To construct a model at the current iteration point $\boldsymbol{x}_k \in \mathbb{R}^n$, we assume that each f_i and g_l satisfies Assumption 5.1. Thus, we calculate the corresponding information about DC components for each objective and constraint at an auxiliary point $\boldsymbol{y}_j \in \mathbb{R}^n$. With it we can easily compute the values of $A_i(\boldsymbol{y}_j, \boldsymbol{x}_k)$, $B_l(\boldsymbol{y}_j)$, $H^1(\boldsymbol{y}_j, \boldsymbol{x}_k)$ and $H^2(\boldsymbol{y}_j)$ and their subgradients by using the subdifferential calculus rules of convex functions [10]. This shows that at any $\boldsymbol{y}_j \in \mathbb{R}^n$ we obtain with a small effort a linearization for each $A_i(\cdot, \boldsymbol{x}_k)$ and B_l. Therefore, we form a bundle (4.1) for each $A_i(\cdot, \boldsymbol{x}_k)$ and B_l which together constitute the bundle $\mathcal{B}_k^1$ of $H^1(\cdot, \boldsymbol{x}_k)$. Since we also get one linearization for the second DC component

H^2 at $\boldsymbol{y}_j$ we can directly construct the bundle $\mathcal{B}_k^2$ for H^2 with formula (4.1) and this bundle does not differ from (5.1) used for the second DC component in model (5.2).

The separate bundles of $A_i(\cdot, \boldsymbol{x}_k)$ and B_l mean that we can construct the classical convex cutting plane model (4.2) for each $A_i(\cdot, \boldsymbol{x}_k)$ and B_l at the current iteration point $\boldsymbol{x}_k$. In the following, they are denoted by $\hat{A}_i^k$ and $\hat{B}_l^k$, respectively. Therefore, the convex cutting plane model of the first DC component is

$$\hat{H}_k^1(\boldsymbol{x}) = \max\{\hat{A}_i^k(\boldsymbol{x}), \hat{B}_l^k(\boldsymbol{x}) \mid i \in I, l \in L\}.$$

However, the treatment of the second DC component H^2 does not differ from Section 5.1 and, thus, its approximation $\hat{H}_k^2$ is the convex cutting plane model (4.2). By combining the approximations of the DC components, the DC cutting plane model of the improvement function is

$$\tilde{H}_k(\boldsymbol{x}) = \hat{H}_k^1(\boldsymbol{x}) - \hat{H}_k^2(\boldsymbol{x}).$$

This approximation has the same structure as model (5.2), but the main difference is that we maintain a more accurate approximation of the first DC component. In addition, the search direction problem

$$\begin{cases} \min & \tilde{H}_k(\boldsymbol{x}_k + \boldsymbol{d}) + \frac{1}{2t}\|\boldsymbol{d}\|^2 \\ \text{s. t.} & \boldsymbol{d} \in \mathbb{R}^n \end{cases} \tag{5.6}$$

closely reminds of (5.3) and, especially, the global solution $\boldsymbol{d}_k \in \mathbb{R}^n$ can be computed in the same way as in Section 5.2.

The flowchart of the MDBDC method is presented in Figure 5.6. By comparing the MDBDC and DBDC (see Figure 5.4), we notice that the overall structure of these methods is identical, if we leave out of consideration the update of the proximity measure t after a serious step in the MDBDC. This is due to the utilization of the improvement function, since it enables us to nearly directly use the DBDC method to minimize the unconstrained single-objective problem with the DC cutting plane model of the improvement function as the objective. In addition, the escaping procedure is executed like in Figure 5.5, but instead of f we use the improvement function $H(\cdot, \boldsymbol{x}_k)$. Therefore, the MDBDC stops at a point $\boldsymbol{x}^* \in \mathbb{R}^n$ which is Clarke stationary for the improvement function. Together with the case (*ii*) of Theorem 5.3 this shows that the solution $\boldsymbol{x}^*$ is weakly Pareto stationary for the original problem (3.6) and $\boldsymbol{x}^* \in X$.

The convergence of the MDBDC is proved to a weakly Pareto stationary point for problem (3.6) after a finite number of steps. In the unconstrained single-objective case, we are also able to show a similar result, but the solution obtained is Clarke stationary. Both convergence results require the fulfilment of two assumptions and they resemble closely the ones used in the DBDC. First, the level set $\mathcal{F}_0 = \{\boldsymbol{x} \in X \mid f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}_0) \text{ for all } i \in I\}$ defined at a starting point $\boldsymbol{x}_0 \in X$ needs to be compact. Second, we assume that the subdifferentials $\partial H^1(\boldsymbol{x}, \boldsymbol{y})$ and $\partial H^2(\boldsymbol{x})$ are polytopes at each $\boldsymbol{x} \in \mathbb{R}^n$.

In the numerical experiments, the MDBDC is compared to the multiobjective proximal bundle method MPB [95, 97] having a somewhat similar structure than the MDBDC. Additionally, the MPB is designed for general multiobjective problems. The comparison of the numerical results show that the MDBDC has a good

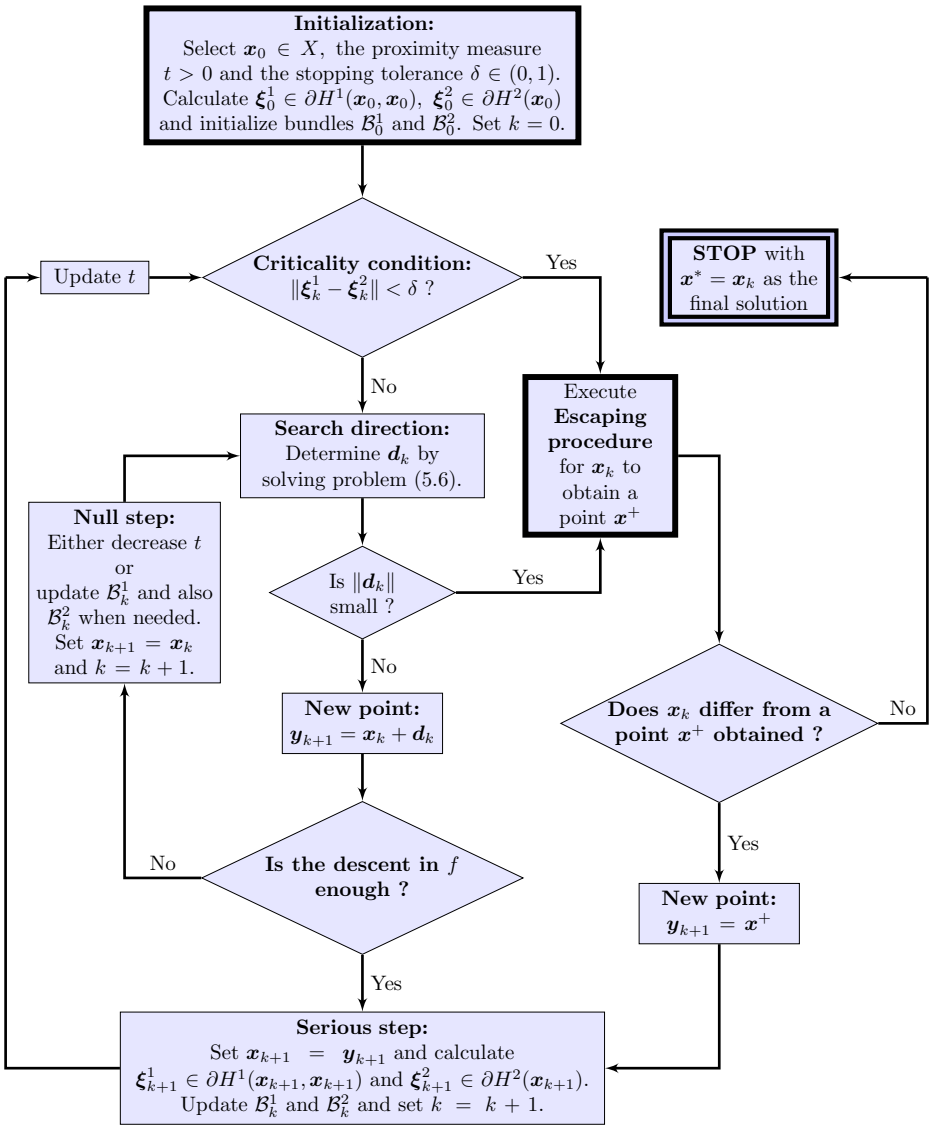


Figure 5.6: The flowchart of the multiobjective double bundle method MDBDC

performance and it yields in 30 % of the test problems a better solution (i.e. every objective has a better value) than the one obtained with the MPB. Furthermore, in the large test problems ($n > 100$) the MDBDC uses nearly always significantly less computational effort than the MPB. All in all, we can conclude that, if the DC structure is known, it is beneficial to use the MDBDC method specially designed for DC functions instead of the MPB method developed for general nonconvex functions.

