TECHNISCHE UNIVERSITÄT
ILMENAU
Faculty of Mathematics and Natural Sciences
Mathematical Methods of Operations Research

Nonconvex and Mixed Integer Multiobjective Optimization with an Application to Decision Uncertainty

by

Julia Niebling

Doctoral Thesis
in partial fulfillment of the requirements
for the degree Dr. rer. nat.

Supervisor:	Prof. Dr. Gabriele Eichfelder
Second Reviewer:	Prof. Dr. Kathrin Klamroth
Third Reviewer:	Prof. Dr. Oliver Stein
Date of Submission:	October 15, 2019
Date of Defense:	December 18, 2019

urn:nbn:de:gbv:ilm1-2019000503

Acknowledgments

> ”We must have perseverance and above all confidence in ourselves. We must believe that we are gifted for something and that this thing must be attained.” Marie Curie (1867-1934)

Doing a doctorate is associated with facing a lot of challenges and motivational gaps, having ups and downs, getting in touch with success and drawbacks. However, I am grateful that I had people in my life who accompanied me on this way.

First, I want to express my gratitude to Prof. Dr. Gabriele Eichfelder for supervising me, guiding me and keeping me on track. Thanks for valuable discussions as well as some cheer-ups by certain, popular comic strips.

I appreciate the many possibilities to participate in conferences and workshops to present my research and meet other scientists. At this point, I would like to thank Dr. Thomas Boeck, Prof. Dr. Marianna De Santis, Prof. Dr. Kathrin Klamroth, Prof. Dr. Andreas Löhne, M. Sc. Stefan Rocktäschel, Prof. Dr. Oliver Stein and Dr. Benjamin Weißing for their scientific support or doing research together.

Moreover, I want to thank the Thuringian State Graduate Support Regulation, the Carl Zeiss foundation, and the DFG-funded Research Training Group 1567 “Lorentz Force Velocimetry and Lorentz Force Eddy Current Testing” for financial support.

Last, but not least, my personal thanks and a lot of love go to my family, my friends and my boyfriend who were always by my side listening to my complaints, encouraging me and motivating me. Thanks, mom and dad, that you recognized my talents early and that you supported me my whole life. Danke!

Abstract

Multiobjective optimization problems commonly arise in different fields like economics or engineering. In general, when dealing with several conflicting objective functions, there is an infinite number of optimal solutions which cannot usually be determined analytically.

This thesis presents new branch-and-bound-based approaches for computing the globally optimal solutions of multiobjective optimization problems of various types. New algorithms are proposed for smooth multiobjective nonconvex optimization problems with convex constraints as well as for multiobjective mixed integer convex optimization problems. Both algorithms guarantee a certain accuracy of the computed solutions, and belong to the first deterministic algorithms within their class of optimization problems. Additionally, a new approach to compute a covering of the optimal solution set of multiobjective optimization problems with decision uncertainty is presented. The three new algorithms are tested numerically. The results are evaluated in this thesis as well.

The branch-and-bound based algorithms deal with box partitions and use selection rules, discarding tests and termination criteria. The discarding tests are the most important aspect, as they give criteria whether a box can be discarded as it does not contain any optimal solution. We present discarding tests which combine techniques from global single objective optimization with outer approximation techniques from multiobjective convex optimization and with the concept of local upper bounds from multiobjective combinatorial optimization. The new discarding tests aim to find appropriate lower bounds of subsets of the image set in order to compare them with known upper bounds numerically.

Kurzfassung

Multikriterielle Optimierungprobleme sind in diversen Anwendungsgebieten wie beispielsweise in den Wirtschafts- oder Ingenieurwissenschaften zu finden. Da hierbei mehrere konkurrierende Zielfunktionen auftreten, ist die Lösungsmenge eines derartigen Optimierungsproblems im Allgemeinen unendlich groß und kann meist nicht in analytischer Form berechnet werden.

In dieser Dissertation werden neue Branch-and-Bound basierte Algorithmen zur Lösung verschiedener Klassen von multikriteriellen Optimierungsproblemen entwickelt und vorgestellt. Der Branch-and-Bound Ansatz ist eine typische Methode der globalen Optimierung. Einer der neuen Algorithmen löst glatte multikriterielle nichtkonvexe Optimierungsprobleme mit konvexen Nebenbedingungen, während ein zweiter zur Lösung multikriterieller gemischt-ganzzahliger konvexer Optimierungsprobleme dient. Beide Algorithmen garantieren eine gewisse Genauigkeit der berechneten Lösungen und gehören damit zu den ersten deterministischen Algorithmen ihrer Art. Zusätzlich wird ein Algorithmus zur Berechnung einer Überdeckung der Lösungsmenge multikriterieller Optimierungsprobleme mit Entscheidungsunsicherheit vorgestellt. Alle drei Algorithmen wurden numerisch getestet. Die Ergebnisse werden ebenfalls in dieser Arbeit ausgewertet.

Die neuen Algorithmen arbeiten alle mit Boxunterteilungen und nutzen Auswahlregeln, sowie Verwerfungs- und Terminierungskriterien. Dabei spielen gute Verwerfungskriterien eine zentrale Rolle. Diese entscheiden, ob eine Box verworfen werden kann, da diese sicher keine Optimallösung enthält. Die neuen Verwerfungskriterien nutzen Methoden aus der globalen skalarwertigen Optimierung, Approximationstechniken aus der multikriteriellen konvexen Optimierung sowie ein Konzept aus der kombinatorischen Optimierung. Dabei werden stets untere Schranken der Bildmengen konstruiert, die mit bisher berechneten oberen Schranken numerisch verglichen werden können.

Table of Contents

1 Introduction

A typical optimization problem consists of two things: an objective function and a feasible set. When minimizing a classical single objective optimization problem, one is interested in finding the smallest value of the objective function with respect to the given feasible set. All other values of the objective function on the feasible set are greater or equal to the desired "minimal" value. When using a total order relation, the existence of this minimal value is equivalent to its uniqueness. All feasible points that get mapped onto the minimal value are called *optimal solutions* of the single objective optimization problem. These terms are often used in a global sense, but also locally optimal solutions may be of interest. As opposed to a globally optimal solution, a locally optimal solution is only required to obtain a minimal value in one of its neighborhoods. Most algorithms for solving optimization problems only aim for finding locally optimal solutions. This is reasonable since locally optimal solutions are also globally optimal if we demand some additional properties. The usual way is to require convexity of the objective function and the feasible set.

This thesis examines multiobjective optimization problems. Hence, we are interested in the simultaneous minimization of more than one objective function. In general, we cannot assume the existence of one point minimizing all objective functions at once. As a consequence, we need to establish a different optimality concept than in single objective optimization. Furthermore, we want to ensure global optimality of the found solutions.

1.1 Organization of this Thesis

This thesis proposes algorithms for different kinds of multiobjective optimization problems in order to find globally optimal solutions. In the next section of the current

chapter, we review the existing literature concerning each kind of multiobjective optimization problem considered in this work. Therein, each subsection contains a survey on existing algorithms and issues about the respective optimization problem. Section 1.2.1 deals with general multiobjective optimization, while Section 1.2.2 reviews existing literature on multiobjective mixed integer optimization. The last subsection in this chapter, Section 1.2.3, offers an overview of works dealing with uncertainty in multiobjective optimization. After the literature review, we continue by stating the main developments and insights for multiobjective optimization contributed by this thesis.

The next chapter, Chapter 2, is dedicated to the introduction into multiobjective optimization clarifying the most important notions and definitions. Many tools in optimization are already established. The ones which play an essential role in this work are mentioned and explained in Chapter 3.

The topics of the next chapters are the newly developed algorithms for the different types of multiobjective optimization problems. In Chapter 4, we start by considering multiobjective optimization problems with smooth objective functions and a feasible set described by continuous variables and convex constraints. The algorithm `MOPBB` proposed in this chapter is one of the main contributions of this thesis and, thus, a convergence analysis and numerical experiments are given. The numerical results are stated in Chapter 5. It consists of illustrative examples and an application in the field of Lorentz Force velocimetry. Moreover, `MOPBB` is compared numerically with another algorithm in this chapter.

The newly developed techniques can also be applied for other optimization problems. The algorithm `MOMIX`, described in Chapter 6, solves multiobjective mixed integer convex optimization problems, i. e., optimization problems with continuous and integer decision variables and convex objective functions. For this procedure, theoretical results concerning convergence and numerical results are presented.

Another new algorithm was developed for multiobjective optimization problems with decision uncertainty and is presented in Chapter 7. *Decision uncertainty* means that the variables are associated with uncertainties. For example, a certain temperature which solves an optimization problem cannot be realized exactly within high precision. We model such problems as set optimization problems. Thus, new methods

in order to compute lower and upper bounds of sets are proposed and used for the new algorithm MOPDUBB to obtain a covering of the optimal solution set. The chapter concludes with several examples of numerical experiments.

All terms and symbols which are standard mathematical vocabulary are listed after Appendix A. All special terms or notions which play an important role in this thesis are mentioned and defined in the relevant parts.

1.2 Literature Review

This section gives an overview of existing algorithms concerning multiobjective optimization and related topics. The first part represents a review of academic research on general multiobjective optimization. Afterwards, we consider more specific optimization, i. e., multiobjective mixed integer convex optimization problems as well as multiobjective problems with uncertainties, and set optimization. As we have not defined the notions for the solutions of a multiobjective optimization problem yet, these are denoted by *optimal solutions* or *images of optimal solutions* in this section. The proper definitions are stated in Section 2.3.

1.2.1 General Multiobjective Optimization with Continuous Variables

Several algorithms already exist for solving multiobjective (convex) optimization problems, see [Ehr05; Jah11] for an introduction and overview. Most of them are based on *scalarization approaches* [Eic08; FV16; MGS09]. In these approaches a new single objective problem depending on some parameters is derived and solved by known methods for single objective optimization problems. For instance, the *weighted sum scalarization* is a sum of the objective functions each of them multiplied by a positive weight. With a set of parameter values (e.g., weights) and more iterations, an approximation of the optimal solution set can be obtained. If we only use local methods to find optimal solutions of the nonconvex single objective problems, we will just obtain locally optimal solutions of the original vector-valued problem. Using a global

solver for each choice of parameters for the scalarization problems is a very time-consuming and inefficient approach. Moreover, most scalarizations turn some of the nonconvex objective functions into constraints, but it is well-known that nonconvex constraints are especially difficult to handle [KSS15]. The weighted sum scalarization avoids this, even though it is not an appropriate scalarization for nonconvex problems. Therefore, the development of global solvers for multiobjective optimization problems without using scalarizations is important. For an introduction to global optimization, see [HT10].

Many methods for global optimization use stochastic strategies. Evolutionary algorithms are commonly used, see, for example, [Deb99; Deb01; Deb+02a; FF95; VFM96]. These algorithms try to find globally optimal solutions by mutation and crossover of individuals of a constructed population, i. e., some feasible points of the optimization problem. In fact, these procedures are able to find a global minimum in an infinite amount of time, but they do not guarantee finding a good solution in a finite time. That is one reason why it is of special interest to propose deterministic algorithms.

A deterministic approach to find globally optimal solutions for multiobjective optimization problems was introduced by Jahn [Jah06]. In this approach, a search region in the preimage space is discretized and then refined into "promising" areas. Though, higher dimensions of the preimage space require a larger number of function evaluations. This is due to the fact that Jahn's method does not use any information about derivatives. Another derivative-free algorithm for multiobjective optimization problems was proposed by Custódio and Madeira in [CM18] which is based on a direct search approach with a multistart strategy and is also able to find globally optimal solutions.

Some other algorithms for multiobjective optimization are based on inner or outer approximation techniques. A popular algorithm for multiobjective linear optimization is Benson's algorithm [Ben98a] and its generalization to multiobjective convex optimization [LRU14]. We consider this algorithm in Section 3.3 further. An approximation algorithm for multiobjective convex or nonconvex optimization can be found in [SKW02]. It combines cones and norms to get a piecewise linear approximation of the set of optimal solutions in the image space.

Other global optimization algorithms are based on branch-and-bound (B&B) methods, see [Adj+98; Aud+00; Cam+18; Dür01; HP09; MF94; TH88; Wan+08]. In [Sch12], general frameworks for geometric B&B algorithms for single and multiobjective optimization are introduced. Additionally, certain bounding operations and convergence theory are described. For single objective optimization problems, two of the most well-known algorithms, which use box partitions, are the DIRECT algorithm [JPS93] and the αBB method [MF94]. The DIRECT algorithm focuses on selecting boxes to have a good balance between a local and a global search strategy. However, it cannot guarantee finding good solutions in finite time [LS13b]. In contrast to this, the αBB algorithm uses lower bounds of the global minimum which are obtained by minimizing a convex underestimator of the objective function. These bounds are then improved until a given accuracy is reached. Other methods for finding lower bounds for global minima use maximal values of the dual problem [Dür01], or use the Lipschitz constant [LS13a; ŽŽ16]. The DIRECT algorithm was also extended to multiobjective optimization problems, see [Cam+18; Wan+08]. However, in most cases the multiobjective version of the DIRECT algorithm shows bad convergence results and has to be accelerated by another global or local optimization method.

The first B&B based algorithm for more than one objective function and with some basic convergence results was introduced by Fernández and Tóth in [FT09]. The described procedure solves biobjective optimization problems, i.e., optimization problem with two objective functions, and is based on interval arithmetic. Another interval B&B algorithm for biobjective optimization problems is proposed by Martin et al. in [Mar+16]. It is similar to the one of [FT09], but it can handle equality constraints. In addition, more discarding tests are applied and it uses a much more elaborated procedure to prune considered boxes to their interesting areas. The paper by Araya et al. [ACA19] proposes further improvements. Recently, another algorithm for biobjective optimization problems was proposed by A. and J. Žilinskas, see [ŽŽ16] and also [PŽŽ17]. They use the Lipschitz property of the objective functions and iterative trisections of the feasible set which is assumed to be a box. An extension to more general constraints might be possible, though it is not obvious how to accomplish this.

Another algorithm for multiobjective optimization problems with box constraints was introduced by Evtushenko and Posykin in [EP13; EP14]. They propose an algorithm

which guarantees a certain accuracy of the determined solutions in the image space. The multiobjective algorithm by Scholz, [Sch12], also provides accuracies of computed solution. This accuracy concept is very similar to the one we use in this thesis. Moreover, some more discarding tests also for unconstrained optimization problems are introduced in the work by [Sch12]. These tests work with necessary optimality conditions. In Section 4.5, we consider most of the known algorithms for multiobjective optimization problems in detail.

1.2.2 Multiobjective Mixed Integer Nonlinear Optimization

So far, most of the existing algorithms for multiobjective mixed integer optimization are only for *linear* optimization problems. Those can be divided into two main classes: decision space search algorithms, i. e., approaches that work in the space of feasible points, and criterion space search algorithms, i. e., methods that work in the image space.

Among the decision space search algorithms, the method proposed by Mavrotas and Diakoulaki, [MD98], is the first B&B algorithm for solving multiobjective mixed binary programs. The authors improved and extended their work in [Mav09; MD05]. Other approaches defining B&B algorithms for multiobjective integer linear programming problems are [EG07; SS08]. There, the aim for the bounding procedure is to define proper hyperplanes in the objective space in order to separate the upper and lower bound sets. Our new algorithms which will be proposed in Chapter 6 belong to the decision space search algorithms.

On the other hand, criterion space search algorithms find images of optimal solutions by addressing a sequence of single objective optimization problems. Once an image of an optimal solution is computed, dominated parts of the criterion space are removed and the algorithms continue the search for new images of optimal solutions. Several contributions in the context of criterion space search algorithms for biobjective and triobjective integer linear programming are given by Boland and co-authors, see [BCS15; BCS16; BCS17a; BCS17b].

As far as we know, the first general purpose method to tackle multiobjective mixed integer *convex* programs is the heuristic approach based on a B&B algorithm proposed by Cacchiani and D'Ambrosio in [CD17].

A classical technique to solve a multiobjective optimization problem is to convert the problem into a parameter-dependent single objective one, known as scalarization. This approach was recently followed by Burachik et al. [BKR19] for multiobjective mixed integer optimization problems (see also the comment in the conclusions of [BKR17]). The scalarized problems are then parameter-dependent single objective mixed integer convex optimization problems. By following this approach, many of these single objective problems have to be solved, one for each choice of the parameter's value. No gained information of pre-solved problems are typically used thereby. Furthermore, it is not clear how to choose the parameter values in a smart way and this is an open challenge: as the set of the images of optimal solutions is in general disconnected and can have huge gaps, many subproblems defined according to different parameter's values might lead to the same objective values and, thus, solving such subproblems is a wasted effort.

Contrary to the heuristic method in [CD17] and the scalarization approach in [BKR19], we propose a procedure which works analytically, and uses already gained information throughout the algorithm.

1.2.3 Dealing with Uncertainties

Dealing with multiobjective optimization for real life problems can lead to an additional difficulty. Often, the calculated optimal solutions cannot be used precisely, because they can only be realized within a certain accuracy. This is, for instance, the case, when a magnet system for a measurement technique for electrically conducting fluids should be constructed, see [EKS17]. There, the optimal direction of the magnetization of each magnet and the optimal magnetization have to be determined such that the so-called Lorentz force is maximized and the weight of the system is minimized. In practice, an (optimally) chosen magnetic direction cannot be realized in any arbitrary accuracy, as magnets can only be produced within some tolerances. Therefore, decision uncertainty has to be taken into account. Another example is the growing media

mixing problem for a plant nursery, see [EKS17; Krü+18; Krü18b]. There, a mixture of peat and compost for the growing media has to be determined, which can also not be mixed exactly by workers.

Also, in case of such uncertainties in the realization of variables, the actual realized solutions should lead to near-optimal values. This kind of uncertainty in optimization is called *decision uncertainty*, which should be distinguished from *parameter uncertainty*, because the inaccuracies are caused by the decision variables, [EKS17]. Parameter uncertainty was considered in several works, for instance, in [BGN09] in the single objective case, or in [FW14; Gob+14; KL12] for the multiobjective case.

Decision uncertainty for single objective optimization problems has been handled with minmax robustness under different names like *robust optimization with implementation error*, e. g., in [BN02], or robust regularization, e. g. in [LP09]. In multiobjective optimization there are also different approaches to treat uncertainties, such as sensitivity analysis [BA06], or evaluating the mean or integral of each objective over the set of possible values of a solution [DG06], or adding a robustness measure as a new objective function [ZMK19]. We follow the so-called *worst-case robustness approach* as it was done in [EKS17], see also [Krü18a].

In the worst case approach one considers all possible outcomes leading to sets which have to be compared. In the single objective case these sets are just intervals, which can be compared much easier numerically. In case of a multiobjective optimization problem with m objectives we have to compare subsets of $\mathbb{R}^m$. This means, we have to solve a specific set-valued optimization problem with a certain set-order relation to handle the uncertainties.

In set-valued optimization different possibilities to compare sets are discussed in the literature. In case of the worst-case approach we have to use the upper-type less order relation, see, for instance, [Kur98; JH11]. When comparing whole sets one also speaks of the set approach in set-valued optimization. So far, there is only a limited number of numerical algorithms to solve such set optimization problems. For unconstrained set-valued optimization and a similar order relation, Löhne and Schrage introduced an algorithm in [LS13c], which is only applicable for linear problems. Jahn presented some derivative-free algorithms, see [Jah15], to find one single solution of the whole set of optimal solutions in case the sets which have to be compared are convex. Köbis

and Köbis extended the method from [Jah15] to the nonconvex case, i. e., when the sets are nonconvex, see [KK16]. However, all methods aim at finding only one single minimal solution. In [Jah18] for the first time a method for nonconvex sets is presented, which uses discretization to compare sets and which can find many minimal solutions, but still not all. The procedure was parallelized and implemented on a CPU and GPU. As set-valued optimization problems have in general an infinite number of optimal solutions, a representation of the whole set of optimal solutions is of interest.

In [EKS17], see also [Krü+18; Krü18b], for some specific multiobjective decision uncertain optimization problems, solution approaches or characterizations of the optimal solution set have been provided. But they are all for specific cases only, as for linear or for monotone objective functions.

1.3 Main Contribution of this Thesis

This thesis proposes a series of algorithms for solving multiobjective optimization problems with different issues. The difficulty of the first type of multiobjective optimization problems is that the objective functions can be nonconvex and we aim for globally optimal solutions. The new algorithm `MOPBB` which handles that kind of optimization problem is one of the first deterministic algorithms which provides an accuracy of the computed solutions. Additionally, the lower bounding procedure establishes non-singleton lower bounds which are described by hyperplanes. Those hyperplanes are constructed only if they improve a lower bound in order to discard a subbox. Thereby, we use existing methods and ideas, and combine them to a new discarding procedure. This work resulted in a publication, see [NE19].

The methods utilized in algorithm `MOPBB` can also be applied to other optimization problems like mixed integer problems. Considering such optimization problems, one has to deal with a nonconvex feasible set, because at least one of the variables is restricted to be integer. We address this difficulty with similar methods as the ones for the multiobjective nonconvex optimization problem with continuous variables. For a first simplification, the objective functions of the multiobjective mixed integer problem are assumed to be convex. The resulting algorithm is the first one which is

able to approximate the optimal solution set of a multiobjective mixed integer convex optimization problem. About those results, we have written a paper submitted for publication, see [DeS+19]. This article is based on a master thesis, see [Roc18], whose main ideas were improved. My contribution to the article [DeS+19] includes the main work on the article (writing, proving) in cooperation with the first author. For the numerical part, I added the new main idea to the existing implementation, improved it and tested the procedure extensively.

The third difficulty in the context of multiobjective optimization considered in this thesis is uncertainty in the realization of the decision variables of a multiobjective optimization problem. We model such problems by a set optimization problem. The images of the objective function of such optimization problems are sets. For comparing these sets in the image space, we need lower and upper bounds. Thus, we developed a completely new bounding procedure for those special types of sets and an algorithm to obtain a covering of the optimal solution set in the preimage and the image space. The base of this topic is a bachelor thesis supervised by myself, see [Roc16], which resulted in a publication [ENR19]. Besides the supervision, my contribution to this work includes collecting, summarizing, and checking of all important facts. Moreover, the implementation by the bachelor student was improved and exhaustively tested by myself.