Chapter 6

Utilizing concavity and convexity in bundling

We have already seen that using the DC decomposition of the objective function we can improve the performance of the bundle method and yield an accurate approximation of the objective. Unfortunately, the useful DC structure is not always available since we may be unable to write it or the objective function is not a DC function. Thus, in such cases we cannot use the most evident way to distinguish the convex and concave behaviour of the objective function. At the same time we may face also a situation that an optimization problem has a large number of variables. Therefore, the computational demand of the bundle methods presented in Chapters 4 and 5 may grow to be too high and cause a failure in the solution process. Due to this, it is important to have also tools to handle large-scale problems.

In this chapter, we introduce another type of approach, which utilizes the linearization error measuring the goodness of a linearization at the current iteration point. This allows us to get some information about the convexity and concavity of the objective function at the current iteration, even though the structure of the objective may be unknown. In particular, if the linearization error is negative then the linearization overestimates the objective function and the corresponding auxiliary point exhibits a "concave" behaviour with respect to the current iteration point. Similarly the positive value of the linearization error means that we have an underestimate of the objective function at the current iteration point and the auxiliary point captures a "convex" behaviour. Therefore, by splitting the auxiliary points according to the sign of the linearization error, we can construct two sets of points, which somehow capture implicitly local information about the DC structure of the objective. In addition, the sizes of these sets can be kept limited, which makes the approach suitable also for large-scale optimization.

Next, we are going to present how this type of divided information can be utilized in the splitting metrics diagonal bundle method (SMDB) originally developed in Publication IV. This bundle method is designed for the unconstrained

nonsmooth optimization problem of the form

$$\begin{cases} \min & f(\boldsymbol{x}) \\ \text{s. t.} & \boldsymbol{x} \in \mathbb{R}^n, \end{cases}$$

where the objective $f : \mathbb{R}^n \to \mathbb{R}$ is semismooth (see, e.g., [18]) and the dimension n is assumed to be large. This differs from the previous bundle methods since they are developed only for small- and medium-scale problems. In addition, instead of the DC structure required in the previous methods we now assume the semismoothess of the objective.

6.1 Splitting the information in direction finding

The aim of the SMDB is to solve large-scale problems. However, when the dimension of the problem grows, the computational demand in the quadratic direction finding problem (4.8) expands and this problem can be too complex and time-consuming to be solved. Therefore, to handle large-scale problems we exploit the same ideas as limited memory bundle methods [50, 51, 67]. This enables us to utilize some characteristic features of the standard bundle methods, namely, null steps, serious steps and a subgradient aggregation. In addition, the search direction can be calculated by using a limited memory approach constructing an approximation of the inverse of the Hessian matrix of the objective function and, thus, the time-consuming quadratic direction finding problem does not need to be solved. To form the approximation of the inverse matrix of the Hessian, we especially utilize the diagonal update formula used in the diagonal bundle method (D-bundle [67]) since this method has a good ability to handle the sparsity of the problem together with large dimensionality. Another benefit is that for the diagonal update formula it is easy to check positive (or negative) definiteness of the generated matrices.

In order to construct the approximation of the inverse of the Hessian, we need some information about the objective function and, characteristic to bundle methods, we assume that Assumption 4.1 holds. Therefore, whenever we obtain a new auxiliary point $\boldsymbol{y}_{k+1} \in \mathbb{R}^n$ at the current iteration point $\boldsymbol{x}_k \in \mathbb{R}^n$, we can generate a function value $f(\boldsymbol{y}_{k+1})$ and a subgradient $\boldsymbol{\xi} \in \partial f(\boldsymbol{y}_{k+1})$. This information provides us the linearization error

$$\alpha_{k+1} = f(\boldsymbol{x}_k) - f(\boldsymbol{y}_{k+1}) - \boldsymbol{\xi}_{k+1}^T(\boldsymbol{x}_k - \boldsymbol{y}_{k+1}),$$

which is used in the SMDB method to detect the "convex" and the "concave" behaviour of the objective function. More precisely, the point is said to exhibit a "convex" or "concave" behaviour with respect to $\boldsymbol{x}_k$ according to the positive or negative sign of the linearization error α_{k+1}, respectively. Therefore, as in [41, 46], we use the sign of the linearization error to split the information into two separate sets. This way we can maintain some information about the nonconvex structure of the objective, since we do not directly perform the somewhat arbitrary downward shifting by replacing linearization errors with subgradient locality measures.

Typically limited memory bundle methods, like the D-bundle, construct only one approximation of the inverse of the Hessian matrix during each iteration.

However, since we divide the auxiliary points into two sets based on the sign of the linearization error, we want to generate different metrics for the convex and concave behaviour of the objective. Therefore, the SMDB maintains two different approximations of the inverse of the Hessian matrix. These approximations are diagonal matrices and at the current iteration point $\boldsymbol{x}_k$ they are denoted by $\boldsymbol{D}_k^+ \in \mathbb{R}^{n\times n}$ and $\boldsymbol{D}_k^- \in \mathbb{R}^{n\times n}$, where the supercript tells whether the points with positive or negative linearization errors are used to generate them. The limited memory diagonal update $\boldsymbol{D}_k^+$ is called the "convex approximation" whereas $\boldsymbol{D}_k^-$ is the "concave approximation". In order to save storage space, both matrices are constructed from the diagonal update formula introduced in [55] requiring only few correction vectors to represent them. Therefore, instead of the whole matrices, it is sufficient to store and manipulate for each approximation its own set of the correction vectors describing the essential knowledge about objective function values and subgradients. In addition, $\boldsymbol{D}_k^+$ is required to be positive definite whereas $\boldsymbol{D}_k^-$ needs to be negative definite. Thus, depending on whether we calculate $\boldsymbol{D}_k^+$ or $\boldsymbol{D}_k^-$, we add the requirement about positive or negative definiteness as a constraint into the problem.

In the SMDB method, the diagonal approximations are used to calculate the search direction. The "convex approximation" $\boldsymbol{D}_k^+$ is used during iteration k, if the linearization error is nonnegative or the previous step was a serious step. In this case, the search direction is

$$\boldsymbol{d}_k = -\boldsymbol{D}_k^+ \tilde{\boldsymbol{\xi}}_k,$$

where $\tilde{\boldsymbol{\xi}}_k$ is an aggregated subgradient of the objective function. This subgradient is a convex combination of three previously calculated subgradients and it is obtained with a procedure presented in [50, 51]. Therefore, the number of stored subgradients can be kept limited since it is enough to store only three subgradients. This is a big benefit compared to the standard bundle method, which typically requires that the number of stored subgradients needs to grow with the dimension of the problem. The "concave approximation" $\boldsymbol{D}_k^-$, in turn, is utilized if the linearization error is negative. However, $\boldsymbol{D}_k^-$ cannot be directly used, but we need to determine the smallest value $p_k \in (0, 1)$ such that the convex combination $p_k\boldsymbol{D}_k^+ + (1 - p_k)\boldsymbol{D}_k^-$ remains positive definite. Thus, we merge the concave and convex information and the search direction is obtained by the formula

$$\boldsymbol{d}_k = -(p_k\boldsymbol{D}_k^+ + (1 - p_k)\boldsymbol{D}_k^-)\tilde{\boldsymbol{\xi}}_k.$$

6.2 Diagonal bundle method SMDB

The simplified structure of the splitting metrics diagonal bundle method (SMDB) combining the D-bundle method [67] with the different usage of the metrics is presented in Figure 6.1. Therefore, in some steps we utilize the knowledge about the convex and concave behaviour of the objective function. However, the main structure of the SMDB resembles the one used in the standard bundle method (see Figure 4.4) and the execution of the SMDB is characterized with the usage of null and serious steps.

In order to decide whether we perform a serious step or a null step after the search direction is computed, we need to generate an auxiliary point $\boldsymbol{y}_{k+1}$. To

obtain it, we perform a specific line search procedure presented in [50, 122] yielding the stepsize $t_L > 0$ and we set $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + t_L \boldsymbol{d}_k$. A necessary condition for the serious step is that the decrease of the objective function is sufficient and, in this case, we can set $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1}$. In order to validate this, we calculate the predicted descent v_k from a specific formula utilizing the diagonal approximation $\boldsymbol{D}_k^+$ and test if the descent condition (4.9) holds. Otherwise we perform a null step and $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$. In this case, we calculate a new aggregated subgradient and improve the diagonal approximations by computing either a convex or concave update.

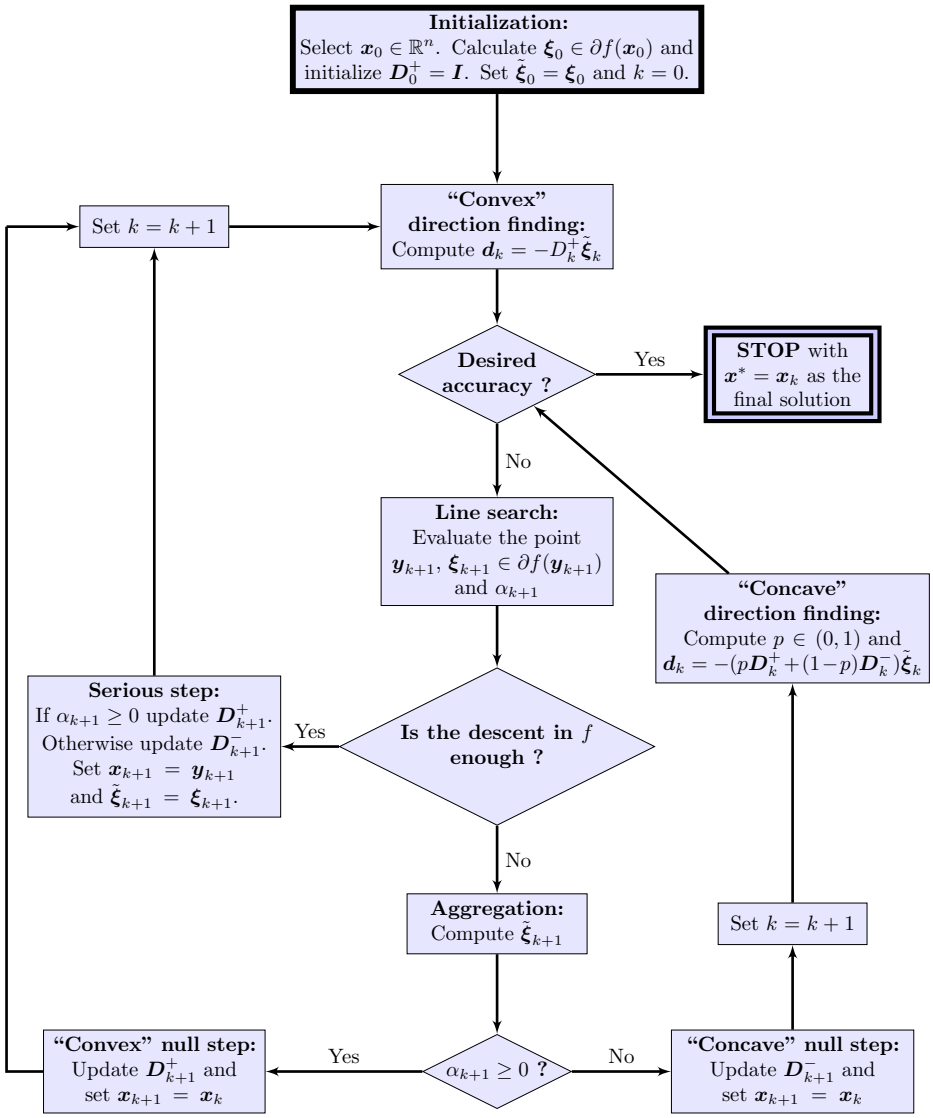


Figure 6.1: The simplified flowchart of the diagonal bundle method SMDB

The convergence result of the SMDB method shows that each accumulation point of the sequence $\{\boldsymbol{x}_k\}$ is Clarke stationary for the objective function. To prove this two assumptions are required to hold. First, the objective function f needs to be semismooth (see, e.g., [18]) and this is a typical requirement in nonconvex bundle methods. Second, the level set $\mathcal{F}_0 = \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0)\}$ is assumed to be compact for the starting point $\boldsymbol{x}_0 \in \mathbb{R}^n$ used in the algorithm. Thus, the latter assumption is the same as the one used in the bundle methods presented in Chapter 5. However, semismoothness is not needed in those bundle methods due to the usage of the DC structure. Instead, we need an assumption about the DC components.

To validate the efficiency of the new SMDB method, we have used in our numerical experiments a set of large-scale nonsmooth minimization problems with up to one million variables. In addition, the results of the SMDB method are compared with the ones obtained from two other limited memory bundle methods LMBM [50, 51] and D-bundle [67] to find out how the new splitting metrics procedure affects the performance of the SMDB. The results show that the new procedure is as efficient as the other methods developed for large-scale problems. In addition, it is able to give some advantage over the traditional usage of the subgradient locality measures in the nonconvex case. Therefore, we can conclude that the SMDB method is a good option to solve nonsmooth optimization problems in the large-scale setting and it can be used to solve also extremely large-scale problems.

Chapter 7

Application to clusterwise linear regression

Nowadays we are able produce and collect huge amounts of data. Therefore, there exists an urgent need to analyse data efficiently and to transform it to useful information. To answer this demand, data mining has developed to be an important field providing efficient tools to classify and highlight useful features of data and give support to decision making. In addition, optimization plays a crucial role in this field since several applications of data mining can be modelled as an optimization problem.

Data classification problems include, for example, clustering [79] and regression analysis [49]. Clustering is an unsupervised classification technique, where the aim is to divide a data set into subsets of similar objects in order to recognize patterns in data. It is one of the most fundamental tasks in data mining and it is applicable in nearly all areas of research including, for example, medical sciences [17, 121], computer sciences [24, 63] and economics [80]. Regression analysis, in turn, concentrates on fitting a regression function to a data set to estimate the dependencies between variables. Therefore, regression analysis can be used to discover trends in data and it has numerous applications which cover different disciplices such as medical science [16], finance [32] and history [27], to name a few. When we have some preliminary knowledge about a data set, we can select some specific structure for the regression function. For example, in linear regression we adjust a hyperplane to a data set. Thus, linear regression consists of finding a linear relationship describing how one or more variables vary as a function of another variable.

By combining clustering and linear regression analysis, we obtain another classification problem, the so-called clusterwise linear regression (CLR). The goal in CLR is to simultaneously divide a data set into mutually exclusive subsets (or clusters) and to form a hyperplane for each cluster to approximate it. CLR has several applications including, for example, the consumer benefit segmentation [124], market segmentation stock-exchange [112], metal inert gas welding process [44], rainfall prediction [11] and PM10 prediction [110].

In this chapter, we review shortly a new approach to solve CLR problems originally presented in Publication V. The novelty of this method relies on the utiliza-

tion of the support vector machines (SVM) approach to model the CLR problem. This way we obtain a new nonsmooth DC formulation of the CLR problem, which is solved by combining the DBDC method with an incremental algorithm [13]. In the following, we give a more detailed description of the approach.

7.1 SVM for clusterwise linear regression

In CLR, we consider a given data set

$$A = \{(\boldsymbol{a}^i, b_i) \in \mathbb{R}^n \times \mathbb{R} \,|\, i = 1, \ldots, m\}.$$

The goal is to partition this set into k clusters and at the same time fit a hyperplane to each cluster to capture a linear trend in that subset of data. Moreover, each point in A needs to be assigned to exactly one cluster in such a way that each cluster is nonempty. Our aim is to formulate a model capable of solving this problem efficiently.

In many CLR models (see, e.g., [13, 14, 25]), the deviation of the point $(\boldsymbol{a}^i, b_i) \in A$ from the closest hyperplane

$$f(\boldsymbol{a}) = \boldsymbol{a}^T \boldsymbol{x} + y \tag{7.1}$$

with coefficients $\boldsymbol{x} \in \mathbb{R}^n$ and $y \in \mathbb{R}$ is penalized, if the point is not located exactly on this hyperplane. This means that even small perturbations are taken into account, even though in a larger scale they may be nearly insignificant. In addition, many data sets typically contain some noise. Therefore, it would be more reasonable to allow small perturbations from the hyperplanes without penalizing them. This kind of idea is utilized in the SVM for the regression method [31, 116], where one hyperplane is fitted to the data set A. In order to generalize this approach to the CLR problems, we first introduce briefly the SVM approach for linear regression.