2 Basics of Multiobjective Optimization

This section is dedicated to the basics of solving multiobjective optimization problems. Since we want to reach a minimum, we have to compare vectors in the multiobjective context. A short introduction to order relations is given in Section 2.1. Although we will only use the typical order relation in $\mathbb{R}^m$ in most of the chapters, this is also the basis for dealing with set order relations in Section 7.1.2. Next, we introduce the basic multiobjective optimization problem as well as the notions for the optimal solutions of this problem. Additionally, some of the different concepts for approximate solutions of a multiobjective optimization problem are mentioned and discussed in Section 2.4.

2.1 Order Relations

Order relations are a common concept in mathematics as they allow to compare two elements of a set. Let S be an arbitrary set. A *binary relation* $\mathcal{R}$ on S is a subset of $S \times S$.

Definition 2.1 [Ehr05, Def. 1.4] Let $\mathcal{R} \subseteq S \times S$ be a binary relation on S. The relation $\mathcal{R}$ is called

(i) *reflexive* if $(x, x) \in \mathcal{R}$ for all $x \in S$;
(ii) *antisymmetric* if $(x, y) \in \mathcal{R}$ and $(y, x) \in \mathcal{R} \Longrightarrow x = y$ for all $x, y \in S$;
(iii) *transitive* if $(x, y) \in \mathcal{R}$ and $(y, z) \in \mathcal{R} \Longrightarrow (x, z) \in \mathcal{R}$ for all $x, y, z \in S$;
(iv) *total* if $(x, y) \in \mathcal{R}$ or $(y, x) \in \mathcal{R}$ for all $x, y \in S$.

Order relations are special binary relations on S which fulfill transitivity, see Definition 2.1 (iii). Other special order relations are defined as follows:

Definition 2.2 [Ehr05, Def. 1.5, 1.8, 1.10] Let $\mathcal{R} \subseteq S \times S$ be a binary relation on S. The relation $\mathcal{R}$ is called

(i) a *preorder/quasi order* if $\mathcal{R}$ is reflexive and transitive;

(ii) a *partial order* if $\mathcal{R}$ is reflexive, transitive and antisymmetric;

(iii) a *total order* if $\mathcal{R}$ is reflexive, transitive, antisymmetric and total.

Remark 2.3 [Jah13, Def. 1.6, Th. 1.18, Def. 1.19] If S is a real linear space, we additionally assume that a partial order $\mathcal{R}$ is compatible with the linear structure of S, i. e., it holds

(i) $(x, y) \in \mathcal{R}, (w, z) \in \mathcal{R} \Longrightarrow (x + w, y + z) \in \mathcal{R}$ for all $x, y, w, z \in S$;

(ii) $(x, y) \in \mathcal{R}, \alpha \in \mathbb{R}_+ \Longrightarrow (\alpha x, \alpha y) \in \mathcal{R}$.

In a real linear space, a partial order relation $\mathcal{R}$ is induced by a convex cone $K \subseteq S$, i. e.,

$$(x, y) \in \mathcal{R} \Leftrightarrow y - x \in K.$$

We call K an *ordering cone* (of the relation $\mathcal{R}$).

As usual, we also write $x\mathcal{R}y$ instead of the pairwise form $(x, y) \in \mathcal{R}$. In this work, we mainly compare elements of the spaces $\mathbb{R}$ and $\mathbb{R}^m$. Therefore, we deal with the componentwise order relations. For the m-dimensional real space and $y^1, y^2 \in \mathbb{R}^m$ we consider:

$$\begin{aligned} y^1 \leq y^2 &\Leftrightarrow y^2 - y^1 \in \mathbb{R}^m_+ &&\Leftrightarrow y^1_j \leq y^2_j \text{ for all } j = 1, \ldots, m \\ y^1 < y^2 &\Leftrightarrow y^2 - y^1 \in \operatorname{int}(\mathbb{R}^m_+) &&\Leftrightarrow y^1_j < y^2_j \text{ for all } j = 1, \ldots, m \end{aligned}$$

The order relation $\leq$ is a partial order relation on $\mathbb{R}^m$ whereas only transitivity holds for $<$. Thus, $<$ is neither a partial order nor a preorder.

In addition to these relations, we make use of the following ones:

$$\begin{aligned} y^1 \lneq y^2 &\Leftrightarrow y^1 \leq y^2 \text{ and } y^1 \neq y^2 \\ y^1 \not\leq y^2 &\Leftrightarrow y^1_j > y^2_j \text{ for at least one } j = 1, \ldots, m. \end{aligned}$$

In case of $y^1 \not\leq y^2$, we say that y^1 *and* y^2 *cannot be compared.*

2.2 The Multiobjective Optimization Problem

Let $M \subseteq \mathbb{R}^n$ be a nonempty set and $f_1, \ldots, f_m \colon \mathbb{R}^n \to \mathbb{R}$ be the given objective functions. The general multiobjective optimization problem is defined by:

$$\begin{aligned} \min \quad & f(x) = \begin{pmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{pmatrix} \\ \text{subject to (s. t.)} \quad & x \in M \end{aligned} \tag{P}$$

Throughout this thesis, we assume different properties for the optimization problem which has to be solved by the proposed algorithms. At the moment, we only assume, that the objective functions $f_j, j = 1, \ldots, m$ are continuous differentiable and the feasible set M is non-empty and compact.

All more specific assumptions are stated in the relevant sections. The image of a set $S \subseteq M$ with respect to (w. r. t.) f is denoted by $f(S) := \{f(x) \mid x \in S\}$.

2.3 Optimality Notions in Multiobjective Optimization

Dealing with multiple objective functions and the partial order relation $\leq$ on $\mathbb{R}^m$ leads to the following optimality concept. In general, there is no feasible point which minimizes each objective function at the same time. Instead, there are several feasible points whose objective values even cannot be compared, and in most circumstances, it is not clear which point of them leads to the smaller objective values. The most common optimality concept for multiobjective optimization problems is *efficiency*.

Definition 2.4 [Ehr05, Def. 2.1, Def. 2.24] Let the optimization problem (P) be given.

(i) A point $x^* \in M$ is said to be an *efficient point/solution* for (P) if there exists no $x \in M$ such that

$$f(x) \leq f(x^*),$$

i. e., such that $f(x) \leq f(x^*)$ and $f(x) \neq f(x^*)$.

(ii) A point $x^* \in M$ is said to be *strictly efficient* for (P) if there exists no $x \in M$, $x \neq x^*$ such that $f(x) \leq f(x^*)$.

(iii) The set of all efficient points is called *efficient set* and is denoted by X_E.

(iv) We say x^1 *dominates* x^2 if $x^1, x^2 \in M$ and $f(x^1) \lneq f(x^2)$ hold.

Definition 2.4 gives a notion for the variables of f, i. e., for points of the preimage space. The next definition states some terms for the points and sets in the image space.

Definition 2.5 [Ehr05, Def. 2.1] Let the optimization problem (P) be given.

(i) A point $y^* = f(x^*)$ is said to be *nondominated* for (P) if x^* is efficient for (P).

(ii) The set of all nondominated points is called *nondominated set* of (P).

The next definition states common notions to say whether a point in the image space is "better" than another one. Also, we give a definition for a special property of a subset of the m-dimensional space.

Definition 2.6 Let $y^1, y^2 \in \mathbb{R}^m$ be given

(i) We say y^1 *dominates* y^2 if $y^1, y^2 \in \mathbb{R}^m$ and $y^1 \lneq y^2$.

(ii) We say y^1 *strictly dominates* y^2 if $y^1, y^2 \in \mathbb{R}^m$ and $y^1 < y^2$.

(iii) A set $\mathcal{N} \subseteq \mathbb{R}^m$ is said to be *stable* if for any $y^1, y^2 \in \mathcal{N}$ either $y^1 \not\leq y^2$ or $y^1 = y^2$ holds.

Figure 2.1 shows an image set of a biobjective optimization problem.

The thick curve on the lower left boundary of $f(M)$ is the nondominated set. The point y^* belongs to this set, because $\{y^*\} - (\mathbb{R}^m_+ \setminus \{0\})$ does not contain any other image point. Thus, y^* is called nondominated. On the other hand, y' is dominated by y^* and, therefore, not a nondominated point of this biobjective optimization problem.

Remark 2.7 In the literature there are several terms for *efficient* and *nondominated* like *solution*, *minimal*, *Pareto-optimal*, *Edgeworth-Pareto/EP-optimal*, *Paretofront*. [Ehr05, Table 2.4] gives an extensive overview of the different notions for the concepts we use.

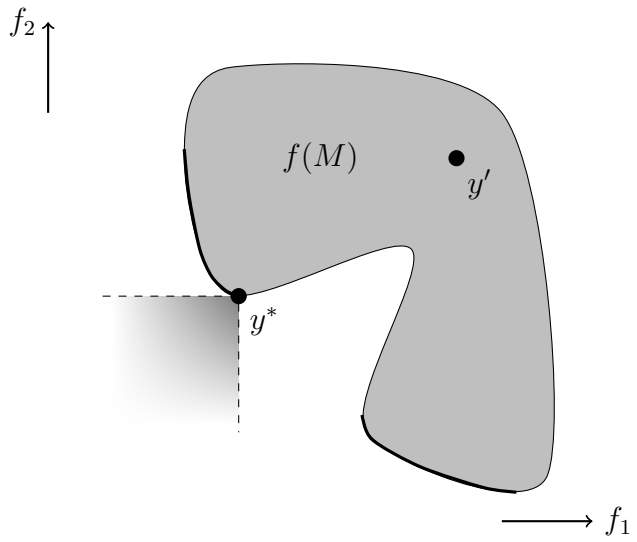


Figure 2.1. Image set of a biobjective optimization problem.

2.4 Approximate Solutions of Multiobjective Problems

The cardinality of the efficient and nondominated set is not finite in general. Therefore, an algorithm for multiobjective optimization problems cannot find all efficient or nondominated points. Instead, it usually calculates a finite representation or approximation of those sets. For real-life problems these methods are more suitable because a user has to choose from a finite set rather than from an infinite one. The algorithms still have to ensure that they determine "nearly"-efficient or "nearly"-nondominated points. We call those points *approximate solutions* — no matter if they belong to the preimage or image set — because they are in some way close to an efficient or nondominated point, respectively. Different formal definitions for approximate solutions are known in the literature. We start by the one utilized in this work. Let $e \in \mathbb{R}^m$ be the m-dimensional all-ones vector $(1, \ldots, 1)^T$.

Definition 2.8 Let $\varepsilon \geq 0$ and $\delta \geq 0$ be given.

(i) A point $\bar{x} \in M$ is an *ε-efficient point* of (P) if there exists no $x \in M$ with

$$f(x) \leqq f(\bar{x}) - \varepsilon e.$$

(ii) A set $\mathcal{A} \subseteq M$ is an (ε, δ)-*efficient set* of (P) if every point of $\mathcal{A}$ is an ε-efficient point of (P) and if for all $x^* \in X_E$ there is an $\bar{x} \in \mathcal{A}$ with $\|\bar{x} - x^*\| \leq \delta$.

This definition of ε-efficiency was introduced in an equivalent version in [Kut79]. A more general concept for approximate solutions for vector optimization can be found in [GJN06].

Besides the notion of Definition 2.8, there are more concepts for approximate solutions of a multiobjective optimization problem known. In what follows, we introduce those concepts and their relation to ε-efficient points and (ε, δ)-efficient sets from Definition 2.8. Those notions are used for the computed points obtained by other procedures which solve multiobjective optimization problems. These procedures will be considered in Section 4.5 in more detail. Most of the following propositions which state the relation of one concept to ε-efficiency are not proved as the relations are very simple.

We start by a concept similar to ε-efficiency, that is also presented in [Lor84].

Definition 2.9 [Sch12, Def. 4.4] For a given $\varepsilon \in \mathbb{R}^m_+$, a point $\hat{x} \in M$ is called *ε-Pareto optimal* for (P) if there is no $x \in M$ satisfying

$$f(x) \lneq f(\hat{x}) - \varepsilon.$$

Note that ε-Pareto optimality is defined for a vector $\varepsilon \in \mathbb{R}^m$, while ε-efficiency is defined for a scalar ε. Nevertheless, both concepts are closely related as follows:

Proposition 2.10 *Let a multiobjective optimization problem (P) be given. Every ε-efficient point of (P) is an εe-Pareto optimal point of (P).*

Therefore, ε-Pareto optimality is a generalization of ε-efficiency as different accuracies for each objective function are allowed. If different accuracies for each objective function of a multiobjective optimization problem are requested for ε-efficient points, the objective functions can be multiplied by appropriate scalars. Additionally, it is easy to see that efficient points of a multiobjective optimization problem are efficient as well when the objective functions are multiplied by positive scalars.

Another concept defines approximate solutions in the image space:

Definition 2.11 [EP14] Let (P) and a scalar $\varepsilon \geq 0$ be given. A discrete set of points $Y^\varepsilon \subseteq f(M)$ is called *ε-Pareto set* for (P) if the following conditions hold:

(i) For any nondominated point y^*, there exists a point $y^\varepsilon \in Y^\varepsilon$ such that

$$y^\varepsilon - \varepsilon e \leq y^*.$$

(ii) Y^ε is a stable set (see Definition 2.6 (iii)).

This definition differs from our definition about ε-efficient points and (ε, δ)-efficient sets. First, an ε-Pareto set Y^ε is defined in the image space, while an (ε, δ)-efficient set $\mathcal{A}$ is a subset of the preimage space. Second, Y^ε has to be stable, whereas this is not required for the image set of $\mathcal{A}$. If we take a subset $\mathcal{A}' \subseteq \mathcal{A}$ such that $f(\mathcal{A}')$ is a stable set, it may happen that there is a δ-neighborhood of an efficient point which does not contain any ε-efficient point of the new set $\mathcal{A}'$. In this case, $\mathcal{A}'$ is not an (ε, δ)-efficient set. The following proposition states the similarities and a further slight difference of both concepts.

Proposition 2.12 *Let a multiobjective optimization problem (P) and an ε-Pareto set Y^ε of (P) be given. Moreover, consider a point $y^\varepsilon \in Y^\varepsilon$ for which a nondominated point y^* with $y^\varepsilon - \varepsilon e \leq y^*$ exists. Then there is an ε-efficient point $x^\varepsilon \in M$ with $f(x^\varepsilon) = y^\varepsilon$.*

Proof. Let y^ε be chosen from the set Y^ε such that a nondominated point y^* exists with $y^\varepsilon - \varepsilon e \leq y^*$. Assume that a preimage x^ε of y^ε is not ε-efficient. Thus, there is an $x \in M$ with $f(x) \neq f(x^\varepsilon) = y^\varepsilon$ with $f(x) \lneq y^\varepsilon - \varepsilon e$. It follows $f(x) \lneq y^\varepsilon - \varepsilon e \leq y^*$, which contradicts the fact that y^* is nondominated for (P). Hence, x^ε is ε-efficient for (P). □

The next concept by [FT09] is defined for biobjective optimization problems:

Definition 2.13 [FT09, Def. 16] Let $x^1, x^2 \in M$ be two feasible points, and $\beta_1, \beta_2 \geq 0$ two non-negative scalars. We say that x^1 *(β_1, β_2)-superdominates* x^2 if it holds

$$f_1(x^1) + \beta_1 < f_1(x^2) \text{ and } f_2(x^1) + \beta_2 < f_2(x^2).$$

Proposition 2.14 *Let a biobjective optimization problem, i. e., (P) with $m = 2$, be given. An ε-efficient point for (P) is not $(\varepsilon, \varepsilon)$-superdominated by any other feasible point of (P). The converse does not hold in general.*

Proof. The first assertion is easy to see. To show that the converse does not hold, let $\bar{x} \in M$ be a point, which is not $(\varepsilon, \varepsilon)$-superdominated by any other feasible point. This means, there is no $x \in M$ with $f_1(x) < f_1(\bar{x}) - \varepsilon$ and $f_1(x) < f_2(\bar{x}) - \varepsilon$. It is still possible that there is a $\tilde{x} \in M$ for which $f_1(\tilde{x}) = f_1(\bar{x}) - \varepsilon$ and $f_2(\tilde{x}) < f_2(\bar{x}) - \varepsilon$ hold. In this case $\bar{x}$ is not ε-efficient of (P). □

3 Optimization Tools Utilized for the New Algorithms

In this chapter, we summarize some known results and techniques which will be used for the new algorithms described in the following chapters. The first section considers *interval arithmetic* which is often used in global single objective optimization to obtain lower and upper bounds for the globally minimal value. Another method for obtaining lower bounds by convex underestimators is explained in the next section. Section 3.3 introduces an algorithm which solves multiobjective linear or multiobjective convex optimization problems. Thereafter, another concept from multiobjective (combinatorial) optimization is the topic of Section 3.4. Finally, we briefly introduce a general B&B algorithm in Section 3.5.

3.1 Interval Arithmetic

In the following, we present the concept of interval arithmetic which is a common tool in global optimization to handle n-dimensional intervals and interval-valued functions. For a more extensive introduction to interval arithmetic, we refer the reader to the common literature, see e.g. [Han92; Moo66; MKC09; Neu90]. A set $X \subseteq \mathbb{R}^n$ is called *n-dimensional box* (or hyper rectangle) if there are two vectors $\underline{x}, \overline{x} \in \mathbb{R}^n$ with $\underline{x} \leq \overline{x}$ such that

$$X := [\underline{x}, \overline{x}] := \{x \in \mathbb{R}^n \mid \underline{x} \leq x \leq \overline{x}\}.$$

For a box $X := [\underline{x}, \overline{x}]$ we define

$$\inf(X) := \underline{x} \text{ and } \sup(X) = \overline{x}.$$

The set of all n-dimensional real boxes is denoted by $\mathbb{IR}^n$. The width of an n-dimensional box $X = [\underline{x}, \overline{x}] \in \mathbb{IR}^n$ is defined as

$$\omega(X) := \|\overline{x} - \underline{x}\|, \tag{3.1}$$

where $\|\cdot\|$ is the Euclidean norm. It is possible to define the basic arithmetic operations to enable the calculation with intervals or boxes.

Definition 3.1 [MKC09, (2.19)] Let $X, Y \in \mathbb{IR}$ and $* \in \{+, -, \cdot, /\}$. Then we define

$$X * Y = \{x * y \mid x \in X, y \in Y\},$$

where X/Y is only defined if $0 \notin Y$ holds.

In addition, this can be extended to *elementary functions*. Elementary functions belong to the set of real, continuous (on every closed interval on which they are defined) functions, [Neu90]. For example, typical elementary functions are $|x|$, $\sqrt{x}$, $\exp x$, $\ln x$, $\sin x$, $\cos x$ and $\arctan(x)$. Usually, toolboxes which enable interval arithmetic work with libraries to allow the box-valued evaluation of elementary functions. With those tools it is further possible to combine elementary functions and interval operations $(+, -, \cdot, /)$, and thus, to evaluate more complicated functions on a box:

Definition 3.2 [Ste17; McC76] A function $h\colon \mathbb{R}^n \to \mathbb{R}$ is *factorable* if it can be formulated as a concatenation of basic arithmetic operations $(+, -, \cdot, /)$ and elementary functions.

For every factorable function we can now define interval extensions and the natural interval extension.

Definition 3.3 [MKC09, Def. 5.2] A function $H\colon \mathbb{IR}^n \to \mathbb{IR}$ is an *interval extension* of $h\colon \mathbb{R}^n \to \mathbb{R}$ if it holds

$$H([x_1, x_1], \ldots, [x_n, x_n]) = [h(x_1, \ldots, x_n), h(x_1, \ldots, x_n)] \text{ for all } x \in \mathbb{R}^n.$$

Note that $[h(x_1, \ldots, x_n), h(x_1, \ldots, x_n)]$ is an interval containing only one point, namely $h(x_1, \ldots, x_n)$.

Definition 3.4 [MKC09, Section 5.4] For a factorable function $h\colon \mathbb{R}^n \to \mathbb{R}$ the *natural interval extension* is the function $H\colon \mathbb{IR}^n \to \mathbb{IR}$ which is obtained if every occurrence of x_i in h is replaced by $X_i,\ i = 1 \ldots, n$, and all basic arithmetic operations and elementary functions are interpreted as interval operations.

The natural interval extension is a special interval extension. Since a function can usually be described by different formulas, each of these formulas leads to another natural interval extension. For example, the function h defined by $h(x) = x - x^2$ can be written by $\tilde{h}(x) = x(1-x)$ as well. Both natural interval extensions $H(X) = X - X \cdot X$ and $\tilde{H}(X) = X \cdot (1 - X)$ are interval extensions for h, but give in general different results when they are evaluated for a box X. Upcoming issues are extensively discussed in common literature, [Han92; Moo66; MKC09; Neu90]. An important property for interval-valued functions is inclusion isotonicity:

Definition 3.5 [MKC09, Def. 5.4] A function $H\colon \mathbb{IR}^n \to \mathbb{IR}$ is *inclusion isotonic* if $\tilde{X}_i \subseteq X_i,\ i = 1, \ldots, n$, implies $H(\tilde{X}_1, \ldots, \tilde{X}_n) \subseteq H(X_1, \ldots, X_n)$.

Remark 3.6 [MKC09, Section 5.4] By Definition 3.1, the operators $+, -, \cdot, /$ on $\mathbb{R}$ are inclusion isotonic. Toolboxes for interval arithmetic are implemented in such a way that the natural interval extension is also inclusion isotonic. For calculations for this thesis, we use the MATLAB toolbox Intlab [Rum99] for interval arithmetic which works with inclusion isotonic natural interval extensions.

Theorem 3.7 *[Moo66] Define* $h(X_1, \ldots, X_n) := \{h(x) \mid x_i \in X_i \text{ for all } i = 1 \ldots, n\}$. *If* H *is an inclusion isotonic interval extension of* h*, it holds that*

$$h(X_1, \ldots, X_n) \subseteq H(X_1, \ldots, X_n).$$

This is the fundamental theorem of interval arithmetic and the reason why interval arithmetic is a common tool for calculating lower bounds of function values on a given

box. As the image set of h on $X \in \mathbb{IR}^n$ is always a subset of the image box obtained by an inclusion isotonic interval extension H to X, $\inf(H(X))$ is a lower bound for all $h(x)$ with $x \in X$.

3.2 Convex Underestimators

Another way to calculate lower bounds for real-valued functions was proposed in [MF94] under the name αBB and is based on the concept of convex underestimators.

Definition 3.8 Let a function $h\colon X \to \mathbb{R}$ on a box $X = [\underline{x}, \overline{x}] \in \mathbb{IR}^n$ be given. A function $\widehat{h}\colon X \to \mathbb{R}$ is called *convex underestimator* for h on X if $\widehat{h}$ is a convex function with $\widehat{h}(x) \leq h(x)$ for all $x \in X$.