In ε-SVM linear regression, we are looking for one hyperplane of the form (7.1) to approximate the set A with a precision $\varepsilon > 0$. Therefore, the coefficients $\boldsymbol{x}$ and y of the hyperplane need to be determined in such a way that at each point $(\boldsymbol{a}^i, b_i) \in A$ the deviation between $f(\boldsymbol{a}_i)$ and the obtained target b_i does not exceed the tolerance ε. Moreover, the hyperplane should be as flat as possible, meaning that we minimize the norm of $\boldsymbol{x}$. This regression problem can be modelled as a constrained nonsmooth optimization problem

$$\begin{cases} \min & \frac{1}{2}\|\boldsymbol{x}\|^2 \\ \text{s.t.} & |\boldsymbol{x}^T \boldsymbol{a}^i + y - b_i| \leq \varepsilon, \quad i = 1, \ldots, m \\ & \boldsymbol{x} \in \mathbb{R}^n,\ y \in \mathbb{R}. \end{cases} \tag{7.2}$$

The disadvantage of this formulation is that it is does not necessarily have any feasible solution, if there does not exist any hyperplane approximating all data points with the required accuracy ε. Thus, it is beneficial to relax the formulation to guarantee feasibility. This can be done by utilizing the penalty function approach [127], which enables us to rewrite (7.2) as an unconstrained convex non-

smooth minimization problem

$$\begin{cases} \min & \frac{1}{2}\|\boldsymbol{x}\|^2 + C\sum_{i=1}^{m} \max\left(0, \left|\boldsymbol{x}^T\boldsymbol{a}^i + y - b_i\right| - \varepsilon\right) \\ \text{s. t.} & \boldsymbol{x} \in \mathbb{R}^n,\ y \in \mathbb{R}, \end{cases} \tag{7.3}$$

where $C > 0$ is a regularization or penalty parameter. We see that similar to (7.2) small perturbations are not punished in the model. However, the set of acceptable solutions is enlarged and this yields more "freedom" in the solution process.

In the CLR problem, our aim is similar, but we adjust k hyperplanes to approximate the data set A with a precision $\varepsilon > 0$. Let $\boldsymbol{x}^j$ and y_j be the coefficients of the jth hyperplane. Therefore, the vectors $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k)^T$ and $\boldsymbol{y} = (y_1, \ldots, y_k)^T$ represent the combined values of the regression coefficients of the hyperplanes. By utilizing the same idea as in (7.2), the *SVM-CLR problem* can be stated in the form

$$\begin{cases} \min & \frac{1}{2}\sum_{j=1}^{k} \|\boldsymbol{x}^j\|^2 \\ \text{s. t.} & \min\limits_{j=1,\ldots,k} \left|(\boldsymbol{x}^j)^T\boldsymbol{a}^i + y_j - b_i\right| \le \varepsilon, \quad i = 1, \ldots, m \\ & \boldsymbol{x} \in \mathbb{R}^{nk},\ \boldsymbol{y} \in \mathbb{R}^k. \end{cases}$$

In the spirit of (7.3), we can rewrite it as an unconstrained nonsmooth minimization problem

$$\begin{cases} \min & F_k(\boldsymbol{x}, \boldsymbol{y}) \\ \text{s. t.} & \boldsymbol{x} \in \mathbb{R}^{nk},\ \boldsymbol{y} \in \mathbb{R}^k, \end{cases} \tag{7.4}$$

where the objective function is

$$F_k(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\sum_{j=1}^{k} \|\boldsymbol{x}^j\|^2 + C\sum_{i=1}^{m} \max\left(0, \min_{j=1,\ldots,k} \left|(\boldsymbol{x}^j)^T\boldsymbol{a}^i + y_j - b_i\right| - \varepsilon\right). \tag{7.5}$$

Unlike (7.3) problem (7.4) is nonconvex and this is due to the fact that we fit several regression functions simultaneously.

One benefit of the new SVM-CLR formulation (7.4) is that it allows us to omit small perturbations in the model unlike in many other CLR formulations. In addition, the objective function (7.5) is a DC function and, thus, we can utilize the DC structure to obtain a solution. In order to write the DC representation, we start with denoting by

$$e_i(\boldsymbol{x}^j, y_j) = \left|(\boldsymbol{x}^j)^T\boldsymbol{a}^i + y_j - b_i\right|$$

the error of the point $(\boldsymbol{a}^i, b_i) \in A$ from the jth hyperplane. This enables us to write the DC representation of (7.5) in the form

$$F_k(\boldsymbol{x}, \boldsymbol{y}) = F_k^1(\boldsymbol{x}, \boldsymbol{y}) - F_k^2(\boldsymbol{x}, \boldsymbol{y})$$

with DC components

$$F_k^1(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\sum_{j=1}^{k} \|\boldsymbol{x}^j\|^2 + C\sum_{i=1}^{m} \max\left\{\sum_{j=1}^{k} e_i(\boldsymbol{x}^j, y_j), \max_{j=1,\ldots,k} \sum_{t=1, t\neq j}^{k} e_i(\boldsymbol{x}^t, y_t) + \varepsilon\right\}$$

and $F_k^2(\boldsymbol{x}, \boldsymbol{y}) = C\sum\limits_{i=1}^{m} \left(\max\limits_{j=1,\ldots,k} \sum\limits_{t=1, t\neq j}^{k} e_i(\boldsymbol{x}^t, y_t) + \varepsilon\right)$.

7.2 Double bundle method DB-SVM-CLR

To solve the SVM-CLR problem (7.4), the modified double bundle method (DB-SVM-CLR) is introduced in Publication V. The method is inspired by the DBDC method presented in Section 5.4 offering an efficient way to utilize the DC structure of (7.4) in the solution process. However, the SVM-CLR problem has several local solutions and, thus, the aim is to locate global or near global ones, since they are the most useful and significant solutions. This is in general a challenging task and it strongly depends on the choice of the starting point. Therefore, in order to provide good starting points, the DB-SVM-CLR method combines the DBDC with a modified version of the incremental algorithm [13].

The main idea in the incremental algorithm is to build clusters as well as hyperplanes incrementally. This means that the solution of the $(k-1)$th SVM-CLR problem, which fits $k-1$ hyperplanes into the data set, is used to produce good starting points for the kth SVM-CLR problem. In this step, the central role is on the so-called kth auxiliary SVM-CLR problem presented in Publication V. In the kth auxiliary problem, the best position of one new hyperplane is searched among the hyperplanes obtained from the solution of the $(k-1)$th SVM-CLR problem. This auxiliary problem is also a DC problem and it is easer and less time-consuming to solve than the original problem. The auxiliary problem is solved starting from several different initial points and this provides a set of starting points for the kth SVM-CLR problem. Thus, we may obtain several different solutions for the kth SVM-CLR problem. After the best solution is selected we can continue the incremental algorithm by adding a new hyperplane.

The DB-SVM-CLR method is presented in Figure 7.1 highlighting the basic structure of the incremental algorithm. First, the method is started by adjusting one hyperplane to the whole data set and, in this case, the corresponding optimization problem is actually convex and easy to solve. After this a new hyperplane is added one at a time until the required number of hyperplanes is reached. Therefore, by solving the kth SVM-CLR problem, we obtain as a by-product a solution for each smaller SVM-CLR problem. In addition, whenever the auxiliary SVM-CLR problem or the SVM-CLR problem is solved we use the DBDC method introduced in Section 5.4 since both problems are unconstrained and have a DC function as the objective. The sets S_1 and S_2 are used to denote the starting points for the auxiliary SVM-CLR problem and the SVM-CLR problem, respectively. For a more detailed description of the selection of these sets see [13].

The solution for each SVM-CLR problem solved during the execution of the DB-SVM-CLR is proved to be approximately Clarke stationary. This result directly follows from the convergence result of the DBDC method and the fact that the DBDC is applied a finite number of times during the process of adjusting new hyperplanes. Therefore, the convergence of the DB-SVM-CLR method requires the fulfilment of the same two assumptions as the DBDC. However, the formulation of the auxiliary SVM-CLR problem and the SVM-CLR problem guarantees that both assumptions are trivially satisfied.