Lemma 3.9 *[MF94, Property 1 and 3] Let a twice continuously differentiable function $h\colon X \to \mathbb{R}$ on a box $X = [\underline{x}, \overline{x}] \in \mathbb{IR}^n$ be given. Let $\lambda_{\min}(x)$ denote the smallest eigenvalue of the Hessian $H_h(x)$ of h at x and choose an $\alpha \geq \max\{0, -\min_{x\in X} \lambda_{\min}(x)\}$. Then the function $h_\alpha\colon X \to \mathbb{R}$ with*

$$h_\alpha(x) := h(x) + \frac{\alpha}{2}(\underline{x} - x)^T(\overline{x} - x) \tag{3.2}$$

is a convex underestimator for h on X and is called αBB-underestimator

A lower bound for $\lambda_{\min}(x)$ over X can be calculated with the help of interval arithmetic and Gerschgorin's theorem:

Lemma 3.10 *[Adj+98, Theorem 3.2] Let $\frac{\partial^2}{\partial x_i \partial x_j} H(X) =: [\underline{a}_{ij}, \overline{a}_{ij}]$ denote the box obtained by the natural interval extension of the second partial derivative of h on the box X according to the i-th and j-th coordinate for all $i, j = 1, \ldots, n$. A lower bound on the minimum eigenvalue $\min_{x\in X} \lambda_{\min}(x)$ is given by*

$$\min_{x\in X} \lambda_{\min}(x) \geq \min_{i=1,\ldots,n} \left\{ \underline{a}_{ii} - \sum_{j\neq i} \max\{|\underline{a}_{ij}|, |\overline{a}_{ij}|\} \right\}.$$

According to this lemma, we can use any toolbox for interval arithmetic to evaluate the right hand side of the equation in Lemma 3.10. Thus, a suitable α is given by

$$\alpha = \max\left\{0, -\min_{i=1,\dots,n}\left\{\underline{a}_{ii} - \sum_{j\neq i}\max\{|\underline{a}_{ij}|, |\overline{a}_{ij}|\}\right\}\right\}. \tag{3.3}$$

For additional and improved lower bounds for $\lambda_{\min}(x)$, see also [Adj+98; SM16].

The main benefit of convex underestimators is that the minimal value of $\widehat{h}$ over X, which can be calculated by standard techniques from convex optimization, yields a lower bound for the values of h on X. There are also other possibilities for the calculation of convex underestimators. For example, in [Adj+98] special convex underestimators for bilinear, trilinear, fractional, fractional trilinear or univariate concave functions were defined. Here, we restrict ourselves to the above proposed convex underestimator. The theoretical results remain true if the above underestimators are replaced by tighter ones. In this work, we only make use of αBB-underestimators. Therefore, "αBB underestimator" is meant implicitly when "convex underestimator" is written.

We can state the following lemma:

Lemma 3.11 *Let $h\colon \mathbb{R} \to \mathbb{R}$, a box $X = [\underline{x}, \overline{x}] \in \mathbb{IR}$ and a subbox $\tilde{X} = [\tilde{\underline{x}}, \tilde{\overline{x}}] \subseteq X$ be given. Define*

$$\begin{aligned}
h_\alpha(x) &:= h(x) + \frac{\alpha}{2}(\underline{x} - x)^T(\overline{x} - x),\\
h_{\tilde{\alpha}}(x) &:= h(x) + \frac{\tilde{\alpha}}{2}(\underline{x} - x)^T(\overline{x} - x),\\
\tilde{h}_\alpha(x) &:= h(x) + \frac{\alpha}{2}(\tilde{\underline{x}} - x)^T(\tilde{\overline{x}} - x),\\
\tilde{h}_{\tilde{\alpha}}(x) &:= h(x) + \frac{\tilde{\alpha}}{2}(\tilde{\underline{x}} - x)^T(\tilde{\overline{x}} - x),
\end{aligned}$$

where $\alpha \geq \max\{0, -\min_{x\in X}\lambda_{\min}(x)\}$ and $\tilde{\alpha} \geq \max\{0, -\min_{x\in\tilde{X}}\lambda_{\min}(x)\}$. In case of $\tilde{\alpha} \leq \alpha$, it holds:

(i) $h_\alpha, h_{\tilde{\alpha}}, \tilde{h}_\alpha$ *and* $\tilde{h}_{\tilde{\alpha}}$ *are convex underestimators for* h *on* $\tilde{X}$.

(ii) $\tilde{h}_{\tilde{\alpha}}(x) \geq h_{\tilde{\alpha}}(x) \geq h_\alpha(x)$ *holds for all* $x \in \tilde{X}$.

(iii) $\tilde{h}_{\tilde{\alpha}}(x) \geq \tilde{h}_\alpha(x) \geq h_\alpha(x)$ *holds for all* $x \in \tilde{X}$.

Proof. The two chains of inequalities in (ii) and (iii) follow by using the properties $\underline{x} \leq \underline{\tilde{x}} \leq x \leq \overline{\tilde{x}} \leq \overline{x}$ for all $x \in \tilde{X}$ and $\tilde{\alpha} \leq \alpha$. Using this, we obtain that all functions are underestimators for h on $\tilde{X}$. The convexity of all functions follows immediately by Lemma 3.9. □

Remark 3.12 If $\tilde{\alpha}$ and α are calculated by Intlab, [Rum99], we always obtain $\tilde{\alpha} \leq \alpha$ because the natural interval extensions are inclusion isotonic. Thus, the requirements of Lemma 3.11 are fulfilled in our context.

The consequence of Lemma 3.11 is that we can obtain better convex undererstimators on a subbox (in the sense that they are closer to the original function) if we recalculate the parameter α or use the boundaries of the subbox. This is actually done in the algorithm. However, for simplicity of presentation in this work, we use only the parameter α for which h_α is a convex underestimator of h on the initial box X even if we consider the function on a subbox of X.

An important benefit, especially of αBB-underestimators, is stated in the following remark.

Remark 3.13 [MF94, Property 4] For all $\alpha \geq 0$ the maximal pointwise difference between h and h_α is $\frac{\alpha}{2}\omega(X)^2$, i. e., $\max_{x \in X} |h(x) - h_\alpha(x)| = \frac{\alpha}{2}\omega(X)^2$ with $\omega(X)$ is the box width of a box X defined in (3.1).

This remark states that the maximal pointwise difference of a factorable function and its αBB underestimator is bounded by the parameter α and the box width of the box on which the underestimator is defined. In the next lemma, we show that the distance between the minimal value of a convex underestimator and other function values of a smooth function h over a box is bounded by a given $\varepsilon > 0$ if the box width is small enough.

Lemma 3.14 *Let a box $X \in \mathbb{IR}^n$, a twice continuously differentiable nonconvex function $h\colon \mathbb{R}^n \to \mathbb{R}$, a constant $\alpha \geq \max\{0, -\min_{x \in X} \lambda_{\min}(x)\}$ and a positive scalar $\varepsilon > 0$ be*

given. Moreover, chose $L > 0$ *such that* $L \geq \sqrt{n} \left| \frac{\partial}{\partial x_i} h(x) \right|$ *holds for all* $i \in \{1, \ldots, n\}$ *and* $x \in X$. *Let* $\tilde{X} = [\underline{\tilde{x}}, \overline{\tilde{x}}]$ *be a box with* $\tilde{X} \subseteq X$ *and*

$$\omega(\tilde{X}) \leq -\frac{L}{\alpha} + \sqrt{\frac{L^2}{\alpha^2} + \frac{\varepsilon}{\alpha}} =: \delta_X. \tag{3.4}$$

Furthermore, let $\tilde{h}_\alpha \colon \mathbb{R}^n \to \mathbb{R}$ *defined by* $\tilde{h}_\alpha(x) := h(x) + \frac{\alpha}{2}(\underline{\tilde{x}} - x)^T(\overline{\tilde{x}} - x)$ *be a convex underestimator of* h *on* $\tilde{X}$. *Then for* $v := \min_{x \in \tilde{X}} \tilde{h}_\alpha(x)$ *it holds that* $|h(x) - v| \leq \frac{\varepsilon}{2}$ *for all* $x \in \tilde{X}$.

Proof. Note that $\alpha \neq 0$ because h is nonconvex. Let $\tilde{x}$ be a minimal solution of $\min_{x \in \tilde{X}} \tilde{h}_\alpha(x)$, i. e., $v = \tilde{h}_\alpha(\tilde{x})$. With Remark 3.13, it follows that

$$|h(\tilde{x}) - v| = |h(\tilde{x}) - \tilde{h}_\alpha(\tilde{x})| \leq \frac{\alpha}{2}\omega(\tilde{X})^2.$$

Let $x, y \in \tilde{X}$ be arbitrarily chosen. By the mean value theorem there exists a point $\xi \in \{\lambda x + (1-\lambda)y \in \mathbb{R}^n \mid \lambda \in (0,1)\}$ with $h(x) - h(y) = \nabla h(\xi)^T(x - y)$. Together with the Cauchy-Schwarz inequality we obtain $|h(x) - h(y)| \leq \|\nabla h(\xi)\| \|x - y\|$. Since

$$\|\nabla h(\xi)\| = \sqrt{\sum_{i=1}^{n} \left(\frac{\partial}{\partial x_i} h(\xi) \right)^2} \leq \sqrt{\sum_{i=1}^{n} \frac{L^2}{n}} = L,$$

we derive $|h(x) - h(y)| \leq L\omega(\tilde{X})$ for all $x, y \in \tilde{X}$. Let $x \in \tilde{X}$ be arbitrarily chosen. Then, due to (3.4), it follows that the distance between v and $h(x)$ is

$$|h(x) - v| \leq |h(x) - h(\tilde{x})| + |h(\tilde{x}) - v| \leq L\omega(\tilde{X}) + \frac{\alpha}{2}\omega(\tilde{X})^2 \leq \frac{\varepsilon}{2}.$$

□

The constant L in the above lemma can be obtained by using techniques from interval arithmetic, e.g., bounds for the partial derivatives of h on a box X can be computed by evaluating the respective natural interval extensions.

As vector-valued functions are considered in this work, we use convex underestimators for every objective function separately.

We denote the vector-valued convex underestimator of the function $f\colon \mathbb{R}^n \to \mathbb{R}^m$ by $f_\alpha\colon \mathbb{R}^n \to \mathbb{R}^m,\ x \mapsto (f_{1,\alpha}(x), \ldots, f_{m,\alpha}(x))^T$, where $f_{j,\alpha}$ is a convex underestimator of $f_j,\ j = 1, \ldots, m$. It is convenient to use the same parameter α for the convex underestimators of each objective function for an easier notation. Lemma 3.14 can easily be generalized to the vector-valued case by considering each objective function on its own.

3.3 Benson's Outer Approximation Algorithm

By using convex underestimators, we formulate in our procedure convex multiobjective optimization problems to our original nonconvex multiobjective optimization problem. For solving convex multiobjective optimization problems, solution methods are known. One of these is Benson's outer approximation algorithm (Benson's algorithm), which was originally developed by H. P. Benson for multiobjective linear optimization problems, [Ben98a; Ben98b]. The aim of this algorithm is to find a representation of the nondominated set by constructing supporting hyperplanes. The nondominated set of a multiobjective linear optimization problem consists of vertices and faces of a polyhedral convex set. Therefore, it is possible to obtain a complete representation of this set. The algorithm is extensively studied in the literature: a dual variant was proposed in [ELS12], while in [ESS11], Benson's algorithm was generalized to multiobjective convex optimization problems. In this latter case, the procedure is only able to compute an approximation of the nondominated set, because the nondominated set is in general a curve. The authors of [HLR14] generalized the original algorithm to linear multiobjective optimization problems with more general ordering cones. In fact, these cones have to be solid, pointed and polyhedral. In [LRU14], this algorithm was further developed for convex multiobjective optimization problems, and in addition, its dual variant is presented. A version for special nonconvex multiobjective optimization problems, i. e., "convexlike" functions, can be found in [Sha17], which uses Wolfe duality.

Here, we briefly recall one version of the algorithm for convex multiobjective optimization problems with the natural ordering cone $\mathbb{R}^m_+$. Therefore, we mainly follow the presentation in [ESS11; LRU14].

Consider the convex multiobjective optimization problem

$$\begin{aligned} \min_{x \in X} \quad & f(x) = (f_1(x), \ldots, f_m(x))^T \\ \text{s. t.} \quad & g(x) \leq 0, \end{aligned} \tag{$MOCP$}$$

where all functions $f_1, \ldots, f_m,\ g_1, \ldots, g_p$ are continuously differentiable, convex functions and $X \in \mathbb{IR}^n$ is a box. The feasible set of ($MOCP$) is denoted by

$$M := \{x \in X \mid g(x) \leq 0\}.$$

It is a convex set and we assume it to be non-empty. We define the *upper image* of ($MOCP$) by

$$\mathcal{P} := \{y \in \mathbb{R}^m \mid f(x) \leq y \text{ for one } x \in \mathbb{R}^n \text{ with } g(x) \leq 0\} = f(M) + \mathbb{R}^m_+.$$

Benson's algorithm aims at finding an outer approximation of the set $\mathcal{P}$ which is indeed an approximation of the (weakly) nondominated set of ($MOCP$), [Ben98a; ESS11]. This approximation is computed by using *supporting hyperplanes* of $\mathcal{P}$.

Definition 3.15 [Roc70, Section 11] Let $\mathcal{P} \subset \mathbb{R}^m$ be a nonempty set, let $\lambda \in \mathbb{R}^m \setminus \{0\}$ and $z \in \partial\mathcal{P}$, where $\partial\mathcal{P}$ is the boundary of the set $\mathcal{P}$. The hyperplane

$$H^{\lambda,z} := \{y \in \mathbb{R}^m \mid \lambda^T y = \lambda^T z\}$$

is called *supporting hyperplane* (of $\mathcal{P}$), if $\lambda^T y \geq \lambda^T z$ holds for all $y \in \mathcal{P}$.

The algorithm begins with the calculation of the so-called ideal point of f on M. For the sake of completeness, the following definition states the definition of the ideal point and the anti-ideal point.

Definition 3.16 [Ehr05, Def. 2.22] Let a function $f\colon \mathbb{R}^n \to \mathbb{R}^m$ and a set $M \in \mathbb{R}^n$ be given.

(i) The *ideal point* $a = (a_1, \ldots, a_m)^T$ of f on M consists of the global minimal values of every $f_j,\ j = 1, \ldots, m$ on M, i. e.,

$$a_j := \min_{x \in M} f_j(x) \text{ for all } j = 1, \ldots, m. \tag{3.5}$$

(ii) The *anti-ideal point* $z = (z_1, \dots, z_m)^T$ of f on M consists of the global maximal values of every $f_j,\ j = 1, \dots, m$ on M, i. e.,

$$z_j := \max_{x \in M} f_j(x) \text{ for all } j = 1, \dots, m. \tag{3.6}$$

Remark 3.17 The anti-ideal point of a function f multiplied by -1 is also the ideal point of $-f$. We distinguish the anti-ideal point from the *nadir point* y^N, which is defined as the componentwise maximum of the nondominated points:

$$y_j^N = \max_{x \in X_E} f_j(x) \text{ for all } j = 1, \dots, m.$$

The ideal and the anti-ideal point exist if the objective functions $f_1, \dots, f_m$ are continuous and the feasible set M is compact. For the remainder of this chapter, we need only the ideal point. The global minima required for the ideal point can be obtained by local optimization methods because we assumed the objective functions to be convex. Thus, the locally minimal solutions and values are globally minimal as well.

Let u^j be the j-th unit vector in $\mathbb{R}^m$. The ideal point a and the m unit vectors form the initial outer approximation of $\mathcal{P}$. This initial outer approximation consists of the m hyperplanes defined by

$$H^{u^j, a} := \{y \in \mathbb{R}^m \mid (u^j)^T a = (u^j)^T y\},\ j = 1, \dots, m.$$

The two possibilities of a representation of a polyhedron are called V-representation (vertices and directions) or H-representation (hyperplanes/half spaces) in the literature [LW17]. It is possible to obtain one representation using the other one and vice versa. The procedure to obtain the V-representation from the H-representation is called *vertex enumeration*, see e. g. [BFM98] for more information.

In every iteration of Benson's algorithm, a vertex $\bar{p}$ of the current outer approximation is chosen. Then the following optimization problem is solved:

$$\begin{aligned} \min_{(x,t) \in \mathbb{R}^{n+1}} \quad & t \\ \text{s. t.} \quad & \bar{p} + te \geq f(x), \\ & x \in M. \end{aligned} \tag{$P_{\bar{p}, M}$}$$

Recall that e is the m-dimensional all-ones vector. Actually, e can be replaced by any vector which belongs to the ordering cone $\mathbb{R}^m_+$. As we adapted Benson's algorithm for the algorithm presented in Chapter 4, and in order to prove some convergence results, we use the vector e. The optimization problem $(P_{\bar{p},\,M})$ is related to a minimization of the well-known Tammer-Weidner functional, see [GW90].

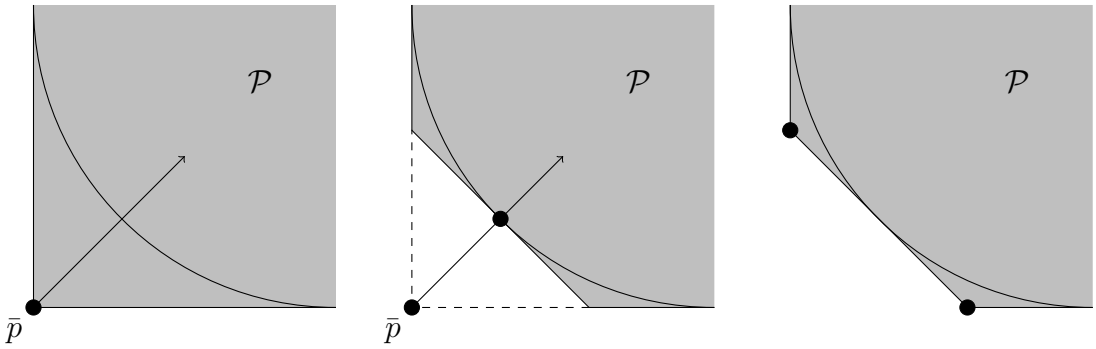
Depending on the minimal value $\tilde{t}$ of $(P_{\bar{p},\,M})$ and a given scalar $\varepsilon > 0$ the algorithm continues:

- $\tilde{t} < \varepsilon$: The point $\bar{p}$ lies close enough to the boundary of $\mathcal{P}$ and is added to the final outer approximation. In particular, if $\tilde{t} = 0$ holds, $\bar{p}$ belongs to the boundary of $\mathcal{P}$. The case of $\tilde{t} < 0$ does not appear in Benson's algorithm if $\bar{p}$ is a vertex of an outer approximation of $\mathcal{P}$.
- $\tilde{t} \geq \varepsilon$: The point $\bar{p}$ lies too far away from the boundary of $\mathcal{P}$ and a supporting hyperplane of $\mathcal{P}$ is determined.

In order to determine the hyperplane different methods are known. In early works, a further single objective optimization problem was solved. However, it is also possible to use a Lagrange multiplier $\tilde{\lambda}$ (if present) to the constraint $\bar{p} + te \geq f(x)$ of $(P_{\bar{p},\,M})$. This fact is stated and proved in the forthcoming Theorem 3.18. The new hyperplane is added to the H-representation of an outer approximation of $\mathcal{P}$. Via vertex enumeration the new vertices of the V-representation are calculated and another vertex, which was not considered yet, can be chosen. If for all computed vertices $\bar{p}$ the minimal value $\tilde{t}$ of $(P_{\bar{p},\,M})$ is smaller than ε, the algorithm stops. The output is the V- or the H-representation of an outer approximation of $\mathcal{P}$.

Figure 3.1 illustrates Benson's outer approximation algorithm by presenting different iterations and the overall result after three iterations.

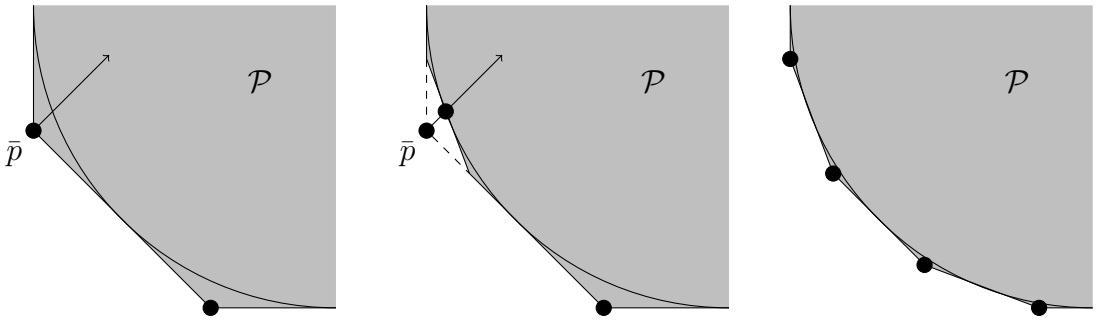
Determining Supporting Hyperplanes The next theorem proves that a Lagrange multiplier to the constraint $\bar{p} + te \geq f(x)$ of $(P_{\bar{p},\,M})$ can be used as a normal vector of a supporting hyperplane of the set $\mathcal{P}$ at $\bar{p} + \tilde{t}e$. Even while this fact is well-known because of duality properties, we show it for the sake of completeness.



a: First iteration: Ideal point is chosen as the first point $\bar{p}$ in order to solve $(P_{\bar{p},\,M})$.

b: First iteration: $(P_{\bar{p},\,M})$ is solved to get a point at the boundary of $\mathcal{P}$ and a supporting hyperplane.

c: First iteration: Via vertex enumeration two new vertices of the outer approximation are computed.



d: Second iteration: One vertex is chosen as $\bar{p}$ in order to solve $(P_{\bar{p},\,M})$.

e: Second iteration: $(P_{\bar{p},\,M})$ is solved to get a point at the boundary of $\mathcal{P}$ and a supporting hyperplane.

f: Outer approximation after three iterations.

Figure 3.1. Visualization of Benson's outer approximation algorithm.

The optimization problem $(P_{\bar{p},\,M})$ can be equivalently written as:

$$\begin{aligned} \min_{(x,t)\in\mathbb{R}^{n+1}} \quad & t \\ \text{s. t.} \quad & f(x) - \bar{p} - te \leq 0, \\ & \tilde{g}(x) \leq 0. \end{aligned} \tag{$P_{\bar{p},\,M}$}$$

The function $\tilde{g}\colon \mathbb{R}^n \to \mathbb{R}^{p+2n}$ is defined by the convex constraints $g_k,\ k = 1, \ldots, p$ and the box constraints for $x \in X = [\underline{x}, \overline{x}]$, i. e.,

$$\tilde{g}(x) := \begin{pmatrix} g(x) \\ \underline{x} - x \\ x - \overline{x} \end{pmatrix}.$$

Theorem 3.18 *Let*

$$L^1(x, t, \lambda, \mu) = t + \sum_{j=1}^{m} \lambda_j (f_j(x) - \bar{p}_j - t) + \sum_{k=1}^{p+2n} \mu_k \tilde{g}_k(x)$$

be the Lagrange function for $(P_{\bar{p},\,M})$ and let $(\tilde{x}, \tilde{t}, \tilde{\lambda}, \tilde{\mu}) \in \mathbb{R}^{n+1+m+p+2n}$ be a KKT point of $(P_{\bar{p},\,M})$. Then the set $\{y \in \mathbb{R}^m \mid \tilde{\lambda}^T y = \tilde{\lambda}^T(\bar{p} + \tilde{t}e)\}$ is a supporting hyperplane for $\mathcal{P} = f(M) + \mathbb{R}^m_+$ at $\bar{p} + \tilde{t}e$.

Proof. As $(\tilde{x}, \tilde{t}, \tilde{\lambda}, \tilde{\mu})$ is a KKT point of $(P_{\bar{p},\,M})$ the following conditions are satisfied:

$$\nabla_{x,t} L^1(\tilde{x}, \tilde{t}, \tilde{\lambda}, \tilde{\mu}) = 0, \tag{3.7}$$

$$f(\tilde{x}) - \bar{p} - \tilde{t}e \leq 0, \tag{3.8}$$

$$\tilde{g}(x) \leq 0, \tag{3.9}$$

$$\tilde{\lambda} \geq 0, \tag{3.10}$$

$$\tilde{\mu} \geq 0, \tag{3.11}$$

$$\tilde{\lambda}^T(f(\tilde{x}) - \bar{p} - \tilde{t}e) + \tilde{\mu}^T \tilde{g}(\tilde{x}) = 0. \tag{3.12}$$

Statement (3.7) is equivalent to

$$\sum_{j=1}^{m} \tilde{\lambda}_j \frac{\partial f_j(\tilde{x})}{\partial x_i} + \sum_{k=1}^{p+2n} \tilde{\mu}_k \frac{\partial \tilde{g}_k(\tilde{x})}{\partial x_i} = 0 \text{ for all } i = 1, \ldots, n \text{ and } \sum_{j=1}^{m} \tilde{\lambda}_j = 1. \tag{3.7'}$$

Moreover, from those conditions one can easily obtain

$$\tilde{\lambda}^T (f(\tilde{x}) - \bar{p} - \tilde{t}e) = 0, \text{ and } \tilde{\mu}^T \tilde{g}(\tilde{x}) = 0. \tag{3.12'}$$

Now consider another convex single objective optimization problem

$$\min_{x \in \mathbb{R}^n} \tilde{\lambda}^T f(x) \text{ s. t. } \tilde{g}(x) \leq 0 \tag{$P_{\tilde{\lambda}}$}$$

and its Lagrange function

$$L^2(x, \sigma) = \tilde{\lambda}^T f(x) + \sum_{k=1}^{p+2n} \sigma_k \tilde{g}_k(x).$$

Choosing $\sigma_k = \tilde{\mu}_k$ for every $k = 1, \ldots, p + 2n$, we obtain for $(\tilde{x}, \sigma)$ by the conditions (3.7'), (3.9), (3.11) and (3.12')

$$\nabla_x L^2(x, \sigma) = 0, \tag{3.13}$$
$$\tilde{g}(x) \leq 0, \tag{3.14}$$
$$\sigma \geq 0, \tag{3.15}$$
$$\sigma^T \tilde{g}(x) = 0. \tag{3.16}$$

Thus, the point $(\tilde{x}, \sigma)$ is a KKT point of $(P_{\tilde{\lambda}})$. Since $(P_{\tilde{\lambda}})$ is a convex optimization problem, the KKT point $(\tilde{x}, \sigma)$ is a minimal solution of $(P_{\tilde{\lambda}})$. Therefore,

$$\begin{aligned}
&\tilde{\lambda}^T f(\tilde{x}) \leq \tilde{\lambda}^T f(x) && \text{for all } x \in M \\
\Leftrightarrow\ &\tilde{\lambda}^T f(\tilde{x}) - \tilde{\lambda}^T (\bar{p} + \tilde{t}e) \leq \tilde{\lambda}^T f(x) - \tilde{\lambda}^T (\bar{p} + \tilde{t}e) && \text{for all } x \in M \\
\Leftrightarrow\ &\tilde{\lambda}^T (f(\tilde{x}) - \bar{p} - \tilde{t}e) \leq \tilde{\lambda}^T (f(x) - \bar{p} - \tilde{t}e) && \text{for all } x \in M
\end{aligned}$$

As $\tilde{\lambda}$ is a Lagrange multiplier, it follows with (3.12')

$$0 = \tilde{\lambda}^T(f(\tilde{x}) - \bar{p} - \tilde{t}e) \leq \tilde{\lambda}^T(f(x) - \bar{p} - \tilde{t}e) \text{ for all } x \in M.$$

From this and as $\tilde{\lambda} \geq 0$ holds, we obtain

$$\begin{aligned}
0 \leq \tilde{\lambda}^T(f(x) - \bar{p} - \tilde{t}e) \qquad & \text{for all } x \in M \\
\Leftrightarrow \tilde{\lambda}^T(\bar{p} + \tilde{t}e) \leq \tilde{\lambda}^T f(x) \qquad & \text{for all } x \in M \\
\Leftrightarrow \tilde{\lambda}^T(\bar{p} + \tilde{t}e) \leq \tilde{\lambda}^T(f(x) + c) \qquad & \text{for all } x \in M, c \in \mathbb{R}^m_+ \\
\Leftrightarrow \tilde{\lambda}^T(\bar{p} + \tilde{t}e) \leq \tilde{\lambda}^T y \qquad & \text{for all } y \in f(M) + \mathbb{R}^m_+. \qquad (3.17)
\end{aligned}$$

By (3.8) we have $\bar{p} + \tilde{t}e = f(\tilde{x}) + c$ for some $c \in \mathbb{R}^m_+$. Thus, $\bar{p} + \tilde{t}e \in f(M) + \mathbb{R}^m_+$ holds. Moreover, because of (3.17), the point $\bar{p} + \tilde{t}e$ is on the boundary of $\mathcal{P}$. Finally with $\tilde{\lambda} \neq 0$, see (3.7'), we conclude that the set $\{y \in \mathbb{R}^m \mid \tilde{\lambda}^T y = \tilde{\lambda}^T(\bar{p} + \tilde{t}e)\}$ is a supporting hyperplane of $f(M) + \mathbb{R}^m_+$ at the point $\bar{p} + \tilde{t}e$. □

The next remarks give some notes about the relation of KKT-points to minimal solutions of $(P_{\bar{p},\,M})$ as well as their existence.

Remark 3.19 If we assume additionally some regularity for $(P_{\bar{p},\,M})$, the assumptions of this theorem hold for every minimal solution $(\tilde{x}, \tilde{t})$ of $(P_{\bar{p},\,M})$. Especially, if we assume that M is a box with $\operatorname{int}(M) \neq \emptyset$, the Slater condition is satisfied for $(P_{\bar{p},\,M})$. We just have to chose an $\hat{x} \in \operatorname{int}(M)$ and $\hat{t}$ large enough such that $\bar{p} + \hat{t}e > f(\hat{x})$ holds to prove this condition.

Remark 3.20 Theoretically, the convex optimization problem $(P_{\bar{p},\,M})$ can be replaced by the following slightly different optimization problem

$$\begin{aligned}
\min_{(x,t)\in\mathbb{R}^{n+1}} \quad & t \\
\text{s. t.} \quad & \bar{p} + te \geq f(x), \\
& x \in M, \\
& t \in M^t := \left[\min_{j=1,\dots,m} \min_{x\in M} f_j(x) - \bar{p}_j,\ \max_{j=1,\dots,m} \max_{x\in M} f_j(x) - \bar{p}_j \right].
\end{aligned} \qquad (P'_{\bar{p},\,M})$$

The feasible set concerning x of both problems is bounded because of the box constraints described by X. The two bounds of M^t exist because all f_j are continuous on the compact set M. The lower bound of M^t is a lower bound of the minimal value of $(P_{\bar{p},\,M})$, because all t smaller than $\min_{j=1,\dots,m} \min_{x\in M} f_j(x) - \bar{p}_j$ cannot be feasible for $(P_{\bar{p},\,M})$. The upper bound $\max_{j=1,\dots,m} \max_{x\in M} f_j(x) - \bar{p}_j$ itself gives a feasible value for t with an arbitrary $x \in M$. Therefore, we can reduce the feasible set of $(P_{\bar{p},\,M})$ concerning t to the interval M^t. As $(P_{\bar{p},\,M})$ and $(P'_{\bar{p},\,M})$ aim for the minimization of t on $M \times \mathbb{R}$ or $M \times M^t$, respectively, $(P_{\bar{p},\,M})$ and $(P'_{\bar{p},\,M})$ have the same minimal solutions and minimal values. For problem $(P'_{\bar{p},\,M})$ a minimal solution exists, because of the continuous objective function and the compact feasible set. Thus, a minimal solution of $(P_{\bar{p},\,M})$ exists as well.

3.4 Local Upper Bounds

For B&B methods upper bounds are important. Within this context we will also need so called *local upper bounds*. The concept of local upper bounds is a versatile tool mainly used in multiobjective combinatorial optimization. Let a finite and stable set of function values $\mathcal{N} \subseteq f(M)$ be given. Recall that for a stable set $\mathcal{N}$ and points $y^1, y^2 \in \mathcal{N}$ either $y^1 \not\leq y^2$ or $y^1 = y^2$ holds. Let $\hat{Z}$ be a box with $f(M) \subseteq \operatorname{int}(\hat{Z})$. The *search region* S is the set which contains all points which are not dominated by $\mathcal{N}$, i. e.,

$$S := \{z \in \operatorname{int}(\hat{Z}) \mid q \not\leq z \text{ for all } q \in \mathcal{N}\} = \operatorname{int}(\hat{Z}) \setminus \left(\bigcup_{q\in\mathcal{N}} \{q\} + \mathbb{R}^m_+ \right). \tag{3.18}$$

This set can be characterized with the help of local upper bounds, which are the elements of the local upper bound set as defined below.

Definition 3.21 [KLV15] Let $\mathcal{N}$ be a finite and stable set and $\hat{Z}$ be a box with $\mathcal{N} \subseteq \operatorname{int}(\hat{Z})$. A list $\mathcal{L} \subseteq \hat{Z}$ is called a *local upper bound set w. r. t.* $\mathcal{N}$ if

(i) $\forall z \in S\ \exists p \in \mathcal{L} : z < p$,
(ii) $\forall z \in \left(\operatorname{int}(\hat{Z})\right) \setminus S\ \forall p \in \mathcal{L} : z \not< p$ and
(iii) $\forall p^1, p^2 \in \mathcal{L} : p^1 \not\leq p^2$ or $p^1 = p^2$.

The next propositions give some equivalent characterizations.

Proposition 3.22 *[KLV15] Let $\mathcal{N}$ be a finite and stable set.*

(i) The search zone for some $p \in \mathbb{R}^m$ related to the box $\hat{Z}$ is defined as

$$C(p) = \{w \in \text{int}(\hat{Z}) \mid w < p\}.$$

(ii) A list $\mathcal{L} \subseteq \hat{Z}$ is called a local upper bound set w. r. t. $\mathcal{N}$, if

(a) $S = \bigcup_{p \in \mathcal{L}} C(p)$,

(b) $C(p)$ is not a subset of $C(\tilde{p})$ for all $p, \tilde{p} \in \mathcal{L}$.

Proposition 3.23 *[KLV15] A set $\mathcal{L}$ is called a local upper bound set w. r. t. a finite and stable set $\mathcal{N}$ if and only if $\mathcal{L}$ consists of all points $p \in \hat{Z}$ that satisfy the following two conditions:*

(i) no point of $\mathcal{N}$ strictly dominates p and

(ii) for any $z \in \hat{Z}$ such that $z \geq p,\ z \neq p$, there exists $\bar{z} \in \mathcal{N}$ such that $\bar{z} < z$, i. e., p is a maximal point with property (i).

The following lemma is useful for understanding further relations between $\mathcal{N}$ and $\mathcal{L}$. It is due to [Kla17; NE19].

Lemma 3.24 *Let $\mathcal{L}$ be a local upper bound set w. r. t. a finite and stable set $\mathcal{N}$. For every $\bar{z} \in \mathcal{N}$ and for every $j \in \{1, \ldots, m\}$ there is a local upper bound $p \in \mathcal{L}$ with $\bar{z}_j = p_j$ and $\bar{z}_r < p_r$ for all $r \in \{1, \ldots, m\} \setminus \{j\}$.*

Proof. Let $\bar{z} \in \mathcal{N} \subseteq \text{int}(\hat{Z})$. As $\mathcal{N}$ is a finite set, there exists a neighborhood of $\bar{z}$ such that no other point of $\mathcal{N}$ is contained in that neighborhood. Hence, choose $\nu > 0$ such that

$$N_\nu(\bar{z}) := \{z \in \mathbb{R}^m \mid \|\bar{z} - z\| \leq \nu\} \subseteq \text{int}(\hat{Z})$$

and with $N_\nu(\bar{z}) \cap \mathcal{N} = \{\bar{z}\}$. Then we obtain for ν small enough

$$N_\nu(\bar{z}) \cap S = N_\nu(\bar{z}) \setminus (\{\bar{z}\} + \mathbb{R}^m_+).$$

Now, fix a $j \in \{1, \dots, m\}$ and let $(\delta_t)_{t \in \mathbb{N}}$ be a null sequence with $\nu > \delta_t > 0$ for all $t \in \mathbb{N}$. Then we consider the sequence $(q^t)_{t \in \mathbb{N}}$ defined componentwise by $q_j^t = \bar{z}_j - \delta_t$ and $q_r^t = \bar{z}_r$ for all $r \in \{1, \dots, m\} \setminus \{j\}$ and for all $t \in \mathbb{N}$. It holds that $\lim_{t \to \infty} q^t = \bar{z}$ and $q^t \in N_\nu(\bar{z}) \cap S$ for all $t \in \mathbb{N}$. Hence, with Definition 3.21 (i), we conclude that there is a local upper bound $p^t \in \mathcal{L}$ with $q^t < p^t$ for every $t \in \mathbb{N}$. Since $\mathcal{L}$ is a finite set, additionally the sequence $(p^t)_{t \in \mathbb{N}}$ contains a constant subsequence with a (limit) value $\bar{p}^j \in \mathcal{L}$. For its limit value it holds that $\bar{z} \leq \bar{p}^j$. Moreover, for all $t \in \mathbb{N}$ of the constant subsequence we have $\bar{z}_r = q_r^t < p_r^t = \bar{p}_r^j$ for all $r \in \{1, \dots, m\} \setminus \{j\}$. By Proposition 3.23 (i) it follows that $\bar{z}_j = \bar{p}_j^j$. □

As a result of Lemma 3.24, for every $\bar{z} \in \mathcal{N}$ there exists a local upper bound $p \in \mathcal{L}$ with $\bar{z} \leq p$.

Figure 3.2 illustrates the sets $\mathcal{N}$ and $\mathcal{L}$ for $m = 2$. The boundary of the box $\hat{Z}$ is drawn in gray. The search region is the lower left white subset of $\text{int}(\hat{Z})$.

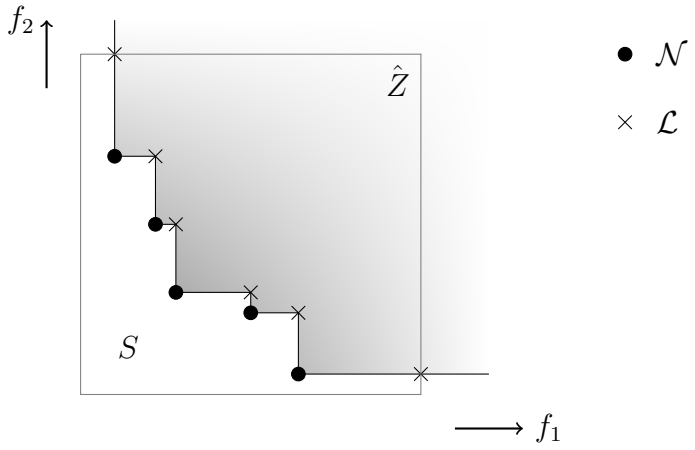


Figure 3.2. Stable set $\mathcal{N}$ and local upper bound set $\mathcal{L}$, $m = 2$, cf. [KLV15, Fig. 1]

In case of $m = 2$, the calculation of the local upper bounds is very simple if the list $\mathcal{N} = \{q^1, \dots, q^k\}$ is sorted in ascending order according to one component. Assume $\mathcal{N}$ is sorted w. r. t. the first component. Then every local upper bound has the form $p^s = (q_1^s, q_2^{s-1})^T$ for $s = 2, \dots, r$. We define $p^1 = (q_1^1, M_2)^T$ and $p^{r+1} = (M_1, q_2^r)^T$, where $(M_1, M_2)^T = \sup(\hat{Z})$.

For higher dimensions, various algorithms can be found in [KLV15] to calculate the set $\mathcal{L}$ from $\mathcal{N}$. Here, we present one of those algorithms, see Algorithm 1. All algorithms proposed in [KLV15] require an already existing local upper bound set $\mathcal{L}$ w. r. t. a set $\mathcal{N}$ and a point $\bar{q}$, which is a new point for $\mathcal{N}$, i. e., $\bar{q}$ is not dominated by any point of $\mathcal{N}$. The update procedure of $\mathcal{N} \cup \{\bar{q}\}$ to a stable set $\mathcal{N}'$ should be done separately. Then the algorithms compute a new upper bound set $\mathcal{L}'$ w. r. t. $\mathcal{N}'$. In the algorithms of this thesis the set $\mathcal{N}$ grows incrementally. Thus, we can directly use the algorithms proposed in [KLV15] every time a new point is added to $\mathcal{N}$.

The initial upper bound set, i. e., for $\mathcal{N} = \emptyset$, is $\mathcal{L} = \sup(\hat{Z})$. Algorithm 1 is based on so-called *redundancy elimination* and uses an enhanced filtering step compared to other procedures. For this, we need the following notations: For any $p \in \mathbb{R}^m$ and $j \in \{1, \dots, m\}$ the point p_{-j} denotes the $(m-1)$-dimensional vector

$$p_{-j} := (p_1, \dots, p_{j-1}, p_{j+1}, \dots, p_m)^T.$$

The vector (a_j, p_{-j}) is the vector $p \in \mathbb{R}^m$ with a j-th component replaced by the j-th component of $a \in \mathbb{R}^m$, i. e., $(a_j, p_{-j}) = (p_1, \dots, p_{j-1}, a_j, p_{j+1}, \dots, p_m)^T$.

Algorithm 1 Update procedure of an upper bound set, cf. [KLV15, Algorithm 3]

Input: $\mathcal{N}, \mathcal{L}$ w. r. t. $\mathcal{N}, \bar{q}$
Output: $\mathcal{L}'$ w. r. t. $\mathcal{N} \cup \{\bar{q}\}$

1: $A \leftarrow \{p \in \mathcal{L} \mid \bar{q} < p\}$
2: **for** $j \in \{1, \dots, m\}$ **do**
3: $\quad B_j \leftarrow \{p \in \mathcal{L} \mid p_j = \bar{q}_j \text{ and } \bar{q}_{-j} < p_{-j}\}$
4: $\quad P_j \leftarrow \emptyset$
5: **for** $p \in A$ **do**
6: $\quad$ **for** $j \in \{1, \dots, m\}$ **do**
7: $\quad\quad P_j \leftarrow P_j \cup \{(\bar{q}_j, p_{-j})\}$
8: **for** $j \in \{1, \dots, m\}$ **do**
9: $\quad P_j \leftarrow \{(\bar{q}_j, p_{-j}) \in P_j \mid (\bar{q}_j, p_{-j}) \not\leq p' \text{ or } (\bar{q}_j, p_{-j}) = p',\ \forall p' \in P_j \cup B_j\}$
10: $\mathcal{L}' \leftarrow (\mathcal{L} \setminus A) \cup \bigcup_{j=1}^{m} P_j$

We explain the procedure of Algorithm 1 by an example.

Example 3.25 [KLV15, Example 2, adapted] Let $m = 3$ and assume

$$\mathcal{N} = \{(3,5,7)^T, (6,5,4)^T\}$$

as well as $\hat{Z} = [(0,0,0)^T, (10,10,10)^T]$. As Algorithm 1 is an incremental algorithm, we start with $\mathcal{N} = \emptyset$, and the initial upper bound set is then $\mathcal{L} = \{(10,10,10)^T\}$. First, the point $(3,5,7)^T$ is added to $\mathcal{N}$. In line 1, A is set to $\{(10,10,10)^T\}$. As $(10,10,10)^T$ has no components with $(3,5,7)^T$ in common, all sets B_j get to be empty in line 3 as well as all P_j. In line 7, for every local upper bound from A we obtain new candidates by replacing the j-th component of the former local upper bound by the j-th component of the new point for $\mathcal{N}$. Thus, we get here $P_1 = \{(3,10,10)^T\}$, $P_2 = \{(10,5,10)^T\}$ and $P_3 = \{(10,10,7)^T\}$. These points are possible candidates for $\mathcal{L}$. However, it may happen that one of the local upper bounds is componentwise greater than a new one. In order to satisfy (iii) of Definition 3.21, only the local upper bounds are used which are not less than any other one. Here, all new upper bounds cannot be compared with each other. Thus, the new local upper bound set is

$$\mathcal{L} = \{(3,10,10)^T, (10,5,10)^T, (10,10,7)^T\}.$$

In the next step, the point $(6,5,4)^T$ is added to $\mathcal{N}$. Note that points are allowed to be only added to $\mathcal{N}$ if they are not dominated by any point of $\mathcal{N}$. The points of $\mathcal{N}$ dominated by the new point have to to be removed from $\mathcal{N}$. Since $(6,5,4)^T$ is neither dominated nor dominates a point of $\mathcal{N}$, it can be added to $\mathcal{N}$ easily. The only local upper bound which components are greater than the ones from $(6,5,4)^T$ is $(10,10,7)^T$. This one is added to A. Now in line 3, $B_2 = \{(10,5,10)^T\}$ holds since this local upper bound shares one component with $(6,5,4)^T$, and all others are larger. For the others, it holds $B_1 = B_3 = \emptyset$. Then in line 7, we get possible new local upper bounds by replacing each component from the ones of A separately by those from $(6,5,4)^T$: $P_1 = \{(6,10,7)^T\}, P_2 = \{(10,5,7)^T\}$ and $P_3 = \{(10,10,4)^T\}$. As B_1 and B_3 are empty and P_1 and P_3 are only singletons, nothing changes for P_1 and P_3 in line 9. For P_2 we are searching for redundant local upper bounds. As $(10,5,7)^T \leq (10,5,10)^T$ holds, the candidate $(10,5,7)^T$ is redundant and P_2 is set to $\emptyset$. The final list of local upper bounds is then $\mathcal{L} = \{(3,10,10)^T, (10,5,10)^T, (6,10,7)^T, (10,10,4)^T\}$.