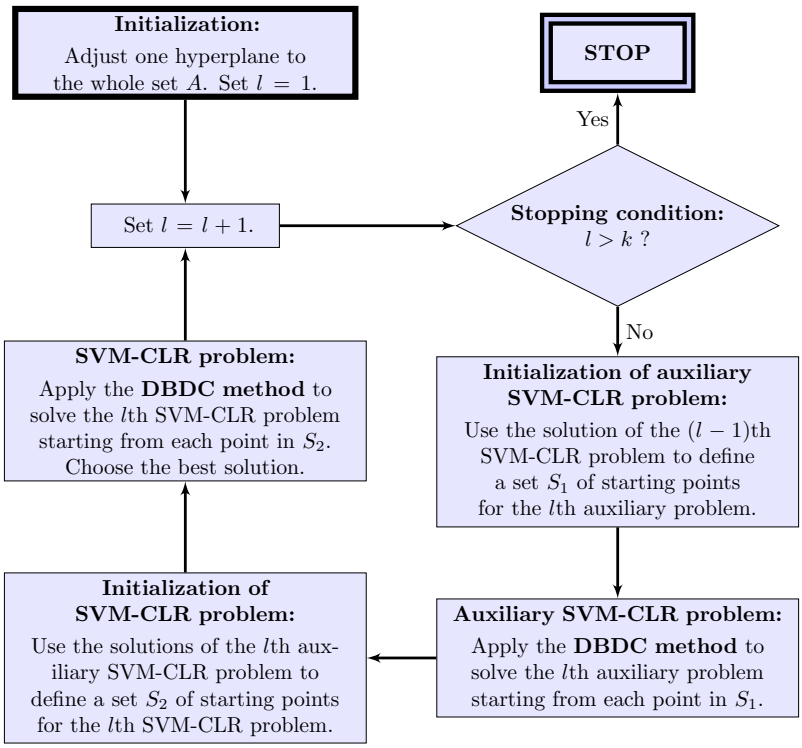


Figure 7.1: The flowchart of the double bundle method DB-SVM-CLR

7.3 Numerical results

To confirm the efficiency and reliability of the DB-SVM-CLR method, numerical experiments are performed in Publication V. The results are compared with a frequently used piecewise quadratic fit function

$$\hat{F}_k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m} \min_{j=1,\ldots,k} \left\{ \left((\boldsymbol{x}^j)^T a^i + y_j - b_i \right)^2 \right\} \tag{7.6}$$

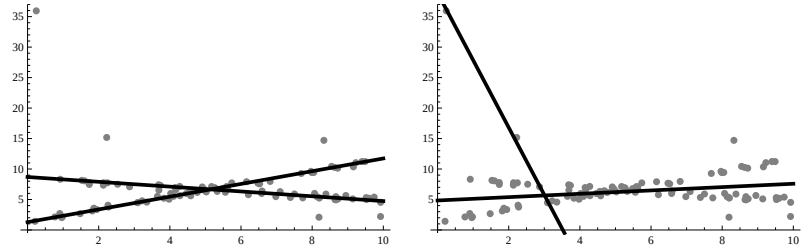
in order to validate the reliability of the new SVM-CLR formulation for the CLR problem and to give motivation for its usage. Note that the fit function (7.6) is often used to formulate the CLR problem [13, 14, 69] and we use the LMBM-CLR method [69] to minimize it. Therefore, by drawing the hyperplanes obtained with the fit function and the SVM-CLR formulation we can compare and illustrate the main differences in the solutions.

In the numerical results, we have concentrated on demonstrating the ability to find hyperplanes in two types of data structures. Altogether, we have used six different data sets and three of them are generated based on hyperplanes whereas three others are constructed using known clusters. To illustrate here the results in both data structures, we have selected data sets Two Lines and Clusters 1 from Publication V.

Figure 7.2 shows the performance in Two Lines data, which is generated by using two hyperplanes and by adding some noise and distinctive outliers. The results clearly show that the solution obtained by using the fit function (7.6) is considerably affected by outliers. Therefore, we are not able to detect the correct hyperplanes. However, the new SVM-CLR formulation (7.5) is a lot more reliable since outliers do not distort the solutions and the hyperplanes are placed correctly.

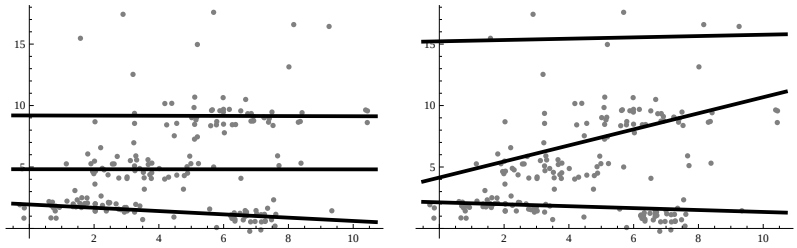
The same kind of observation can also be made in Clusters 1 data, which is generated based on four known clusters. The results of both the SVM-CLR formulation (7.5) and the fit function (7.6) are presented in Figure 7.3 showing that the performance of the fit function is again distorted by outliers. In this case, it means that one hyperplane out of three is placed to approximate the outliers. Thus, only two hyperplanes are able to approximate the clusters. The SVM-CLR formulation, in turn, is able to place all hyperplanes quite correctly and in this way captures more useful information about the data set.

All in all, the numerical results show that the new SVM-CLR formulation (7.5) outperforms the fit function (7.6) in nearly all test problems. Unlike the fit function, the SVM-CLR model is also able to distinguish hyperplanes correctly in the cases, where data points cover the input space quite evenly. In addition, a significant feature of the new model is the ability to tolerate outliers, which often appear in data sets. Therefore, we can conclude that the DB-SVM-CLR method is a robust and useful alternative to solve CLR problems.



a) SVM-CLR model for $\varepsilon = 0.5$ b) Fit function

Figure 7.2: Result for Two Lines data set with two adjusted hyperplanes



a) SVM-CLR model for $\varepsilon = 0.5$ b) Fit function

Figure 7.3: Result for Clusters 1 data set with three adjusted hyperplanes

Chapter 8

Conclusions

The existing bundle methods are considered to be efficient and reliable methods for nonsmooth optimization (NSO). The power of these methods typically rely on the bundling of subgradients and a convex cutting plane model of the objective function, which in the convex case gives a good estimate for the problem. However, in the nonconvex setting, a convex cutting plane model is not able to describe the concave behaviour of the objective, which can result in an inaccurate model unable to capture the most relevant parts of the nonconvex objective function. Therefore, in this dissertation, we have focused on developing new bundle methods for nonsmooth DC optimization, where the objective function is assumed to be a difference of two convex (DC) functions. These types of bundle methods have not been considered before, even though by explicitly utilizing the DC structure, we are able to take into account both the convex and concave behaviour of the objective function. In addition, DC functions cover a broad class of nonconvex functions frequently encountered in optimization.

In this dissertation, the DC structure has enabled us to form separate approximations for both the convex and concave parts of the objective function. Therefore, by combining these approximations we have obtained a new nonconvex DC cutting plane model, which is a more realistic and accurate approximation of the nonconvex function than the convex cutting plane model. In addition, this approach has made it possible to avoid the somewhat arbitrary downward shifting of the first order approximations typically used in nonconvex bundle methods to guarantee the interpolation property of the model.

Based on the new nonconvex DC cutting plane model we have presented new bundle methods for nonsmooth DC optimization. The first one is the proximal bundle method (PBDC) and it is designed to find critical points for unconstrained DC problems. Since criticality has some poor features, we have developed the double bundle method (DBDC) for unconstrained DC problems to improve the optimality condition by guaranteeing Clarke stationarity instead of criticality. To cover even a wider set of problems, we have designed the multiobjective double bundle method (MDBDC) capable of handling both multiobjective and constrained DC problems. The numerical results have shown that each of these new bundle methods is efficient and has a good performance. One major benefit of these methods is their ability to find global minimizers, even though they are only local solution methods. One reason for this seems to be the nonconvex DC cutting plane model, which is able to capture relevant information about the objective function.

Unfortunately, the DC decomposition of the objective function is not always available or possible to construct. Thus, we have also introduced the splitting metrics diagonal bundle method (SMDB) utilizing implicitly the DC structure of the objective function. This method has been designed to cover large-scale problems in unconstrained nonsmooth optimization, while the PBDC, DBDC and MDBDC are capable of handling small- or medium-scale problems only. The numerical results of the SMDB have confirmed the efficiency of the method and shown that the implicit utilization of the DC structure gives some advantages over the subgradient locality measures used in more traditional large-scale bundle methods.