As already mentioned, several algorithms to compute $\mathcal{L}$ from $\mathcal{N}$ exist. Next to Algorithm 1, which is Algorithm 3 in [KLV15], we also use Algorithm 5 of [KLV15], which is based on *avoidance of redundancies*. Both procedures work fine, but computational tests by [KLV15] reveal that Algorithm 1, i. e., Algorithm 3 of [KLV15], should be used for lower dimensions (up to $m = 5$). For higher dimensions Algorithm 5 of [KLV15] is preferable.

The following remark states a property of consecutive sets of $\mathcal{N}$ and $\mathcal{L}$ during an optimization algorithm or the procedure where $\mathcal{L}$ is determined incrementally.

Remark 3.26 Let $\mathcal{N}'$ and $\mathcal{N}$ be two consecutive stable sets, and let $\mathcal{L}'$ and $\mathcal{L}$ be the related local upper bound sets. Then for every $p \in \mathcal{L}$ there exists a local upper bound $p' \in \mathcal{L}'$ such that $p \leq p'$. This can be seen, for example, by induction considering the updating procedure in Algorithm 1.

3.5 A Basic Branch-and-Bound Method

All forthcoming algorithms of this thesis are B&B based algorithms. Looking at the abundance of publications considering and using B&B methods, it is clear that B&B is widely used in many application scopes. For example, [Adj+98; Aud+00; Dür01; HP09; KSS15; TH88] use B&B approaches for single objective optimization and [ACA19; Cam+18; FT09; Mar+16; Wan+08; ŽŽ16] for multiobjective optimization. In particular, the work of [Sch12] deals with B&B algorithms for both single and multiobjective optimization extensively.

The main idea of the B&B approach is to divide the feasible set or a super set of it iteratively into subsets. Usually, these sets are boxes because they can simply be bisected. The subsets or subboxes are further examined whether they contain any efficient points of the given optimization problem. In case of no efficient solutions this subbox does not have to be further bisected and can be discarded. A typical B&B algorithm is stated in Algorithm 2, see also, for example, [FT09].

The lists $\mathcal{L}_W$ and $\mathcal{L}_S$ are the working list and the solution list, respectively. The selection, bisection and termination rule as well as the discarding tests are specified for

Algorithm 2 Basic B&B algorithm

Input: multiobjective optimization problem with box constraints described by X

Output: $\mathcal{L}_S$

1: $\mathcal{L}_W \leftarrow \{X\},\ \mathcal{L}_S \leftarrow \emptyset$
2: **while** $\mathcal{L}_W \neq \emptyset$ **do**
3: Select a box X^* from $\mathcal{L}_W$ and delete it from $\mathcal{L}_W$ *Selection rule*
4: Bisect X^* into subboxes X^1, X^2 *Bisection rule*
5: **for** $l = 1, 2$ **do**
6: **if** X^l cannot be discarded **then** *Discarding tests*
7: **if** X^l satisfies a termination rule **then** *Termination rule*
8: Store X^l in $\mathcal{L}_S$
9: **else** Store X^l in $\mathcal{L}_W$

every B&B algorithm separately. The rules for selection, bisection and termination are usually heuristically motivated. Instead, a discarding test has to ensure that no box with efficient points gets eliminated. For each algorithm of the forthcoming chapters, we examine those rules and tests individually.

4 A Global Solution Method for Multiobjective Nonconvex Optimization

This chapter proposes an algorithm for solving multiobjective nonconvex optimization problems globally, i. e., the objective functions $f_1, \ldots, f_m \colon \mathbb{R}^n \to \mathbb{R}$ do not have to be convex functions. The considered multiobjective nonconvex optimization problem is (MOP) with inequality constraints described by $g_1, \ldots, g_p \colon \mathbb{R}^n \to \mathbb{R}$.

$$\begin{aligned} \min_{x \in X} \quad & f(x) = \begin{pmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{pmatrix} \\ \text{s. t.} \quad & g_k(x) \leq 0 \text{ for all } k = 1, \ldots, p. \end{aligned} \tag{MOP}$$

We assume the following properties for (MOP):

- f_j is a twice continuously differentiable function for all $j = 1, \ldots, m$,
- f_j is a factorable function for all $j = 1, \ldots, m$, see Definition 3.2,
- g_k is a continuously differentiable and convex function for all $k = 1, \ldots, p$,
- X is an n-dimensional box, see Section 3.1.

The feasible set of (MOP) is thus defined by

$$M := \{x \in X \mid g_k(x) \leq 0 \text{ for all } k = 1, \ldots, p\}.$$

Analogously to [FT09], the new algorithm is based on a B&B approach. However, the algorithm is suitable for an arbitrary number of objective functions and we provide a new discarding test procedure based on the concept of convex underestimators from

the αBB method, [MF94]. This results in convex multiobjective optimization problems which have to be considered on each subbox. We combine approaches from multiobjective convex optimization with the concept of the local upper bounds in order to obtain improved lower bounds. For an introduction to local upper bounds, see Section 3.4. Finally, we are able to prove that the new algorithm finds an approximation of the set of globally optimal solutions for multiobjective optimization problems with predefined quality in finite time.

Section 4.1 deals with a new discarding test for the B&B algorithm and presents a termination procedure which is able to guarantee (ε, δ)-efficiency of the output set. Then the used rules for the selection and bisection steps, which are already well-known in the literature, are introduced in Section 4.2. The complete algorithm is stated in Section 4.3. Later on, in Section 4.4, we prove the termination and correctness of the proposed algorithm. The algorithm and the proofs are also published our article [NE19].

At the end of this chapter, we discuss the relation of the new algorithm to other global solution methods for multiobjective optimization problems. These algorithms will be shortly introduced in this part.

4.1 Discarding Test and Termination Procedure

First, we state how lower and upper bounds for the B&B algorithm are computed. Afterwards, the complete discarding test is described. Since the termination procedure is deeply related to the discarding test, this procedure is described in this section as well.

4.1.1 Computing Lower and Upper Bounds

We generate a stable set $\mathcal{L}_{PNS}$ of objective values (called the *provisional nondominated set*) representing upper bounds for the global nondominated set for (MOP). For this, we need the images of some feasible points of (MOP). Each point q as a new candidate for $\mathcal{L}_{PNS}$ is checked if it is dominated by any other point of $\mathcal{L}_{PNS}$. In this case, q will

not be included in $\mathcal{L}_{PNS}$. Otherwise, q will be added to $\mathcal{L}_{PNS}$ and all points dominated by q will be removed. The procedure is described in Algorithm 3 as well.

Algorithm 3 Updating procedure for $\mathcal{L}_{PNS}$

Input: finite list $\mathcal{L}_{PNS} \subseteq \mathbb{R}^m, q^* \in \mathbb{R}^m$

Output: updated list $\mathcal{L}_{PNS}$

1: **for** $q \in \mathcal{L}_{PNS}$ **do**
2: **if** $q \leq q^*$ **then break for-loop**
3: **else if** $q^* \leq q$ **then**
4: **if** $q^* \notin \mathcal{L}_{PNS}$ **then** Add q^* to $\mathcal{L}_{PNS}$ and remove q from $\mathcal{L}_{PNS}$
5: **else** Remove q from $\mathcal{L}_{PNS}$

Figure 4.1 shows an example for the provisional nondominated set $\mathcal{L}_{PNS}$ of a two-dimensional image set $f(M)$.

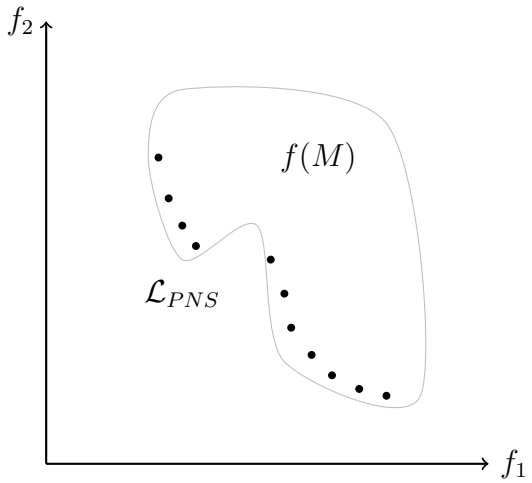


Figure 4.1. Example for $\mathcal{L}_{PNS}$, $m = 2$.

Having this list of upper bounds, a discarding test for a given box X^* also requires a lower bound for the values of f over

$$M^* := X^* \cap M = \{x \in X^* \mid g(x) \leq 0\}.$$

Let $LB \subseteq \mathbb{R}^m$ be a set with $f(M^*) \subseteq LB + \mathbb{R}^m_+$. If $LB \subseteq (\mathcal{L}_{PNS} + \mathbb{R}^m_+) \setminus \mathcal{L}_{PNS}$, the box X^* can be discarded, because every point of $f(M^*)$ is in $(\mathcal{L}_{PNS} + \mathbb{R}^m_+) \setminus \mathcal{L}_{PNS}$. Hence, every point of $f(M^*)$ is dominated by one point of $\mathcal{L}_{PNS}$. For the set LB we

first recall the approach proposed so far in the literature, (I), and then present our new approach which consists of two steps (II) and (III):

(I) Proposed in [FT09]: LB is chosen as the lower vertex of a box which contains all values of f on X^* and which is generated by the natural interval extension F of f on X^*. Note that this approach cannot take the convex constraints into account.

(II) LB is chosen as the ideal point of the convex underestimators of the functions f_j over M^*, see Definition 3.16, and set $LB = \{a = (a_1, \ldots, a_m)\}$. For an illustration, see Figure 4.2.

(III) Find a tighter and not necessarily singleton set LB with $f(M^*) \subseteq LB + \mathbb{R}^m_+$ and with $(LB + \mathbb{R}^m_+) \setminus (f(M^*) + \mathbb{R}^m_+)$ as small as possible by using convex underestimators and techniques from multiobjective convex optimization. We illustrate and discuss this new discarding test in Section 4.1.2.

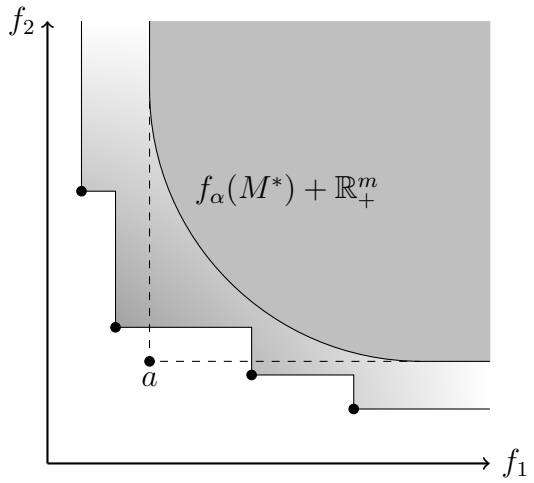


Figure 4.2. Upper image set $f_\alpha(M^*) + \mathbb{R}^m_+$ for $m = 2$, $\mathcal{L}_{PNS}$ and an ideal point a of f_α.

We start by briefly discussing the first step of our new approach, i. e., we show that a from (II) (see (3.5)) already delivers a lower bound:

Lemma 4.1 *Let $f_{j,\alpha}$ be a convex underestimator of f_j on X^* for $j = 1, \ldots, m$ and define the ideal point $a \in \mathbb{R}^m$ on M^* by* (3.5)*. Then $a \leq f(x)$ holds for all $x \in M^*$, i. e., $f(M^*) \subseteq \{a\} + \mathbb{R}^m_+$.*

Proof. Let $j \in \{1, \dots, m\}$. Because $f_{j,\alpha}$ is a convex underestimator of f_j on $X^* \supseteq M^*$ and from the definition of a_j it follows that $a_j \leq f_{j,\alpha}(x) \leq f_j(x)$ holds for all $x \in M^*$. □

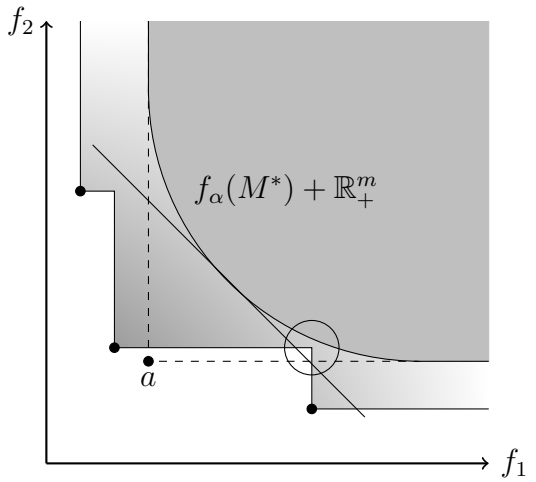
Numerical experiments show that using (I) or (II) makes no big difference if there are no convex contraints. In some cases the lower bounds by interval arithmetic are better than the ones by convex underestimators. In other cases, the converse holds. However, (II) has the advantage that the maximal error between the computed lower bound and actual function values is bounded (see Remark 3.13 and Lemma 3.14). Moreover, it is possible to improve (II) to (III) by using the convexity of f_α, which is not possible for (I). This is the basic idea of our new discarding test.

The tighter bounds will be reached by adding cuts as known from Benson's outer approximation algorithm for convex multiobjective optimization problems, see Section 3.3 or [Ben98a; ESS11]. The cuts separate selected points p from the upper image set of the convex underestimators such that the cuts are supporting hyperplanes, see Figures 4.3a and 4.3c. This leads to new sets LB as illustrated in Figures 4.3b and 4.3d. As we can see, the cut in Figure 4.3a would lead to a set LB which does not allow to discard the box. This is due to the white triangle, which is circled in Figure 4.3a. This triangle is not dominated by any point of $\mathcal{L}_{PNS}$ but it is in $LB + \mathbb{R}^2_+$. However, the cut in Figures 4.3c and 4.3d allows to discard the box.

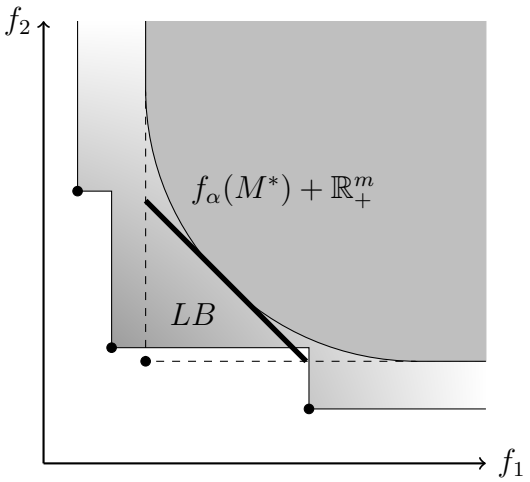
Here, the local upper bounds from Section 3.4 play an important role as these are exactly the points p which have to be separated from the upper image set. The set of upper bounds serves as the stable set $\mathcal{N}$. The local upper bound set is now further denoted by $\mathcal{L}_{LUB}$.

4.1.2 The Discarding Test Procedure

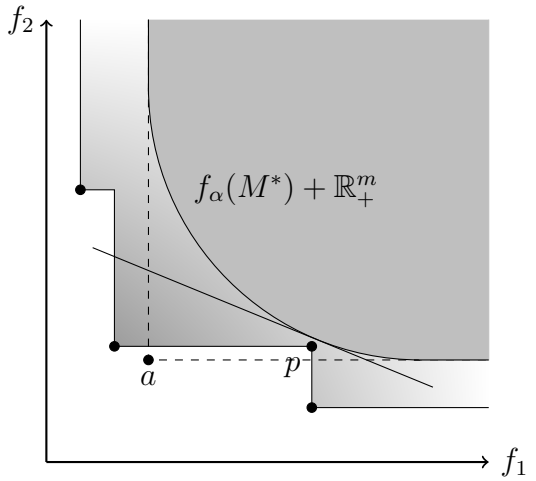
As visualized in Figure 4.3, the local upper bounds w.r.t. the set $\mathcal{L}_{PNS}$ are important as we can discard a box X^* if no local upper bound is contained in the set $f_\alpha(M^*) + \mathbb{R}^m_+$. Recall that M^* is the intersection between the box X^* and the feasible set M.



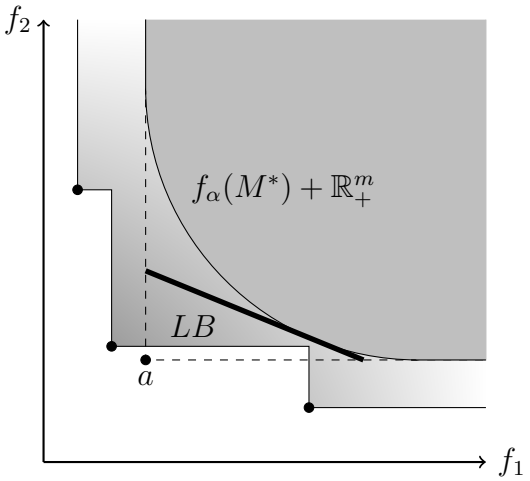
a: A cut which separates a from $f_\alpha(M^*) + \mathbb{R}^m_+$.



b: A lower bound LB for the situation in Figure 4.3a.



c: A cut which separates p from $f_\alpha(M^*) + \mathbb{R}^m_+$.



d: A lower bound LB for the situation in Figure 4.3c.

Figure 4.3. Possible cuts to obtain a tighter set LB with $f(M^*) \subseteq LB + \mathbb{R}^m_+$, $m = 2$.

Lemma 4.2 *Let a box $X^* \in \mathbb{IR}^n, X^* \subseteq X$ be given and let $f_{j,\alpha}$ be a convex underestimator of f_j on X^* for $j = 1, \dots, m$. Let $\mathcal{L}_{LUB} \subseteq \mathbb{R}^m$ be the local upper bound set w. r. t. $\mathcal{L}_{PNS}$. If*

$$\bar{p} \notin f_\alpha(M^*) + \mathbb{R}^m_+ \text{ holds for all } \bar{p} \in \mathcal{L}_{LUB}, \tag{4.1}$$

X^* does not contain any efficient point of (MOP).

Proof. Assume that there is some efficient point x^* of (MOP) with $x^* \in M^*$. Because $f_{j,\alpha}$ is a convex underestimator of f_j on X^* for all $j \in \{1, \dots, m\}$ no local upper bound $\bar{p} \in \mathcal{L}_{LUB}$ belongs to $f(M^*) + \mathbb{R}^m_+$. Thus, $f(x^*) \not\leq \bar{p}$ for all $\bar{p} \in \mathcal{L}_{LUB}$. From Definition 3.21 (i) it follows that $f(x^*)$ cannot be an element of the search region S. With (3.18), we conclude that there exists a point $q \in \mathcal{L}_{PNS}$ with $q \leq f(x^*)$. As q is the image of a feasible point of (MOP) and as x^* is efficient for (MOP), we obtain $f(x^*) = q \in \mathcal{L}_{PNS}$. By Lemma 3.24, a local upper bound $p \in \mathcal{L}_{LUB}$ exists with $f(x^*) \leq p$ or, equivalently, $p \in \{f(x^*)\} + \mathbb{R}^m_+$. Again, as $f_{j,\alpha}$ are convex underestimators of f_j on X^* for all $j \in \{1, \dots, m\}$, we conclude $p \in \{f_\alpha(x^*)\} + \mathbb{R}^m_+$ which is a contradiction to (4.1). Thus, M^* contains no efficient point. The points of $X^* \setminus M^*$ cannot be efficient for (MOP), because they are infeasible. □

Condition (4.1) can be tested by solving a convex single objective optimization problem for every local upper bound. Instead of solving such an optimization problem for each point $\bar{p} \in \mathcal{L}_{LUB}$, we solve it for a few points and immediately obtain the information on how to generate and improve an outer approximation of $f_\alpha(M^*) + \mathbb{R}^m_+$. This information can be used to efficiently reduce the number of points for which the single objective optimization problem has to be solved. This corresponds to generating tighter lower bounds for the values of f over a box. Thus, let $\bar{p} \in \mathcal{L}_{LUB}$ and let a box X^* be given. We check if $\bar{p} \in f_\alpha(M^*) + \mathbb{R}^m_+$ holds by solving the following convex single objective optimization problem:

$$\begin{aligned} \min_{(x,t) \in \mathbb{R}^{n+1}} \quad & t \\ \text{s. t.} \quad & \bar{p} + te \geq f_\alpha(x), \\ & g(x) \leq 0, \\ & x \in X^*. \end{aligned} \tag{$P_{\bar{p},\, X^*}$}$$

A minimal solution of $(P_{\bar{p},\,X^*})$ is named $(\tilde{x},\tilde{t})$. If $\tilde{t} \leq 0$, it holds that $\bar{p} \in f_\alpha(M^*) + \mathbb{R}^m_+$. Otherwise, if $\tilde{t} > 0$, the point $\bar{p}$ lies outside of $f_\alpha(M^*)+\mathbb{R}^m_+$ and can be separated from $f_\alpha(M^*) + \mathbb{R}^m_+$ with a supporting hyperplane. This is used for our new discarding test. The test is applied to a box X^* and consists of a finite number of iterations where an outer approximation of $f_\alpha(M^*) + \mathbb{R}^m_+$ is determined. The initial outer approximation is $\{a\} + \mathbb{R}^m_+$, where a is the ideal point of f_α on M^*, see (3.5). In each iteration a local upper bound $\bar{p}$ is chosen. The first step is the comparison of the current outer approximation of $f_\alpha(M^*) + \mathbb{R}^m_+$ with $\bar{p}$ by checking if the inequalities which describe the outer approximation are satisfied. In the case in which $\bar{p}$ is not an element of this outer approximation, the next iteration, namely choosing a next local upper bound, starts. Otherwise we continue with solving $(P_{\bar{p},\,X^*})$ to obtain the position of $\bar{p}$ w. r. t. $f_\alpha(M^*) + \mathbb{R}^m_+$. The abovementioned cases ($\tilde{t} > 0$, $\tilde{t} \leq 0$) can occur. We extend this to three cases to reduce the effort as only ε-efficiency is the aim for a given scalar $\varepsilon > 0$. Hence, we distinguish between the following cases for the minimum value $\tilde{t}$ of $(P_{\bar{p},\,X^*})$:

(1) $\tilde{t} \leq 0$, i. e., $\bar{p} \in f_\alpha(M^*) + \mathbb{R}^m_+$: Efficient points in X^* are possible. Thus, X^* cannot be discarded and we distinguish between the following two subcases:
   (1a) $\tilde{t} < -\frac{\varepsilon}{2}$: Stop the whole discarding test in order to bisect X^* later.
   (1b) $-\frac{\varepsilon}{2} \leq \tilde{t} \leq 0$: Construct a supporting hyperplane to improve the outer approximation of $f_\alpha(M^*) + \mathbb{R}^m_+$ and choose the next local upper bound. Moreover, set a flag that X^* cannot be discarded.

(2) $\tilde{t} > 0$, i. e., $\bar{p} \notin f_\alpha(M^*)+\mathbb{R}^m_+$: Construct a supporting hyperplane to improve the outer approximation of $f_\alpha(M^*) + \mathbb{R}^m_+$ and choose the next local upper bound.

Only if case (2) holds for every local upper bound, the box X^* can be discarded, see Lemma 4.2. The case (1b) is motivated by obtaining ε-efficient points at the end of the algorithm as we will prove later. To store X^* in the solution list, there has to be at least one $\bar{p}$ where case (1b) is fulfilled and no $\bar{p}$ for which (1a) holds.

During the discarding test new supporting hyperplanes are constructed if $\tilde{t} \geq -\frac{\varepsilon}{2}$ holds. The support vector of such a hyperplane is $\tilde{y} := \bar{p}+\tilde{t}e$. For calculating a normal vector $\lambda^* \in \mathbb{R}^m$ of the supporting hyperplane a procedure is given in [ESS11], where a linear single objective optimization problem has to be solved. Alternatively to this and with help of properties of duality theory, we can also use a Lagrange multiplier

$\lambda^* \in \mathbb{R}^m$ of the constraint $\bar{p} + te \geq f_\alpha(x)$ to obtain a normal vector of the supporting hyperplane, as explained in detail in Section 3.3. Algorithm 4 describes the procedure, where the flag $\mathcal{D}$ stands for the decision to discard a box after the algorithm and the flag $\mathcal{B}$ for bisecting the box, respectively. It can be applied to a box X^* if $M^* \neq \emptyset$ holds.

Algorithm 4 Discarding test

Input: $X^* \in \mathbb{IR}^n,\ f \in \mathcal{C}^2(\mathbb{R}^n, \mathbb{R}^m),\ g \in \mathcal{C}(\mathbb{R}^n, \mathbb{R}^p),\ \mathcal{L}_{PNS},\ \mathcal{L}_{LUB} \subseteq \mathbb{R}^m,\ \varepsilon > 0,\ \alpha$
Output: flags $\mathcal{D},\ \mathcal{B}$

1: Compute for every objective function its convex underestimator on X^* and its corresponding minimum x^j on M^*
2: Update $\mathcal{L}_{PNS}$ by $f(x^j)$ and $\mathcal{L}_{LUB} =: \{p^1, \ldots, p^r\}$ by Algorithms 1 and 3
3: $\mathcal{D} \leftarrow 1,\ \mathcal{B} \leftarrow 0$
4: **for** $s = 1, \ldots, r$ **do**
5: **if** p^s is inside the current outer approximation of $f_\alpha(M^*) + \mathbb{R}^m_+$ **then**
6: Solve (P_{p^s, X^*}) with minimal solution $(\tilde{x}, \tilde{t})$
7: **if** $\tilde{t} < -\frac{\varepsilon}{2}$ **then**
8: **break for-loop**
9: Set flags $\mathcal{D} \leftarrow 0$ and $\mathcal{B} \leftarrow 1$
10: **else if** $\tilde{t} \leq 0$ **then**
11: Update outer approximation of $f_\alpha(M^*) + \mathbb{R}^m_+$ and set flag $\mathcal{D} \leftarrow 0$
12: **else** Update outer approximation of $f_\alpha(M^*) + \mathbb{R}^m_+$

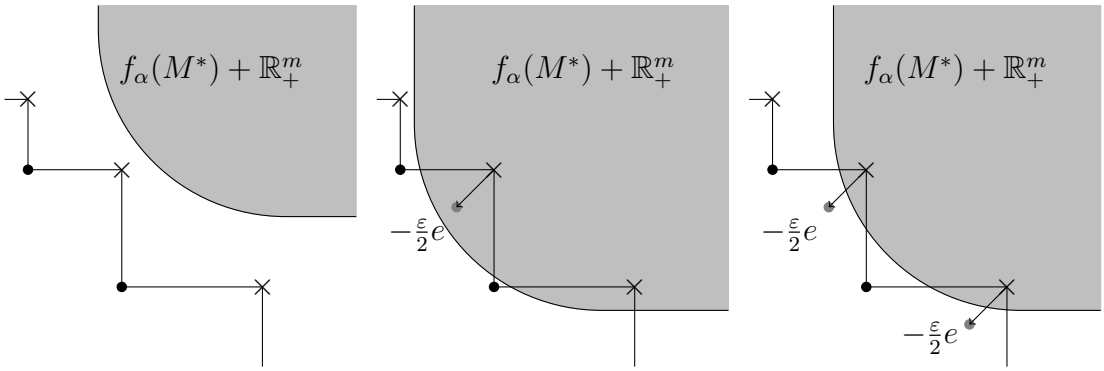
If the feasible set M^* is empty, we can discard the box. It might be difficult to verify numerically that the feasible set is indeed empty. A quickly evaluable and efficient sufficient condition was proposed in [FT09] and uses interval arithmetic to obtain lower and upper bounds of $g_k,\ k = 1, \ldots, p$. In case of a lower bound of g_k on X^* which is greater than 0 for at least one $k \in \{1, \ldots, p\}$, the whole box X^* is infeasible. If all upper bounds of g_k on a box X^* are less than 0 for all $k = 1, \ldots, p,\ X^* = M^*$ holds. Then the constraints can be neglected for X^* and all its subboxes. It is possible that none of the two cases occurs. Then the discarding test continues and takes the constraints into account while solving some single objective optimization problems.

Note that the condition of line 5 of Algorithm 4 can be checked by evaluating a finite number of inequalities which are given by the current outer approximation. The following theorem gives the correctness of this discarding test.

Theorem 4.3 *Let a box $X^* \subseteq X \in \mathbb{IR}^n$ and (MOP) be given. Let $\mathcal{L}_{LUB} \subseteq \mathbb{R}^m$ be a local upper bound set w. r. t. $\mathcal{L}_{PNS}$. If X^* contains an efficient point of (MOP), the output of Algorithm 4 is $\mathcal{D} = 0$, i. e., X^* will not be discarded by Algorithm 4.*

Proof. Assume that there is some efficient point x^* of (MOP) with $x^* \in M^*$. Suppose the output of Algorithm 4 applied to X^* is $\mathcal{D} = 1$. This means that for all local upper bounds either the conditions in lines 7 and 10 are not satisfied or they are not contained in the current outer approximation (see line 5). If the latter occurs, the local upper bound is clearly not in $f_\alpha(M^*) + \mathbb{R}^m_+$. For the other local upper bounds $\bar{p} \in \mathcal{L}_{LUB}$, which do not satisfy lines 7 and 10, but do satisfy line 5, the optimization problem $(P_{\bar{p},\, X^*})$ has a minimal value $\tilde{t} > 0$. Hence, $\bar{p} \notin f_\alpha(M^*) + \mathbb{R}^m_+$ holds for all $\bar{p} \in \mathcal{L}_{LUB}$ and with Lemma 4.2 we have a contradiction to the assumption that X^* contains an efficient point. □

All possibilities for the results of the discarding test applied to a box X^* in the two-dimensional case are illustrated in Figure 4.4.



a: Discard box X^*.

b: Do not discard box X^*, but bisect it.

c: Neither discard box X^*, nor bisect it.

Figure 4.4. Possible situations during a discarding test applied to box X^*.
• - $\mathcal{L}_{PNS}$; × - $\mathcal{L}_{LUB}$.

The list $\mathcal{L}_{PNS}$ is growing during the algorithm. Hence, we obtain better upper bounds for the nondominated set. It is possible that a box X^* was not discarded and not bisected by Algorithm 4, even though it could be discarded if we compared it with a later version of the list $\mathcal{L}_{PNS}$. Theoretically, after every update of $\mathcal{L}_{PNS}$ and $\mathcal{L}_{LUB}$, it should be checked whether these new lists can discard a box from the solution list. This requires the repetition of Algorithm 4 to every box of the solution list. As $\mathcal{L}_{PNS}$ and $\mathcal{L}_{LUB}$ can possibly change after every evaluation of Algorithm 4, this might be a very time-consuming and inefficient procedure. Thus, these two lists get constructed in a first `while`-loop which has the style like the loop in Algorithm 2. Then a new loop compares the boxes from the solution list with the fixed lists $\mathcal{L}_{PNS}$ and $\mathcal{L}_{LUB}$. To evaluate the comparisons, Algorithm 4 is applied again, but without the updating in line 2. This adapted method can be found in Algorithm 5. Additionally, in that procedure we reuse the already calculated approximation of $f_\alpha(M^*)+\mathbb{R}^m_+$ and, what is more important, we save the feasible points $\tilde{x}$ which are a minimal solution of $(P_{\overline{p},\,X^*})$ if its corresponding $\tilde{t}$ is between $-\frac{\varepsilon}{2}$ and 0. Note that the $\tilde{t} < -\frac{\varepsilon}{2}$ case is not possible for a box X^*, which passed Algorithm 4 with any bisecting ($\mathcal{B} = 1$). This fact will be shown in Lemma 4.10. The points $\tilde{x}$ with $-\frac{\varepsilon}{2} \leq \tilde{t} \leq 0$ are collected in the list $\mathcal{X}$ and will serve as the possible points of the (ε, δ)-efficient set $\mathcal{A}$.

Algorithm 5 Discarding test with static lists $\mathcal{L}_{PNS}, \mathcal{L}_{LUB}$

Input: $X^* \in \mathbb{IR}^n,\ f \in \mathcal{C}^2(\mathbb{R}^n, \mathbb{R}^m),\ g \in \mathcal{C}(\mathbb{R}^n, \mathbb{R}^p),\ \mathcal{L}_{PNS},\ \mathcal{L}_{LUB} = \{p^1, \ldots, p^k\} \subseteq \mathbb{R}^m,\ \varepsilon > 0,\ \alpha$

Output: list $\mathcal{X}$, flag $\mathcal{D}$

1: **if** there is no current outer approximation of $f_\alpha(M^*) + \mathbb{R}^m_+$ calculated yet **then**
2: Calculate ideal point a of f_α on M^* and initialize $\{a\} + \mathbb{R}^m_+$ as an outer approximation of $f_\alpha(M^*) + \mathbb{R}^m_+$
3: $\mathcal{D} \leftarrow 1$, $\mathcal{X} \leftarrow \emptyset$
4: **for** $s = 1, \ldots, r$ **do**
5: **if** p^s is inside the current outer approximation of $f_\alpha(M^*) + \mathbb{R}^m_+$ **then**
6: Solve $(P_{p^s,\,X^*})$ with minimal solution $(\tilde{x}, \tilde{t})$
7: Update outer approximation of $f_\alpha(M^*) + \mathbb{R}^m_+$
8: **if** $\tilde{t} \leq 0$ **then**
9: $\mathcal{X} \leftarrow \mathcal{X} \cup \tilde{x}$ and set flag $\mathcal{D} \leftarrow 0$

The next theorem shows that boxes with efficient points do not get discarded by Algorithm 5.

Theorem 4.4 *Let a box $X^* \subseteq X \in \mathbb{IR}^n$ and (MOP) be given. Let $\mathcal{L}_{LUB} \subseteq \mathbb{R}^m$ be a local upper bound set w. r. t. $\mathcal{L}_{PNS}$. If X^* contains an efficient point of (MOP), the output of Algorithm 5 is $\mathcal{D} = 0$, i. e., X^* will not be discarded by Algorithm 5.*

Proof. The proof is analogous to the proof of Theorem 4.3. □

4.1.3 Some Notes on the Termination Procedure

In contrast to many other B&B-algorithms, the present algorithm does not use one specific termination rule which can usually be applied directly. In [FT09], Fernández and Tóth use a similar termination rule to this one:
Store X^* in $\mathcal{L}_S$ if the following condition for given $\varepsilon, \delta > 0$ holds: $\omega(X^*) < \varepsilon$ and $\omega(F(X^*)) < \delta$.
Recall, $F\colon \mathbb{IR}^n \to \mathbb{IR}^m$ is the natural interval extension of f which evaluates the function with interval arithmetic, see Definition 3.4 or [Neu90]. Moreover, $\omega(X^*)$ denotes the box width of the box X^*, see (3.1).

We use a much more detailed termination procedure which is included in the discarding tests, see Algorithms 4 and 5. Thus, the first two `while`-loops use the following rules:

Termination rule for first `while`-loop: *Store the box X^* in the solution list if the output of Algorithm 4 applied to X^* is $\mathcal{D} = 0$ and $\mathcal{B} = 0$.*

Termination rule for second `while`-loop: *Store the box X^* in the solution list if the output of Algorithm 5 applied to X^* is $\mathcal{D} = 0$.*

As the algorithm should provide an (ε, δ)-efficient set at the end, a third `while`-loop is used. In this loop the termination rule is:

Termination rule for third `while`-loop: *Store the box X^* in the solution list if the output of Algorithm 5 applied to X^* is $\mathcal{D} = 0$, $\omega(X^*) \leq \delta$ and at least one of the $x \in \mathcal{X}$ is ε-efficient for (MOP).*

Note that the last condition is not easy to prove. Nevertheless, it is possible to find sufficient criteria for $x \in \mathcal{X}$ such that this condition holds for x. Hence, we use a weaker condition which is explained in more detail in Section 4.3. To summarize, these termination rules guarantee the (ε, δ)-efficiency of the calculated approximation set that will be shown in Section 4.4.2.

4.2 Selection and Bisection Rules

The selection rule is mostly heuristically motivated. In general, it is useful to consider boxes which can deserve "good bounds" at an early stage. Then boxes with worse bounds can be discarded early. Thus, we need a criterion which gives a direction to "good bounds" and which can be examined without or with a little further effort. A typical selection rule is the one proposed in [FT09].

Selection rule: *Select the box $X^* \in \mathcal{L}_W$ with a minimum lower bound of f_1.*

In our algorithm this lower bound will be calculated by underestimating f_1 within the considered box by a convex underestimator. In [FT09], the lower bound is calculated by interval arithmetic. Certainly, it is possible to replace f_1 by any $f_j,\ j \in \{1, \ldots, m\}$ or by a weighted sum of the objectives or similar. The observation of some numerical test runs shows that the nondominated set is explored firstly at the lower values for f_1 if the above selection rule is used. If f_1 is replaced by another f_j, the exploration starts at the smallest values for the j-th objective function.

The choosing of an appropriate bisection rule is another heuristic task. Consider a sequence of subboxes $X \supset X^1 \supset \ldots \supset X^k \supset \ldots$, where every X^k is obtained by bisecting the box $X^{k-1}, k = 1, \ldots,$ as defined by a bisection rule. We require that the box widths of such a sequence form a null sequence, i. e., $\lim_{k\to\infty} \omega(X^k) = 0$. Therefore, we use the following bisection rule.

Bisection rule: *Bisect the box $X^* \in \mathcal{L}_W$ perpendicularly to a direction of maximum width.*

4.3 The Complete Algorithm

Having now the new discarding and termination procedure as well as the rules for selection and bisection, we can present the complete algorithm. The whole algorithm for (MOP) is given in Algorithm 6.

The algorithm consists of three `while`-loops. The list $\mathcal{L}_{S,t},\ t = 1, 2, 3$ is the solution list of the t-th `while`-loop and becomes the working list for the next loop if $t = 1, 2$. The first loop from line 3 handles the basic discarding test, which was explained at the beginning of Section 4.1 in detail. It generates the list $\mathcal{L}_{PNS}$ until this list is close to the nondominated set in dependence of ε. All boxes X^* from $\mathcal{L}_{S,1}$ with $M^* = X^* \cap M$ have the following properties:

$$\exists \bar{p} \in \mathcal{L}_{LUB} : \bar{p} \in f_\alpha(M^*) + \mathbb{R}^m_+ \tag{4.2}$$

$$\forall p \in \mathcal{L}_{LUB} : p - \frac{\varepsilon}{2} e \notin f_\alpha(M^*) + \mathbb{R}^m_+ \tag{4.3}$$

The first property is true, because if all local upper bounds were outside of the set $f_\alpha(M^*) + \mathbb{R}^m_+$, the box X^* would be discarded, see Lemma 4.2 for the proof. If the second property did not hold for X^*, this box would not be stored in the solution list $\mathcal{L}_{S,1}$, because line 7 of Algorithm 4 would be satisfied and X^* would be bisected into subboxes.

With the second and third `while`-loop of `MOPBB` we do not lose characteristics (4.2) and (4.3). The second loop only checks whether some boxes from $\mathcal{L}_{S,1}$ can be discarded by the final lists $\mathcal{L}_{PNS}$ and $\mathcal{L}_{LUB}$. In the third loop we check for every box $X^* \in \mathcal{L}_{S,2}$ whether $\omega(X^*)$ is less than or equal to δ for a predefined $\delta > 0$. If the box X^* is not small enough, we bisect it and apply the discarding test to both subboxes. Moreover, we compute the (ε, δ)-efficient set $\mathcal{A}$, which can be stated as:

$$\mathcal{A} := \{x \in M \mid f(x) \in \mathcal{L}_{PNS}\}$$
$$\cup \bigcup_{X^* \in \mathcal{L}_{S,2}} \left\{ x \in M^* \;\middle|\; \begin{array}{l} (\exists\, \bar{p} \in \mathcal{L}_{LUB}, t \leq 0 : (x,t) \text{ is a min. solution of } (P_{\bar{p},X^*})) \\ \wedge\, (\omega(X^*) \leq \delta) \\ \wedge\, \left((\exists \tilde{p} \in \mathcal{L}_{LUB} : f(x) \leq \tilde{p}) \vee (\omega(X^*) < \sqrt{\tfrac{\varepsilon}{\alpha}})\right) \end{array} \right\}$$

Algorithm 6 MOPBB: Algorithm to find an (ε, δ)-efficient set of (MOP)

Input: $X \in \mathbb{IR}^n,\ f \in \mathcal{C}^2(\mathbb{R}^n, \mathbb{R}^m),\ g \in \mathcal{C}(\mathbb{R}^n, \mathbb{R}^p),\ \varepsilon > 0,\ \delta > 0$

Output: $\mathcal{A},\ \mathcal{L}_{S,3},\ \mathcal{L}_{PNS},\ \mathcal{L}_{LUB}$

$\mathcal{L}_W \leftarrow \{X\}, \mathcal{L}_{S,1} \leftarrow \emptyset, \mathcal{L}_{S,2} \leftarrow \emptyset, \mathcal{L}_{S,3} \leftarrow \emptyset, \mathcal{A} \leftarrow \emptyset, \mathcal{L}_{PNS} \leftarrow \emptyset, \mathcal{L}_{LUB} \leftarrow \emptyset$
Calculate α such that $f_{j,\alpha}$ is a convex underestimator of f_j on $X,\ j = 1, \ldots, m$
while $\mathcal{L}_W \neq \emptyset$ **do**
    Select a box X^* from $\mathcal{L}_W$ and delete it from $\mathcal{L}_W$
    Bisect X^* perpendicularly to a direction of maximum width $\rightarrow X^1, X^2$
    for $l = 1, 2$ **do**
        Apply Algorithm 4 to X^l
        if $\mathcal{B} = 1$ **then** Store X^l in $\mathcal{L}_W$
        else if $\mathcal{D} = 0$ **then** Store X^l in $\mathcal{L}_{S,1}$
        else Discard X^l
while $\mathcal{L}_{S,1} \neq \emptyset$ **do**
    Select a box X^* from $\mathcal{L}_{S,1}$ and delete it from $\mathcal{L}_{S,1}$
    Apply Algorithm 5 to X^*
    if $\mathcal{D} = 0$ **then** Store X^* in $\mathcal{L}_{S,2}$
    else Discard X^*
while $\mathcal{L}_{S,2} \neq \emptyset$ **do**
    Select a box X^* from $\mathcal{L}_{S,2}$ and delete it from $\mathcal{L}_{S,2}$
    Apply Algorithm 5 to X^* and obtain $\mathcal{X}$
    if $\mathcal{D} = 1$ **then** Discard X^*
    else if $\mathcal{D} = 0$ **and** $\omega(X^*) \leq \delta$ **then**
        $\{x^1, \ldots, x^k\} \leftarrow \mathcal{X}$
        for $s = 1, \ldots, r$ **do**
            if $f(x^s) \leq \bar{p}$ for at least one $\bar{p} \in \mathcal{L}_{LUB}$ **or** $\omega(X^*) < \sqrt{\frac{\varepsilon}{\alpha}}$ **then**
                $\mathcal{A} \leftarrow \mathcal{A} \cup \{x^s\}$
                Store X^* in $\mathcal{L}_{S,3}$
    if $\mathcal{D} = 0$ **and** no point of $\mathcal{X}$ was stored in $\mathcal{A}$ **then**
        Bisect X^* perpendicularly to a direction of maximum width $\rightarrow X^1, X^2$
        Store X^1 and X^2 in $\mathcal{L}_{S,2}$
$\mathcal{A} \leftarrow \mathcal{A} \cup \{x \in X \mid f(x) \in \mathcal{L}_{PNS}\}$

Note that the union in the description of $\mathcal{A}$ is not a disjoint union. Moreover, the list $\mathcal{L}_{S,2}$ in this description includes all boxes which have been in $\mathcal{L}_{S,2}$ during the execution of MOPBB at least once. In fact, these are all boxes of $\mathcal{L}_{S,2}$ after the second `while`-loop in line 15 and those which were obtained by bisection in line 27. The proof of the (ε, δ)-efficiency of $\mathcal{A}$ is a main part of Section 4.4.2. To explain the definition of the sets in the second line of the description of $\mathcal{A}$ we have to consider the third `while`-loop: When a subbox X^* is considered in this loop, $(P_{\bar{p},\, X^*})$ is solved for some local upper bounds $\bar{p}$. A point x, which belongs to a minimal solution (x, t) of $(P_{\bar{p},\, X^*})$ and with a nonpositive t, is a possible candidate for an ε-efficient point and thus $x \in \mathcal{X}$. If $f(x)$ is less than or equal to a local upper bound, then x is added to $\mathcal{A}$. Furthermore, if $\omega(X)$ is bounded by $\sqrt{\frac{\varepsilon}{\alpha}}$ (usually smaller than δ) we add all elements of $\mathcal{X}$ to $\mathcal{A}$. Note that $\mathcal{X} \neq \emptyset$ if and only if $\mathcal{D} = 0$. Moreover, the preimages of the points of the final list $\mathcal{L}_{PNS}$, i. e., points of the first set in the description of $\mathcal{A}$, are also added to $\mathcal{A}$ at the end of the algorithm.