To consider the usage of the DC structure in practical applications, we have proposed a bundle method based on the DBDC to solve clusterwise linear regression (CLR) problems. The main idea has been to use the support vector machines (SVM) approach to model the CLR problem after which we have constructed the DC decomposition of the new formulation. Finally, the new model has been solved with the new bundle method (DB-SVM-CLR) combining the DBDC and the incremental approach. The numerical results have proved that the DB-SVM-CLR is a robust and useful alternative to provide solutions for CLR problems.

In summary, we have introduced new bundle methods for nonsmooth optimization, which utilize either explicitly or implicitly the DC structure of the objective function. With these methods the aim has been to offer approaches, which take into account the real nonconvex structure of the objective in order to better handle nonconvexity in optimization. Even though these new methods have shown to be efficient, some work and open questions still remain. First, the methods developed will be used to solve some other practical applications encountered, for example, in data mining. Second, the strongest optimality condition used in our methods is Clarke stationarity. However, inf-stationarity is a stronger optimality condition for DC problems and, therefore, another goal could be to design a method guaranteeing inf-stationarity. In addition, the SMDB method only implicitly utilizes the DC structure. Thus, we will design a bundle method for large-scale problems using the DC decomposition explicitly, since this could give an even bigger benefit that the implicit usage of the DC structure.

Bibliography

[1] Ahmadi, A. A., and Hall, G. DC decomposition of nonconvex polynomials with algebraic techniques. *Mathematical Programming* (2017). DOI 10.1007/s10107-017-1144-5

[2] Apkarian, P., Noll, D., and Prot, O. A trust region spectral bundle method for nonconvex eigenvalue optimization. *SIAM Journal on Optimization 19*, 1 (2008), 281–306.

[3] Astorino, A., Frangioni, A., Fuduli, A., and Gorgone, E. A nonmonotone proximal bundle method with (potentially) continuous step decisions. *SIAM Journal on Optimization 23*, 3 (2013), 1784–1809.

[4] Astorino, A., Frangioni, A., Gaudioso, M., and Gorgone, E. Piecewise-quadratic approximations in convex numerical optimization. *SIAM Journal on Optimization 21*, 4 (2011), 1418–1438.

[5] Astorino, A., Fuduli, A., and Gaudioso, M. DC models for spherical separation. *Journal of Global Optimization 48*, 4 (2010), 657–669.

[6] Astorino, A., Fuduli, A., and Gaudioso, M. Margin maximization in spherical separation. *Computational Optimization and Applications 53*, 2 (2012), 301–322.

[7] Bagirov, A. M. A method for minimization of quasidifferentiable functions. *Optimization Methods and Software 17*, 1 (2002), 31–60.

[8] Bagirov, A. M., and Ganjehlou, A. N. A quasisecant method for minimizing nonsmooth functions. *Optimization Methods and Software 25*, 1 (2010), 3–18.

[9] Bagirov, A. M., Karasözen, B., and Sezer, M. Discrete gradient method: Derivative-free method for nonsmooth optimization. *Journal of Optimization Theory and Applications 137*, 2 (2008), 317–334.

[10] Bagirov, A. M., Karmitsa, N., and Mäkelä, M. M. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*. Springer, Cham, Heidelberg, 2014.

[11] Bagirov, A. M., Mahmood, A., and Barton, A. Prediction of monthly rainfall in Victoria, Australia: Clusterwise linear regression approach. *Atmospheric Research 188* (2017), 20–29.

[12] Bagirov, A. M., and Ugon, J. Codifferential method for minimizing nonsmooth DC functions. *Journal of Global Optimization 50*, 1 (2011), 3–22.

[13] Bagirov, A. M., Ugon, J., and Mirzayeva, H. Nonsmooth nonconvex optimization approach to clusterwise linear regression problems. *European Journal of Operational Research 229*, 1 (2013), 132–142.

[14] Bagirov, A. M., Ugon, J., and Mirzayeva, H. Nonsmooth optimization algorithm for solving clusterwise linear regression problems. *Journal of Optimization Theory and Applications 164*, 3 (2015), 755–780.

[15] Bagirov, A. M., and Yearwood, J. A new nonsmooth optimization algorithm for minimizer sum-of-squares clustering problems. *European Journal of Operational Research 170*, 2 (2006), 578–596.

[16] Bagley, S. C., White, H., and Golomb, B. A. Logistic regression in the medical literature: Standards for use and reporting, with particular attention to one medical domain. *Journal of Clinical Epidemiology 54*, 10 (2001), 979–985.

[17] Ben-Dor, A., Shamir, R., and Yakhini, Z. Clustering gene expression patterns. *Journal of Computational Biology 6*, 3–4 (2004), 281–297.

[18] Bihain, A. Optimization of upper semidifferentiable functions. *Journal of Optimization Theory and Applications 44*, 4 (1984), 545–568.

[19] Bomze, I. M., and Locatelli, M. Undominated d.c. decompositions of quadratic functions and applications to branch-and-bound approaches. *Computational Optimization and Applications 28*, 2 (2004), 227–245.

[20] Bradley, P. S., Fayyad, U. M., and Mangasarian, O. L. Mathematical programming for data mining: Formulations and challenges. *INFORMS Journal on Computing 11*, 3 (1999), 217–238.

[21] Bränndlund, U., Kiwiel, K. C., and Lindberg, P. O. A descent proximal level bundle method for convex nondifferentiable optimization. *Operations Research Letters 17*, 3 (1995), 121–126.

[22] Burke, J. V., Lewis, A. S., and Overton, M. L. Approximating subdifferentials by random sampling of gradients. *Mathematics of Operations Research 27*, 3 (2002), 567–584.

[23] Burke, J. V., Lewis, A. S., and Overton, M. L. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization 15*, 3 (2005), 751–779.

[24] Cagnina, L., Errecalde, M., Ingaramo, D., and Rosso, P. An efficient particle swarm optimization approach to cluster short texts. *Information Sciences 265* (2014), 36–49.

[25] Carbonneau, R. A., Caporossi, G., and Hansen, P. Globally optimal clusterwise regression by mixed logical-quadratic programming. *European Journal of Operational Research 212*, 1 (2011), 213–222.

[26] Carrizosa, E., Guerrero, V., and Romero Morales, D. Visualizing data as objects by DC (difference of convex) optimization. *Mathematical Programming* (2017). DOI 10.1007/s10107-017-1156-1

[27] Chatterjee, S., and Hadi, A. S. *Regression Analysis by Example*, 5th ed. John Wiley & Sons, New Jersey, 2012.

[28] Chen, X. Smoothing methods for nonsmooth, nonconvex minimization. *Mathematical Programming 134*, 1 (2012), 71–99.

[29] Clarke, F. H. *Optimization and Nonsmooth Analysis*. Wiley-Interscience, New York, 1983.

[30] Clarke, F. H., Ledyaev, Y. S., Stern, R. J., and Wolenski, P. R. *Nonsmooth Analysis and Control Theory*. Springer, New York, 1998.

[31] Collobert, R., and Bengio, S. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research 1* (2001), 143–160.

[32] Cook, D. O., Kieschnick, R., and McCullough, B. Regression analysis of proportions in finance with self selection. *Journal of Empirical Finance 15*, 5 (2008), 860–867.

[33] Demyanov, V. F., and Rubinov, A. M. *Constructive Nonsmooth Analysis*. Peter Lang, Frankfurt am Main, 1995.

[34] Fendl, H. *A feasible second order bundle algorithm for nonsmooth, nonconvex optimization problems with inequality constraints and its application to certificates of infeasibility*. PhD thesis, University of Vienna, 2011.

[35] Fendl, H., and Schichl, H. A feasible second order bundle algorithm for nonsmooth, nonconvex optimization problems with inequality constraints: I. Derivation and convergence. *arXiv preprint: arXiv:1506.07937* (2015).

[36] Fendl, H., and Schichl, H. A feasible second order bundle algorithm for nonsmooth nonconvex optimization problems with inequality constraints: II. Implementation and numerical results. *arXiv preprint: arXiv:1506.08021* (2015).

[37] Ferrer, A. Representation of a polynomial function as a difference of convex polynomials, with an application. In *Generalized Convexity and Generalized Monotonicity* (Eds. Hadjisavvas, N., Martínez-Legaz, J. E., Penot, J.-P.), Lecture Notes in Economics and Mathematical Systems 502, Springer, Berlin (2001), 189–207.

[38] Ferrer, A., and Martínez-Legaz, J. E. Improving the efficiency of DC global optimization methods by improving the DC representation of the objective function. *Journal of Global Optimization 43*, 4 (2009), 513–531.