4.4 Convergence Results

This section details the results regarding the finiteness and correctness of the new algorithm MOPBB.

4.4.1 Termination

To show the termination of the algorithm MOPBB we have to verify that each `while`-loop of MOPBB is finite. We start by showing the termination of the first `while`-loop.

Lemma 4.5 *The first `while`-loop (lines 3-10) of* MOPBB *terminates.*

Proof. Assume the first `while`-loop does not terminate. Hence, there must be an infinite sequence of boxes $X \supset X^1 \supset \ldots \supset X^k \supset \ldots$ which were not discarded, but bisected after applying Algorithm 4 on each box. Thus, every box X^k will be stored in $\mathcal{L}_W$ and bisected in another iteration, where the box X^{k+1} is one of the two obtained subboxes. Obviously, the box width decreases among the sequence of boxes,

i. e., $\omega(X^k) \geq \omega(X^{k+1})$ for every $k \in \mathbb{N}$ and converges to 0 (because we divide the boxes perpendicular to a side with maximal width). For δ_X as in (3.4) choose the first box $X^{\tilde{k}}$ with $\omega(X^{\tilde{k}}) \leq \delta_X$. Let $a \in \mathbb{R}^m$ be the ideal point of f on $M^{\tilde{k}} := X^{\tilde{k}} \cap M$ (see (3.5)). From Lemma 3.14 for every objective function it results that for all $x \in M^{\tilde{k}}$ and all $j = 1, \ldots, m$ it holds $|f_j(x) - a_j| \leq \frac{\varepsilon}{2}$, or

$$f(x) \in \left(\{a\} + \mathbb{R}^m_+\right) \cap \left(\left\{a + \frac{\varepsilon}{2}e\right\} - \mathbb{R}^m_+\right). \tag{4.4}$$

Now consider the minima of each convex underestimator. Choose an arbitrary point $\tilde{x}^j \in \operatorname{argmin}\{f_{j,\alpha}(x) \mid x \in M^{\tilde{k}}\}$ for every $j = 1, \ldots, m$. The images of the points $\tilde{x}_j$ under the original function, i. e., $f(\tilde{x}^j)$, are potential points of the list $\mathcal{L}_{PNS}$ (see line 2 of Algorithm 4) and clearly satisfy (4.4) as well. Let us choose one of these and denote it by q. This point will be added to $\mathcal{L}_{PNS}$ if there is no other point from the current list $\mathcal{L}_{PNS}$ dominating q.

Because of the assumption that $X^{\tilde{k}}$ will be bisected, there must be a local upper bound $\bar{p}$ with the minimal solution $(\tilde{x}, \tilde{t})$ of $(P_{\bar{p},\, X^{\tilde{k}}})$, where $\tilde{t}$ is less than $-\frac{\varepsilon}{2}$. Now, we check for each local upper bound if this is possible. If we show that for any local upper bound $\bar{p}$ the minimal solution of $(P_{\bar{p},\, X^{\tilde{k}}})$ is never less than $-\frac{\varepsilon}{2}$, we can conclude that the assumption is wrong and $X^{\tilde{k}}$ will not be bisected.

First, consider all local upper bounds which do not belong to $\{a\} + \mathbb{R}^m_+$. Therefore, let $\bar{p} \in \mathcal{L}_{LUB} \setminus (\{a\} + \mathbb{R}^m_+)$, i. e., there is a $u \in \{1, \ldots, m\}$ with $\bar{p}_u < a_u$. The condition in line 5 of Algorithm 4 is not satisfied and $(P_{\bar{p},\, X^{\tilde{k}}})$ will not be solved.

Next, we consider those $\bar{p} \in \mathcal{L}_{LUB}$ with $\bar{p} \in \mathcal{L}_{LUB} \cap (\{a\} + \mathbb{R}^m_+)$, if there exist any, and distinguish two cases: The first of which is

$$|\bar{p}_u - a_u| = \bar{p}_u - a_u \leq \frac{\varepsilon}{2} \text{ for one } u \in \{1, \ldots, m\}. \tag{4.5}$$

Problem $(P_{\bar{p},\, X^{\tilde{k}}})$ is solved and has a minimal solution $(\tilde{x}, \tilde{t})$. Then $\bar{p}_u + \tilde{t} \geq f_{u,\alpha}(\tilde{x}) \geq a_u$ holds. Hence, $\frac{\varepsilon}{2} \geq \bar{p}_u - a_u \geq -\tilde{t}$, which leads to $\tilde{t} \geq -\frac{\varepsilon}{2}$. In the second case, there is some $\bar{p} \in \mathcal{L}_{LUB} \cap (\{a\} + \mathbb{R}^m_+)$ with

$$|\bar{p}_j - a_j| = \bar{p}_j - a_j > \frac{\varepsilon}{2} \text{ for all } j \in \{1, \ldots, m\} \tag{4.6}$$

or equivalently $\bar{p} \in \{a + \frac{\varepsilon}{2}e\} + \text{int}(\mathbb{R}^m_+)$. It follows for every $j \in \{1, \ldots, m\}$ that $a_j + \frac{\varepsilon}{2} < \bar{p}_j$. But we know there is a point q (see above) which is a candidate for $\mathcal{L}_{PNS}$ and belongs to the set $(\{a\} + \mathbb{R}^m_+) \cap (\{a + \frac{\varepsilon}{2}e\} - \mathbb{R}^m_+)$. Define

$$y := \begin{cases} q' & \text{if there is a } q' \in \mathcal{L}_{PNS} \text{ with } q' \leq q \\ q & \text{otherwise,} \end{cases}$$

and thus $y \in \mathcal{L}_{PNS}$. By considering each component of a, y and $\bar{p}$, we obtain the inequalities $y_j \leq q_j \leq a_j + \frac{\varepsilon}{2} < \bar{p}_j$ for all $j = 1, \ldots, m$. Hence, y strictly dominates $\bar{p}$, which is a contradiction to Proposition 3.23 (i). Thus, the existence of a local upper bound in $\{a + \frac{\varepsilon}{2}e\} + \text{int}(\mathbb{R}^m_+)$ is not possible. Clearly, it is not possible for $X^{\tilde{k}}$ to satisfy the conditions for bisection. Hence, the assumed infinite sequence of subboxes does not exist. According to this, the first `while`-loop will terminate. □

Lemma 4.6 *The second `while`-loop (lines 11-15) of* MOPBB *terminates.*

Proof. The termination of the second `while`-loop is obvious, because it has exactly $|\mathcal{L}_{S,1}|$ iterations. □

Lemma 4.7 *The third `while`-loop (lines 16-28) of* MOPBB *terminates.*

Proof. Assume the third `while`-loop does not terminate. Hence, there must be an infinite sequence of boxes $X^1 \supset \ldots \supset X^k \supset \ldots$ with $X^1 \in \mathcal{L}_{S,2}$ after the second loop, which were not discarded, but bisected after applying Algorithm 5 on each box. Hence, every box X^k will be stored in $\mathcal{L}_{S,2}$ and bisected in another iteration, where the box X^{k+1} is one of the two obtained subboxes. Obviously, the box width decreases among the sequence of boxes, i. e., $\omega(X^k) \geq \omega(X^{k+1})$ for every $k \in \mathbb{N}$ and converges to 0 (because we divide the boxes perpendicular to a side with maximal width). Let us choose the first box $X^{\tilde{k}}$ with $\omega(X^{\tilde{k}}) < \min\{\delta, \sqrt{\frac{\varepsilon}{\alpha}}\}$. For $X^{\tilde{k}}$ we have $\mathcal{D} = 0$, otherwise it will be discarded. Therefore, the conditions in lines 20 and 23 of MOPBB are satisfied and $X^{\tilde{k}}$ will be stored in $\mathcal{L}_{S,3}$. This contradicts the assumption that $X^{\tilde{k}}$ will be bisected. Eventually, the assumed infinite sequence of subboxes does not exist and the third `while`-loop terminates. □

Proposition 4.8 *By Lemmas* 4.5 *to* 4.7 *we obtain that the whole algorithm* MOPBB *terminates.*

4.4.2 Correctness

First, we state that all efficient points x of (MOP) are contained in the union of boxes from the final list $\mathcal{L}_{S,3}$:

Lemma 4.9 *Let $\mathcal{L}_{S,3}$ be the output of* MOPBB *for arbitrary $\varepsilon, \delta > 0$ and let $\mathcal{L}_{S,1}$ and $\mathcal{L}_{S,2}$ be the lists after the first and the second* `while`*-loops, respectively. Then*

$$X_E \subseteq \bigcup_{X^* \in \mathcal{L}_{S,3}} X^* \subseteq \bigcup_{X^* \in \mathcal{L}_{S,2}} X^* \subseteq \bigcup_{X^* \in \mathcal{L}_{S,1}} X^*.$$

Proof. This is a direct consequence of Theorems 4.3 and 4.4 and the way the lists are constructed. □

The next two lemmas show that in Algorithm 5 the case of $\tilde{t} < -\frac{\varepsilon}{2}$ is not possible in the second and third `while`-loops of MOPBB (Algorithm 6) as mentioned in Section 4.1 on page 51.

Lemma 4.10 *Let $X^* \in \mathbb{IR}^m$ be chosen from the working list $\mathcal{L}_{S,1}$ during the second* `while`*-loop of* MOPBB *and hence be an input for Algorithm* 5*. If $(P_{\bar{p},\, X^*})$ is solved for any $\bar{p} \in \mathcal{L}_{LUB}$ within Algorithm* 5*, we obtain a minimal solution (x^*, t^*) with $t^* \geq -\frac{\varepsilon}{2}$.*

Proof. Let $\bar{p} \in \mathcal{L}_{LUB}$ be inside the current outer approximation of $f_\alpha(X^* \cap M) + \mathbb{R}^m_+$ and let (x^*, t^*) be the minimal solution of $(P_{\bar{p},\, X^*})$. Assume now that $t^* < -\frac{\varepsilon}{2}$ holds. In particular, by $f_\alpha(x^*) \leq \bar{p} + t^* e < \bar{p}$ we obtain $\bar{p} \in f_\alpha(X^* \cap M) + \mathbb{R}^m_+$. Because of $X^* \in \mathcal{L}_{S,1}$, this box was not discarded in the first `while`-loop of MOPBB, i. e., $\mathcal{D} = 0$ and $\mathcal{B} = 0$. For the next steps consider X^* during the first `while`-loop, where Algorithm 4 is called. Let $\mathcal{L}'_{LUB}$ be the set of local upper bounds at this time. Then it holds that

$$\forall p' \in \mathcal{L}'_{LUB} : \ (P_{p',\, X^*}) \text{ was solved with minimal solution } (x', t') \Rightarrow t' \geq -\frac{\varepsilon}{2}. \quad (4.7)$$

Now, we distinguish two cases, the first is $\bar{p} \in \mathcal{L}'_{LUB}$. Because $\bar{p} \in f_\alpha(X^* \cap M) + \mathbb{R}^m_+$ holds, the problem $(P_{\bar{p},\, X^*})$ was solved. As (x^*, t^*) is feasible for $(P_{\bar{p},\, X^*})$ with $t^* < -\frac{\varepsilon}{2}$, this contradicts (4.7).

The second case is $\bar{p} \notin \mathcal{L}'_{LUB}$, i. e., $\bar{p}$ was added to the set of local upper bounds after X^* was considered in the first `while`-loop. Then there exists a $p^* \in \mathcal{L}'_{LUB}$ with $\bar{p} \leq p^*$ and $\bar{p} \neq p^*$, see Remark 3.26. Because of $\bar{p} \in f_\alpha(X^* \cap M) + \mathbb{R}^m_+$, it also holds that $p^* \in f_\alpha(X^* \cap M) + \mathbb{R}^m_+$ and the optimization problem (P_{p^*, X^*}) was solved in Algorithm 4. Then (x^*, t^*) is also feasible for (P_{p^*, X^*}), because $f_\alpha(x^*) \leq \bar{p} + t^* e \leq p^* + t^* e$, which contradicts (4.7). □

Lemma 4.11 *Let $X^* \in \mathbb{IR}^m$ be chosen from the working list in $\mathcal{L}_{S,2}$ during the third `while`-loop of* MOPBB *and hence be an input for Algorithm* 5*. If $(P_{\bar{p}, X^*})$ is solved for any $\bar{p} \in \mathcal{L}_{LUB}$ within Algorithm* 5*, we obtain a minimal solution (x^*, t^*) with $t^* \geq -\frac{\varepsilon}{2}$.*

Proof. Let $\bar{p} \in \mathcal{L}_{LUB}$ be inside the current outer approximation of $f_\alpha(X^* \cap M) + \mathbb{R}^m_+$ and let (x^*, t^*) be the minimal solution of $(P_{\bar{p}, X^*})$. Assume now that $t^* < -\frac{\varepsilon}{2}$. In particular, by $f_\alpha(x^*) \leq \bar{p} + t^* e < \bar{p}$ we obtain $\bar{p} \in f_\alpha(X^* \cap M) + \mathbb{R}^m_+$. Note that the set $\mathcal{L}_{LUB}$ is fixed after the first loop.

Because of Lemma 4.10 the box X^* was not considered in the second `while`-loop, i. e., $X^* \notin \mathcal{L}_{S,2}$ after line 15. But there exists a box $\tilde{X}$ with $X^* \subseteq \tilde{X}$, which has been considered and not discarded in this loop. Let $\tilde{f}_\alpha$ be the componentwise convex underestimator of f on $\tilde{X} = [\underline{x'}, \overline{x'}]$, i. e., $\tilde{f}_{j,\alpha}(x) = f(x) - \frac{\alpha}{2}(\underline{x'} - x)^T(\overline{x'} - x)$ for all $x \in \tilde{X}, j = 1, \ldots, m$. The convex underestimator $\tilde{f}_\alpha$ is clearly less than f_α on X^*, i. e., $\tilde{f}_\alpha(x) \leq f_\alpha(x)$ for all $x \in X^*$, see also Lemma 3.11. Because of $\bar{p} \in f_\alpha(X^* \cap M) + \mathbb{R}^m_+$ it also holds that $\bar{p} \in \tilde{f}_\alpha(\tilde{X} \cap M) + \mathbb{R}^m_+$ and the optimization problem $(P_{\bar{p}, \tilde{X}})$ was solved in Algorithm 5. Let $(\tilde{x}, \tilde{t})$ be a minimal solution of $(P_{\bar{p}, \tilde{X}})$. Because of Lemma 4.10 we have $\tilde{t} \geq -\frac{\varepsilon}{2}$. Since $X^* \subseteq \tilde{X}$ and $f_\alpha(x^*) \geq f^*_\alpha(x^*)$ the pair (x^*, t^*) is also feasible for $(P_{\bar{p}, X^*})$, which contradicts the minimality of $(\tilde{x}, \tilde{t})$. □

Based on the list $\mathcal{L}_{LUB}$ and thus on the list $\mathcal{L}_{PNS}$ which are calculated by MOPBB, one obtains a set, which contains all nondominated points of (MOP). We prove this in Theorem 4.13. This set, which looks like a staircase-shaped tube or pipe in the two-dimensional case, is defined as follows:

$$T := \left(\bigcup_{\bar{p} \in \mathcal{L}_{LUB}} \{\bar{p}\} - \mathbb{R}^m_+ \right) \setminus \left(\bigcup_{\bar{p} \in \mathcal{L}_{LUB}} \left\{ \bar{p} - \tfrac{\varepsilon}{2} e \right\} - \operatorname{int}(\mathbb{R}^m_+) \right). \tag{4.8}$$

An illustration of such a set T is shown in Figure 4.5. Recall that $\hat{Z}$ was defined as a box with $f(X) \subseteq \text{int}(\hat{Z})$.

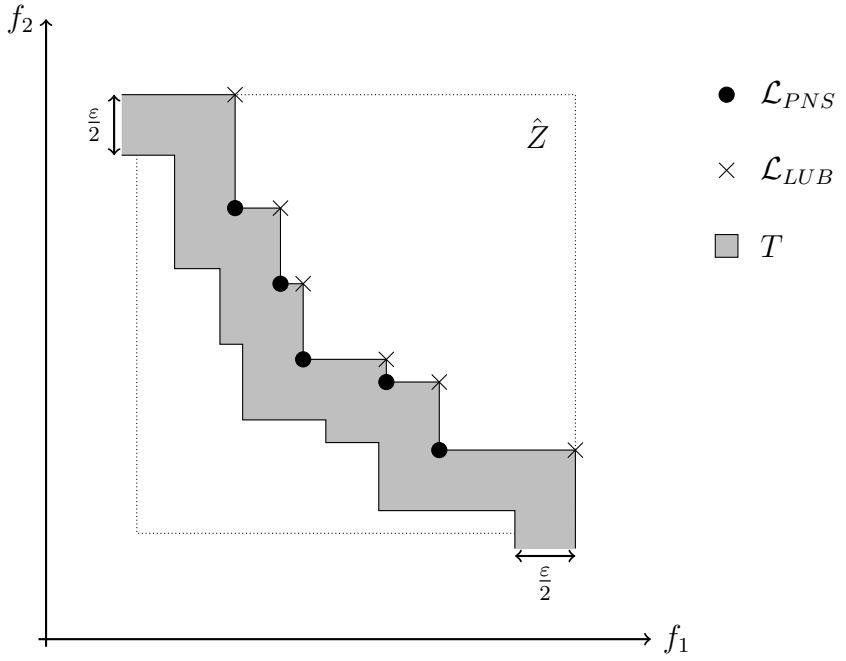


Figure 4.5. The set T, which depends on $\frac{\varepsilon}{2}$; $m = 2$.

The set T contains all points from $\mathcal{L}_{PNS}$, which will be shown in the next lemma.

Lemma 4.12 *Let $\mathcal{L}_{LUB}$ be the local upper bound set w. r. t. $\mathcal{L}_{PNS}$. Let T be defined as in* (4.8)*. Then $\mathcal{L}_{PNS} \subseteq T$.*

Proof. Because of Lemma 3.24 there exists for every point $q \in \mathcal{L}_{PNS}$ a point $p^q \in \mathcal{L}_{LUB}$ with $q \in \{p^q\} - \mathbb{R}^m_+$. Assume q belongs to a set $\{p - \frac{\varepsilon}{2}e\} - \text{int}(\mathbb{R}^m_+)$ for any $p \in \mathcal{L}_{LUB}$. Then it holds for every $j \in \{1, \ldots, m\}$ that $q_j < p_j - \frac{\varepsilon}{2} < p_j$. Certainly, this contradicts Proposition 3.23 (i). Thus, $\mathcal{L}_{PNS} \subseteq T$. □

Theorem 4.13 *Let $\mathcal{L}_{LUB}$ be the local upper bound set w. r. t. $\mathcal{L}_{PNS}$ with $\mathcal{L}_{PNS}$ from* MOPBB *and let T be defined as in* (4.8)*. Let $\bar{x}$ be an efficient point of (MOP). Then $f(\bar{x}) \in T$, i. e., the nondominated set of (MOP) is a subset of T.*

Proof. Let $\bar{x} \in X_E$, i. e., $\bar{x}$ is an efficient point of (MOP). We assume that $f(\bar{x}) \notin T$. There are two possibilities: First, $f(\bar{x})$ lies above T, i. e., $f(\bar{x}) \in \hat{Z} \setminus (T - \mathbb{R}^m_+)$. By

$f(\bar{x}) \notin T - \mathbb{R}^m_+$, it holds for all $p \in \mathcal{L}_{LUB}$ that $f(\bar{x}) \not\leq p$. In particular, $f(\bar{x}) \notin S$, where S is the search region (see (3.18)). Otherwise, we have a contradiction to Definition 3.21 (i). Using the definition of S, it follows that there exists a point $q \in \mathcal{L}_{PNS}$ with $q \leq f(\bar{x})$. Since q is an image of a feasible point and $\bar{x}$ is efficient, $q = f(\bar{x})$ holds. However, we have assumed that $f(\bar{x}) \notin T$, which contradicts $q \in \mathcal{L}_{PNS} \subseteq T$.

The other case is $f(\bar{x}) \in (T - \mathbb{R}^m_+) \setminus T$. Hence, there is a local upper bound $\bar{p} \in \mathcal{L}_{LUB}$ with $f(\bar{x}) < \bar{p} - \frac{\varepsilon}{2}e$, which leads to the chain of inequalities $f_\alpha(\bar{x}) \leq f(\bar{x}) < \bar{p} - \frac{\varepsilon}{2}e$. By Lemma 4.9 there is some box $X \in \mathcal{L}_{S,1}$ with $\bar{x} \in X$ and thus $\bar{p} - \frac{\varepsilon}{2} \in f_\alpha(X \cap M) + \mathbb{R}^m_+$. This is a contradiction to (4.3). Therefore, $f(\bar{x})$ belongs to T. □

Remark 4.14 As a consequence of Theorem 4.13 we also have $f(M) \subseteq T + \mathbb{R}^m_+$, i. e., $f(M) \cap (\mathbb{R}^m \setminus (T + \mathbb{R}^m_+)) = \emptyset$, which means that no image of a feasible point of (MOP) lies below T. This is due to the fact that the ordering cone $\mathbb{R}^m_+$ is a pointed closed convex cone and $f(M)$ is a compact set, and thus external stability holds, (cf. [SNT85, Th. 3.2.9]).