[39] Floudas, C. A., and Pardalos, P. M. (Eds.) *Optimization in Computational Chemistry and Molecular Biology: Local and Global Approaches*, vol. 40. Springer, New York, 2013.

[40] Frangioni, A. Generalized bundle methods. *SIAM Journal on Optimization 13*, 1 (2002), 117–156.

[41] Fuduli, A., Gaudioso, M., and Giallombardo, G. A DC piecewise affine model and a bundling technique in nonconvex nonsmooth minimization. *Optimization Methods and Software 19*, 1 (2004), 89–102.

[42] Fuduli, A., Gaudioso, M., and Giallombardo, G. Minimizing nonconvex nonsmooth functions via cutting planes and proximity control. *SIAM Journal on Optimization 14*, 3 (2004), 743–756.

[43] Fuduli, A., Gaudioso, M., and Nurminski, E. A. A splitting bundle approach for non-smooth non-convex minimization. *Optimization 64*, 5 (2015), 1131–1151.

[44] Ganjigatti, J. P., Pratihar, D. K., and Choudhury, A. R. Global versus cluster-wise regression analyses for prediction of bead geometry in MIG welding process. *Journal of Materials Processing Technology 189*, 1–3 (2007), 352–366.

[45] Gaudioso, M., Giallombardo, G., Miglionico, G., and Bagirov, A. M. Minimizing nonsmooth DC functions via successive DC piecewise-affine approximations. *Journal of Global Optimization* (2017). DOI 10.1007/s10898-017-0568-z

[46] Gaudioso, M., and Gorgone, E. Gradient set splitting in nonconvex nonsmooth numerical optimization. *Optimization Methods and Software 25*, 1 (2010), 59–74.

[47] Gígola, C., and Gomez, S. A regularization method for solving the finite convex min-max problem. *SIAM Journal on Numerical Analysis 27*, 6 (1990), 1621–1634.

[48] Gulpinar, N., Le Thi, H. A., and Moeini, M. Robust investment strategies with discrete asset choice constraints using DC programming. *Optimization 59*, 1 (2010), 45–62.

[49] Györfi, L., Kohler, M., Krzyżak, A., and Walk, H. *A Distribution-Free Theory of Nonparametric Regression.* Springer, New York, 2002.

[50] Haarala, M., Miettinen, K., and Mäkelä, M. M. New limited memory bundle method for large-scale nonsmooth optimization. *Optimization Methods and Software 19*, 6 (2004), 673–692.

[51] Haarala, N., Miettinen, K., and Mäkelä, M. M. Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Mathematical Programming 109*, 1 (2007), 181–205.

[52] Hare, W., Sagastizábal, C., and Solodov, M. A proximal bundle method for nonsmooth nonconvex functions with inexact information. *Computational Optimization and Applications 63*, 1 (2016), 1–28.

[53] Hare, W., and Sagastizábal, C. A redistributed proximal bundle method for nonconvex optimization. *SIAM Journal on Optimization 20*, 5 (2010), 2442–2473.

[54] Hartman, P. On functions representable as a difference of convex functions. *Pacific Journal of Mathematics 9*, 3 (1959), 707–713.

[55] Herskovits, J., and Goulart, E. Sparse quasi-Newton matrices for large scale nonlinear optimization. In *Proceedings of the 6th Word Congress on Structural and Multidisciplinary Optimization*, Rio de Janeiro, Brazil (2005).

[56] Hintermüller, M. A proximal bundle method based on approximate subgradients. *Computational Optimization and Applications 20*, 3 (2001), 245–266.

[57] Hiriart-Urruty, J.-B. Generalized differentiability, duality and optimization for problems dealing with differences of convex functions. In *Convexity and Duality in Optimization* (Eds. Ponstein, J.), Lecture Notes in Economics and Mathematical Systems 256, Springer, Berlin (1985), 37–70.

[58] Hiriart-Urruty, J.-B. From convex optimization to nonconvex optimization. Necessary and sufficient conditions for global optimality. In *Nonsmooth Optimization and Related Topics* (Eds. Clarke, F. H., Dem'yanov, V. F., Giannessi, F.), Ettore Majorana International Sciences Series 43, Springer, Boston (1989), pp. 219–239.

[59] Hiriart-Urruty, J.-B., and Lemaréchal, C. *Convex Analysis and Minimization Algorithms II.* Springer, Berlin, 1993.

[60] Holmberg, K., and Tuy, H. A production-transportation problem with stochastic demand and concave production costs. *Mathematical Programming 85*, 1 (1999), 157–179.

[61] Horst, R., and Thoai, N. V. DC programming: Overview. *Journal of Optimization Theory and Applications 103*, 1 (1999), 1–43.

[62] Horst, R., and Tuy, H. *Global Optimization: Deterministic Approaches*, 1st ed. Springer, Berlin, 1990.

[63] Hou, J., Liu, W., E, X., and Cui, H. Towards parameter-independent data clustering and image segmentation. *Pattern Recognition 60* (2016), 25–36.

[64] Hou, L., and Sun, W. On the global convergence of a nonmonotone proximal bundle method for convex nonsmooth minimization. *Optimization Methods and Software 23*, 2 (2008), 227–235.

[65] Joki, K., Bagirov, A. M., Karmitsa, N., Mäkelä, M. M., and Taheri, S. Double bundle method for finding Clarke stationary points in nonsmooth DC programming. *SIAM Journal on Optimization*, to appear (2018).

[66] Karas, E., Ribeiro, A., Sagastizábal, C., and Solodov, M. A bundle-filter method for nonsmooth convex constrained optimization. *Mathematical Programming 116*, 1–2 (2009), 297–320.

[67] Karmitsa, N. Diagonal bundle method for nonsmooth sparse optimization. *Journal of Optimization Theory and Applications 166*, 3 (2015), 889–905.

[68] Karmitsa, N., and Bagirov, A. M. Limited memory discrete gradient bundle method for nonsmooth derivative-free optimization. *Optimization 61*, 12 (2012), 1491–1509.

[69] Karmitsa, N., Bagirov, A. M., and Taheri, S. Limited memory bundle method for solving large clusterwise linear regression problems. *TUCS Technical Report No. 1172*, Turku Centre for Computer Science, Turku (2016).

[70] Kelley, J. E. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics 8*, 4 (1960), 703–712.

[71] Kiwiel, K. C. A descent method for nonsmooth convex multiobjective minimization. *Large Scale Systems 8*, 2 (1985), 119–129.

[72] Kiwiel, K. C. *Methods of Descent for Nondifferentiable Optimization.* Lecture Notes in Mathematics 1133. Springer, Berlin, 1985.

[73] Kiwiel, K. C. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming 46*, 1–3 (1990), 105–122.

[74] Kiwiel, K. C. Restricted step and Levenberg-Marquardt techniques in proximal bundle methods for nonconvex nondifferentiable optimization. *SIAM Journal on Optimization 6*, 1 (1996), 227–249.

[75] Kiwiel, K. C. A bundle Bergman proximal method for convex nondifferentiable minimization. *Mathematical Programming 85*, 2 (1999), 241–258.

[76] Kiwiel, K. C. Efficiency of proximal bundle methods. *Journal of Optimization Theory and Applications 104*, 3 (2000), 589–603.

[77] Kiwiel, K. C. Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM Journal on Optimization 18*, 2 (2007), 379–388.

[78] Kiwiel, K. C. A nonderivative version of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM Journal on Optimization 20*, 4 (2010), 1983–1994.

[79] Kogan, J., Nicholas, C., and Teboulle, M. *Grouping Multidimensional Data: Recent Advances in Clustering.* Springer, Berlin, 2006.

[80] Kuo, R. J., Ho, L. M., and Hu, C. M. Cluster analysis in industrial market segmentation through artificial neural network. *Computers & Industrial Engineering 42*, 2–4 (2002), 391–399.

[81] Kärkkäinen, T., and Heikkola, E. Robust formulations for training multilayer perceptrons. *Neural Computation 16*, 4 (2004), 837–862.

[82] Le Thi, H. A., and Pham Dinh, T. Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms. *Journal of Global Optimization 11*, 3 (1997), 253–285.

[83] Le Thi, H. A., and Pham Dinh, T. The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of Operations Research 133*, 1–4 (2005), 23–46.

[84] Le Thi, H. A., and Pham Dinh, T. Difference of convex functions algorithms (DCA) for image restoration via a Markov random field model. *Optimization and Engineering 18*, 4 (2017), 873–906.

[85] Lemaréchal, C. An extension of Davidon methods to nondifferentiable problems. In *Nondifferentiable Optimization* (Eds. Balinski, M. L. and Wolfe, P.), Mathematical Programming Study 3, Springer, Berlin (1975), pp. 95–109.

[86] Lemaréchal, C. Combining Kelley's and conjugate gradient methods. In *Abstracts of IX International Symposium on Mathematical Programming*, Budapest, Hungary (1976).

[87] Lewis, A. S., and Overton, M. L. Nonsmooth optimization via quasi-Newton methods. *Mathematical Programming 141*, 1–2 (2013), 135–163.

[88] Lipp, T., and Boyd, S. Variations and extension of the convex–concave procedure. *Optimization and Engineering 17*, 2 (2016), 263–287.

[89] Long, Q., Wu, C., Wang, X. and Wu, Z. A modified quasisecant method for global optimization. *Applied Mathematical Modelling 51*, (2017), 21–37.

[90] Lukšan, L. Dual method for solving a special problem of quadratic programming as a subproblem at linearly constrained nonlinear minimax approximation. *Kybernetika 20*, 6 (1984), 445–457.

[91] Lukšan, L., and Vlček, J. A bundle-Newton method for nonsmooth unconstrained minimization. *Mathematical Programming 83*, 1–3 (1998), 373–391.

[92] Lukšan, L., and Vlček, J. Globally convergent variable metric method for convex nonsmooth unconstrained minimization. *Journal of Optimization Theory and Applications 102*, 3 (1999), 593–613.

[93] Mäkelä, M. M. Survey of bundle methods for nonsmooth optimization. *Optimization Methods and Software 17*, 1 (2002), 1–29.

[94] Mäkelä, M. M., Eronen, V.-P., and Karmitsa, N. On nonsmooth multiobjective optimality conditions with generalized convexities. In *Optimization in Science and Engineering* (Eds. Rassias, T. M., Floudas, C. A. and Butenko, S.), Springer, New York (2014), pp. 333–357.

[95] Mäkelä, M. M., Karmitsa, N., and Wilppu, O. Proximal bundle method for nonsmooth and nonconvex multiobjective optimization. In *Mathematical Modeling and Optimization of Complex Structures* (Eds. Neittaanmäki, P., Repin, S. and Tuovinen, T.), Computational Methods in Applied Sciences 40, Springer, Cham (2016), pp. 191–204.

[96] Mäkelä, M. M., and Neittaanmäki, P. *Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control*. World Scientific Publishing Co., Singapore, 1992.

[97] Miettinen, K., and Mäkelä, M. M. Interactive bundle-based method for nondifferentiable multiobjeective optimization: NIMBUS. *Optimization 34*, 3 (1995), 231–246.

[98] Mifflin, R., and Sagastizábal, C. A $\mathcal{VU}$-algorithm for convex minimization. *Mathematical Programming 104*, 2–3 (2005), 583–608.

[99] Mordukhovich, B. *Variational Analysis and Generalized Differentiation I and II*, 1st ed. Springer, Berlin, 2006.

[100] Moreau, J. J., Panagiotopoulos, P. D., and Strang, G. (Eds.) *Topics in Nonsmooth Mechanics*. Birkhäuesr Verlag, Basel, 1988.

[101] Nelder, J. A. and Mead, R. A simplex method for function minimization. *The Computer Journal 7*, 4 (1965), 308–313.

[102] Noll, D. Cutting plane oracles to minimize non-smooth non-convex functions. *Set-Valued and Variational Analysis 18*, 3–4 (2010), 531–568.

[103] de Oliveira, W. Proximal bundle methods for nonsmooth DC programming. *Manuscript* (2017). URL https://drive.google.com/file/d/0ByLZhUZ45Y-HQnVvOEZ3REw0Sk0/view (2.2.2018)

[104] de Oliveira, W., and Sagastizábal, C. Level bundle methods for oracles with on-demand accuracy. *Optimization Methods and Software 29*, 6 (2014), 1180–1209.

[105] Olsson, D. M., and Nelson, L. S. The Nelder-Mead simplex procedure for function minimization. *Technometrics 17*, 1 (1975), 45–51.

[106] Ovcharova, N., and Gwinner, J. A study of regularization techniques of nondifferentiable optimization in view of application to hemivariational inequalities. *Journal of Optimization Theory and Applications 162*, 3 (2014), 754–778.

[107] Pham Dinh, T., and Le Thi, H. A. Convex analysis approach to D.C. programming: Theory, algorithms and applications. *Acta Mathematica Vietnamica 22*, 1 (1997), 289–355.

[108] Pham Dinh, T., and Le Thi, H. A. A DC optimization algorithm for solving the trust-region subproblem. *SIAM Journal on Optimization 8*, 2 (1998), 476–505.

[109] Pham Dinh, T., and Le Thi, H. A. Recent advances in DC programming and DCA. In *Transactions on Computational Intelligence XIII* (Eds. Nguyen, N.-T. and Le Thi, H. A.), Lecture Notes in Computer Science 8342, Springer, Berlin (2014), pp. 1–37.

[110] Poggi, J.-M., and Portier, B. PM10 forecasting using clusterwise regression. *Atmospheric Environment 45*, 38 (2011), 7005–7014.

[111] Powell, M. J. D. An efficient method for finding the minimizer of a function of several variables without calculating derivatives. *The Computer Journal 7*, 2 (1964), 155–162.

[112] Preda, C., and Saporta, G. Clusterwise PLS regression on a stochastic process. *Computational Statistics & Data Analysis 49*, 1 (2005), 99–108.

[113] Rockafellar, R. T. *Convex Analysis*. Princeton University Press, Princeton, New Jersey, 1970.

[114] 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 (1992), 121–152.

[115] Shor, N. Z. *Minimization Methods for Non-Differentiable Functions*. Springer, Berlin, 1985.

[116] Smola, A. J., and Schölkopf, B. A tutorial on support vector regression. *Statistics and Computing 14* (2004), 199–222.

[117] Souza, J. C. O., Oliveira, P. R., and Soubeyran, A. Global convergence of a proximal linearized algorithm for difference of convex functions. *Optimization Letters 10*, 7 (2016), 1529–1539.

[118] Stella, L., Themelis, A., and Patrinos, P. Forward–backward quasi-Newton methods for nonsmooth optimization problems. *Computational Optimization and Applications 67*, 3 (2017), 443–487.

[119] Toland, J. F. On subdifferential calculus and duality in nonconvex optimization. *Mémoires de la Société Mathématique de France 60* (1979), 177–183.

[120] Tuy, H. *Convex Analysis and Global Optimization*, 1st ed. Kluwer Academic Publishers, Dordrecht, 1998.

[121] Veloso, R., Portela, F., Santos, M. F., Silva, Á., Rua, F., Abelha, A., and Machado, J. A clustering approach for predicting readmissions in intensive medicine. *Procedia Technology 16* (2014), 1307–1316.

[122] Vlček, J., and Lukšan, L. Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization. *Journal of Optimization Theory and Applications 111*, 2 (2001), 407–430.

[123] Wang, S. Algorithms for multiobjective and nonsmooth optimization. In *Methods of Operations Research* (Eds. Kleinschmidt, P., Radermacher, F. J., Sweitzer, W. and Wildermann, H.), Athenäum Verlag, Frankfurt am Main (1989), pp. 131–142.

[124] Wedel, M., and Kistemaker, C. Consumer benefit segmentation using clusterwise linear regression. *International Journal of Research in Marketing 6*, 1 (1989), 45–59.

[125] Wolfe, P. A method of conjugate subgradients for minimizing nondifferentiable functions. In *Nondifferentiable Optimization* (Eds. Balinski, M. L. and Wolfe, P.), Mathematical Programming Studies 3, Springer, Berlin (1975), pp. 145–173.

[126] Zangwill, W. I. Minimizing a function without calculating derivatives. *The Computer Journal 10*, 3 (1967), 293–296.

[127] Zangwill, W. I. Non-linear programming via penalty functions. *Management Science 13*, 5 (1967), 344–358.

Annales Universitatis Turkuensis

Turun yliopisto
University of Turku

Painosalama Oy, Turku, Finland 2018

ISBN 978-951-29-7276-0 (PRINT)
ISBN 978-951-29-7277-7 (PDF)
ISSN 0082-7002 (PRINT) | ISSN 2343-3175 (PDF)