Next, we show the (ε, δ)-efficiency of the output set $\mathcal{A}$. Recall that the set $\mathcal{X}$ is calculated individually for every considered subbox X^* in Algorithm 5. All $x \in \mathcal{X}$ may be ε-efficient points of (MOP). In MOPBB, line 23, two conditions are checked and we will prove in the next lemmas that for those $x \in \mathcal{X}$ which satisfy one of the conditions in line 23, ε-efficiency indeed holds.

Lemma 4.15 *Every subbox $X^* \in \mathbb{IR}^n$ of X which is not discarded in the third `while`-loop of* MOPBB *and with $\omega(X^*) < \sqrt{\frac{\varepsilon}{\alpha}}$ contains a point $\tilde{x}$ which is ε-efficient.*

Proof. Set $\tilde{\delta} := \sqrt{\frac{\varepsilon}{\alpha}}$ and let $X^* \subseteq X$ be a box with $\omega(X^*) < \tilde{\delta}$. By Remark 3.13 we know that for arbitrary $x \in X^*$ and all $j = 1, \ldots, m$ it holds that

$$f_j(x) - f_{j,\alpha}(x) = |f_j(x) - f_{j,\alpha}(x)| \leq \frac{\alpha}{2}\omega(X)^2 < \frac{\alpha}{2} \cdot \frac{\varepsilon}{\alpha} = \frac{\varepsilon}{2}.$$

As f_α is a convex underestimator of f (componentwise), we obtain

$$f_{j,\alpha}(x) \leq f_j(x) < f_{j,\alpha}(x) + \frac{\varepsilon}{2} \text{ for every } j \in \{1, \ldots, m\} \text{ and all } x \in X^*, \qquad (4.9)$$

which is equivalent to $f(x) \in \left(\left\{f_\alpha(x) + \frac{\varepsilon}{2}e\right\} - \text{int}(\mathbb{R}^m_+)\right) \cap \left(\{f_\alpha(x)\} + \mathbb{R}^m_+\right)$. The box X^* was not discarded in the third `while`-loop. Hence, Algorithm 5 applied to X^* delivered the output $\mathcal{D} = 0$ and a list $\mathcal{X} \neq \emptyset$, see line 9 of Algorithm 5. Consider an $\tilde{x} \in X^* \cap M$, which was calculated in line 6 of Algorithm 5 w. r. t. the corresponding local upper bound $p^s \in \mathcal{L}_{LUB}$. Note that for the minimal solution $(\tilde{x}, \tilde{t})$ of (P_{p^s, X^*}) we have $-\frac{\varepsilon}{2} \leq \tilde{t} \leq 0$. Recall that $-\frac{\varepsilon}{2} \leq \tilde{t}$ holds because of Lemma 4.11. Suppose now that there is an $\hat{x} \in M$ with $f(\hat{x}) \leq f(\tilde{x}) - \varepsilon e$ and $f(\hat{x}) \neq f(\tilde{x}) - \varepsilon e$, which is equivalent to $f(\hat{x}) \in \{f(\tilde{x})\} - \{\varepsilon e\} - (\mathbb{R}^m_+ \setminus \{0_m\})$. Hence, with the above shown property (4.9) we conclude that

$$\begin{aligned} f(\hat{x}) &\in \{f(\tilde{x})\} - \{\varepsilon e\} - (\mathbb{R}^m_+ \setminus \{0_m\}) \\ &\subseteq \left(\left(\left\{f_\alpha(\tilde{x}) + \frac{\varepsilon}{2}e\right\} - \text{int}(\mathbb{R}^m_+)\right) \cap \left(\{f_\alpha(\tilde{x})\} + \mathbb{R}^m_+\right)\right) - \{\varepsilon e\} - (\mathbb{R}^m_+ \setminus \{0_m\}) \\ &\subseteq \left\{f_\alpha(\tilde{x}) - \frac{\varepsilon}{2}e\right\} - \text{int}(\mathbb{R}^m_+), \end{aligned} \tag{4.10}$$

which is equivalent to $f(\hat{x}) < f_\alpha(\tilde{x}) - \frac{\varepsilon}{2}$.

As $(\tilde{x}, \tilde{t})$ is a feasible point of (P_{p^s, X^*}), it holds that $p^s + \tilde{t}e \geq f_\alpha(\tilde{x})$. Given this and $\tilde{t} \leq 0$, we obtain $f_\alpha(\tilde{x}) \leq p^s$. By (4.10) it follows that $f(\hat{x}) < p^s - \frac{\varepsilon}{2}$ and, thus, $f(\hat{x})$ is not in $T + \mathbb{R}^m_+$, which contradicts Remark 4.14 that no feasible image is below T. Hence, $\tilde{x}$ is ε-efficient. □

Lemma 4.16 *Let $\mathcal{A}$ be the set generated by* MOPBB. *Then every $\tilde{x} \in \mathcal{A}$ is an ε-efficient point of* (MOP).

Proof. Let $\tilde{x}$ be an arbitrary element of $\mathcal{A}$. If $\tilde{x}$ is added in line 29 to $\mathcal{A}$, the point $f(\tilde{x})$ is an element of $\mathcal{L}_{PNS}$ and thus by Lemma 3.24 there is a local upper bound $\bar{p} \in \mathcal{L}_{LUB}$ with $f(\tilde{x}) \leq \bar{p}$. Suppose there is an $\hat{x} \in M$ with $f(\hat{x}) \leq f(\tilde{x}) - \varepsilon e$ and $f(\hat{x}) \neq f(\tilde{x}) - \varepsilon e$. With $f(\tilde{x}) \leq \tilde{p}$ we obtain $f(\hat{x}) \leq \tilde{p} - \varepsilon e$ and hence $f(\hat{x}) \notin T$, but $f(\hat{x})$ lies below T. That contradicts Remark 4.14.

If $\tilde{x}$ is added in line 24, $\tilde{x}$ belongs to a box X^* with $\omega(X^*) \leq \delta$ which was not discarded in the third `while`-loop, and moreover, $\tilde{x} \in \mathcal{X}$. Hence, there is a local upper bound $\bar{p} \in \mathcal{L}_{LUB}$ such that $(\tilde{x}, \tilde{t})$ is a minimal solution of $(P_{\bar{p}, X^*})$ and $-\frac{\varepsilon}{2} \leq \tilde{t} \leq 0$. Recall that $-\frac{\varepsilon}{2} \leq \tilde{t}$ holds because of Lemma 4.11. As the first condition in line 23 of MOPBB is

satisfied, we have $f(\tilde{x}) \leq \tilde{p}$ for a local upper bound $\tilde{p} \in \mathcal{L}_{LUB}$. By the same arguments as at the beginning of this proof, we can show that $\tilde{x}$ is ε-efficient.

If the first condition in line 23 is not satisfied, but the second one is, i. e., $\omega(X^*) < \sqrt{\frac{\varepsilon}{\alpha}}$, we have to show that each $x \in \mathcal{X}$ is ε-efficient. For this we refer the reader to the proof of Lemma 4.15, because the points in $\mathcal{X}$ for a box X^* with $\omega(X^*) < \sqrt{\frac{\varepsilon}{\alpha}}$ are exactly those ε-efficient points from Lemma 4.15. □

Theorem 4.17 *Let $\mathcal{A}$ be the set generated by* MOPBB. *Then $\mathcal{A}$ is an (ε, δ)-efficient set of* (MOP).

Proof. With Lemma 4.16, we know that $\mathcal{A}$ only contains ε-efficient points.

Let $x^* \in X_E$ be efficient for (MOP). With Lemma 4.9, we know that a box X^* which contains x^* cannot be discarded in any `while`-loop. Thus, choose now a box $X^* \in \mathcal{L}_{S,3}$ with $x^* \in X^*$. This box exists because the algorithm terminates; see Lemmas 4.5 to 4.7. In lines 23 and 24 of MOPBB, X^* is stored in $\mathcal{L}_{S,3}$ and an ε-efficient point $x \in \mathcal{X} \subseteq X^*$ is added to the set $\mathcal{A}$. Moreover, $\omega(X^*) \leq \delta$ and thus $\|x - x^*\| \leq \delta$. □

4.5 Discussion of Related Procedures

This section provides an overview of global algorithms for solving multiobjective optimization problems. Commonly, a *global algorithm* is an algorithm which is able — or aims — to find globally optimal or globally efficient points. This section includes deterministic algorithms as well as heuristic algorithms. First, we consider B&B algorithms, which are deterministic and ensure an accuracy of the calculated solutions like MOPBB. In the second section, we explain heuristically motivated procedures. One of them is the most-known and state-of-the-art evolutionary algorithm NSGA-II.

4.5.1 B&B Algorithms

All B&B algorithms have some aspects in common: Next to executing partitions in the preimage or image space, they need lower and upper bounds for the minimal values or nondominated points. Thereby, upper bounds are always objective values of feasible points. For obtaining feasible points, different strategies can be used. Since most methods are for solving multiobjective optimization problems with box constraints, the midpoint of a subbox is a suitable choice. The lower bounding procedures are often a part of the discarding tests, see Section 4.5.1.1.

Despite all multiobjective B&B algorithms aim to find approximate solutions of a multiobjective optimization problem, not all of them ensure a certain accuracy of their computed solution. In Section 2.4, we introduced different concepts for approximate solutions. The discarding tests are mostly developed in such a way that the obtained solutions fulfill the properties of one of these specific optimality concepts.

A very general B&B algorithm for multiobjective optimization problems with convergence theory was presented in [Sch12]. There, the feasible set is only described by box constraints. The bounding procedure consists of computing lower and upper bounds for all objective values on a subbox, but it is not further specified. The algorithm described in [Sch12, Sec. 4.3] ensures that its output only consists of ε-Pareto optimal points, see Definition 2.9. One difference to MOPBB is that the algorithm by [Sch12] works with an additional upper bounding procedure. This one is necessary in [Sch12] to prove the ε-Pareto optimality of the points belonging to the output set. Another

important difference between MOPBB and the general framework by [Sch12] is the lower bounding procedure. While the objective functions are considered separately in [Sch12] (see also the forthcoming Definition 4.18), MOPBB considers all objective functions as one unit.

Another algorithm which ensures some accuracies is the one proposed in [EP13; EP14]. This algorithm generates for multiobjective optimization problems with box constraints a set of points which do not dominate each other, i. e., a stable set. This set is an ε-Pareto set, see Definition 2.11.

The B&B algorithms by [FT09; Mar+16; ACA19] have the same foundations, which are set by [FT09], and are based on the other, respectively. Therefore, they have similar convergence results. The authors of [FT09] name different termination rules. One of them is that a box X^* is stored in the solution list $\mathcal{L}_S$ if $X^* \cap M \neq \emptyset$, $\omega(F_1(X^*)) < \varepsilon_1$ and $\omega(F_2(X^*)) < \varepsilon_2$ hold for two given scalars $\varepsilon_1, \varepsilon_2 > 0$. In this case, the algorithm of [FT09] ensures that all points of the boxes from a final solution list are not $(2\varepsilon_1, 2\varepsilon_2)$-superdominated, see Definition 2.13.

4.5.1.1 Other Discarding Tests

The most used discarding test executes the pairwise comparison of a lower bound of the objective functions on a box with the upper bounds which are found so far. Thereby, computing a lower bound can be done in different ways. For example, the natural interval extension F of a function f on a box X^* gives another box $F(X^*)$, which includes all function values. The lower left corner of this box, i. e., $\inf(F(X^*))$, is a lower bound of all function values $f(x)$ with $x \in X^*$. Since toolboxes for interval arithmetic like Intlab (for MATLAB) are well developed, this is a fast and simple method. In contrast to this, MOPBB uses the convex underestimators and their minimal values to get lower bounds. Even any other underestimator can be chosen like underestimators based on Lipschitz constants, see [EP13; ŽŽ16]. For those it is even possible to obtain minimal solutions analytically, and thus, no time-consuming single objective optimization problem has to be solved throughout the algorithm. In contrast to other B&B algorithm, MOPBB does not only determine singleton lower bounds by considering each objective function decoupled from the other ones. By applying

the approach of Benson's algorithm, the discarding test of MOPBB takes lower bounds described by hyperplanes into account.

Interval arithmetic can be used further. By evaluating the partial derivatives, a monotonicity test is developed in [FT09] for biobjective optimization problems. A box can be discarded, if both objective functions have the same monotonicity along the same direction in this box and the "best" points belong also to a boundary of another feasible box. Figure 4.6 shows this case: Both functions decrease on X^* while x_i increases. Thus, those points with $x_i = \overline{x}_i$ are the "best points" of X^* and, thereby, candidates for the efficient points. Since these points belong also to another box, namely X^{**}, the box X^* can be discarded.

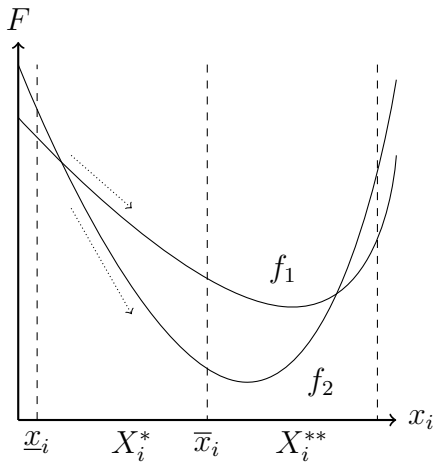


Figure 4.6. Monotonicity test, cf. [FT09].

Another type of discarding test is about using criteria based on first order conditions for optimality of multiobjective optimization problems. For instance, the authors of [GDC14] present three discarding tests of this type. These tests explore conditions such that there exist no multipliers $\lambda \in \mathbb{R}^m$, $\mu \in \mathbb{R}^p$ to satisfy the well-known Fritz-John or Karush-Kuhn-Tucker conditions for multiobjective problems for all points of a box. As a consequence, no point of the box fulfills a necessary condition to be efficient and the box can be discarded. Similar tests based on first order conditions were presented in [Sch12] as well, which also includes a generalization of the abovementioned monotonicity test to more than two objective functions. Interval arithmetic is here again a suitable tool to perform those type of discarding tests.

It is also possible to discard (parts of) boxes in the image space. For example, the authors of [Mar+16] deal with a tree-search structure based on boxes of $\mathbb{IR}^n \times \mathbb{IR}^2$. The overall discarding test consists of different steps: First, classic discarding tests are applied, for example, the monotonicity test by [FT09], or first order tests by [GDC14]. If a box could not be discarded directly, so-called *dominance decomposition* is applied. This method aims to contract boxes in the image space based on the upper bound set. Figure 4.7 illustrates different variants to contract the box Y into subboxes. A first variant is shown in the middle picture. The points $\hat{y}^1$ and $\hat{y}^2$ are the closest upper bounds to the image box Y which do not belong to Y. Then the box Y can be contracted to Y' because the remaining parts cannot contain any nondominated points. A second variant is based on one upper bound inside the box Y, see right picture of Figure 4.7. The part which is dominated by this upper bound can be excluded. Thus, Y is reduced to two (or three) new boxes $Y^l \cup Y^{lb}$ and Y^b. A very detailed variant is shown in the left picture: There, Y is contracted into 10 smaller subboxes, but this would lead to higher computational costs in the further procedure. Given a set of contracted subboxes of Y, the box of the preimage space can be narrowed as well. For that, classical contractors from the literature are applied to a *numerical constraint satisfaction problem*, [Mar+16].

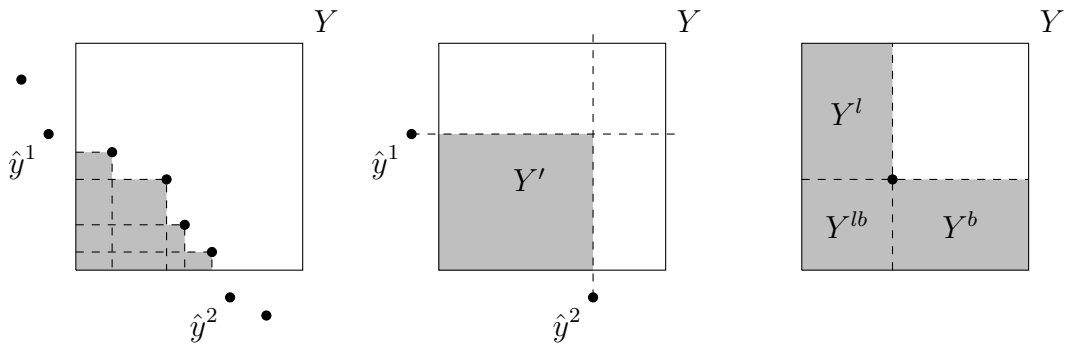


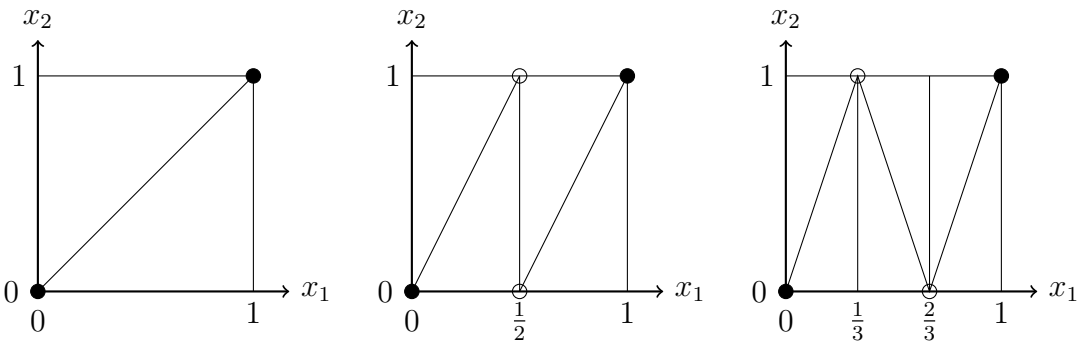
Figure 4.7. Dominance decomposition of a box in the image space, cf. [Mar+16].

4.5.1.2 Other Selection and Partition Rules

A selection rule to choose a box from the working list of a B&B algorithm is heuristically motivated, but also affects the whole algorithm. The aim of a selection rule should be to choose a box which delivers "good" upper bounds. As soon as the upper

bounds are very close to the nondominated set, the discarding tests are able to discard other boxes earlier. For that reason, a common rule is the one proposed in Section 4.2 which is used for MOPBB. Other algorithms introduce new distance measures to evaluate which box should be chosen next, see, for instance, [ACA19]. In [Mar+16], every box from the working list is stored with a normalized sum of objectives, i. e., the objectives are scaled such that they have the same range, and summarized afterwards. Then the box with a smallest normalized sum of objectives is selected as a next box for bisection.

Usually, a B&B bisects boxes into two subboxes. However, it is not necessary to partition a box into exactly two smaller boxes. The works by [CCC00b; CCC00a; CGC00] discuss so-called *multisection* rules theoretically and numerically. Another approach described by [ŽŽ16] is to trisect a chosen box, because the discarding test of their algorithm needs only the endpoints of a diagonal of the box. The advantage of trisection is that only two new end points have to be computed to get the three new diagonals, see Figure 4.8. For bisection also two new points are required, but this cannot be further improved. Thus, with the same amount of new required points the boxes can be reduced more.



a: Original box **b:** Bisection **c:** Trisection

Figure 4.8. Bisection and trisection of a box, solid points are the end points of the original diagonal, circles show the points which have to be computed to get new end points of diagonals, cf. [ŽŽ16].

Moreover, a partition rule specifies the direction along which a box gets partitioned. In MOPBB a box is bisected perpendicularly to the direction of maximum width which

is the case for many B&B algorithms. We will see in Section 6.3.1 that in the presence of integer variables some other partition rules are possible and useful.

4.5.1.3 Handling of Equality Constraints

A first biobjective B&B algorithm which is able to deal with equality constraints is proposed in [Mar+16]. The equality constraints are handled via so-called *interval constraint propagation*. Its aim is to find feasible points, i. e., solutions to a constraint system described by equality and inequality constraints. This can be used to further contract boxes of the preimage and image space. The authors propose to use already known *dominance constructors*, see [Mar+16, Subsection 3.3] for an overview of contractors and related literature. Note that in this thesis, we only consider multiobjective optimization problems with inequality constraints.

4.5.1.4 Convergence Rate

A definition for a convergence rate of multiobjective B&B algorithms was given in [Sch12] for so-called *multiobjective bounding operations*.

Definition 4.18 [Sch12, Def. 4.5] Let $X \subseteq \mathbb{R}^n$ be a box and consider a function $f\colon X \to \mathbb{R}^m$. A *multiobjective bounding operation* is a procedure to calculate for any subbox $X^* \subseteq X$ lower and upper bounds $LB_i(X^*),\ UB_i(X^*) \in \mathbb{R}$ such that

$$LB_i(X^*) \leq f_i(x) \leq UB_i(X^*)$$

for all $i = 1, \ldots, m$ and all $x \in X^*$. Moreover, with a $c \in X^*$ define

$$\begin{aligned} LB(X^*) &= (LB_1(X^*), \ldots, LB_m(X^*)), \\ UB(X^*) &= (UB_1(X^*), \ldots, UB_m(X^*)), \\ OV(X^*) &= (f_1(c), \ldots, f_m(c)). \end{aligned}$$

Formally, we obtain the multiobjective bounding operation

$$(LB(X^*), UB(X^*), OV(X^*)).$$

The bounding procedure of MOPBB does not fit into this definition completely. First, MOPBB only deals with lower bounds, which do not have to be singletons. For OV, MOPBB uses objective values as well. In this general framework of [Sch12], any feasible point of the box X^* can be chosen for c, for example, the midpoint or minimal solutions of some objective functions are possible. The definition of the convergence rate is the following.

Definition 4.19 [Sch12, Def. 4.6] Let $X \subseteq \mathbb{R}^n$ be a box and $f\colon X \to \mathbb{R}^m$. Furthermore, consider the multiobjective optimization problem

$$\min_{x \in X} f(x) = (f_1(x), \ldots, f_m(x))^T.$$

We say a multiobjective bounding operation $(LB(X^*), UB(X^*), OV(X^*))$ is *convergent* if there exist a fixed constant $C > 0$ such that for all boxes $X^* \subseteq X$, it holds

$$\|UB(X^*) - LB(X^*)\|_1 \leq C\omega(X^*).$$

Recall that $\omega(X^*)$ is the box width of the box X^*, see (3.1). Definition 4.19 is a generalization of the definition of the rate of convergence of 1 in single objective optimization given in [Sch12, Def. 2.4]:

Definition 4.20 [Sch12, Def. 2.2, 2.4] Let $X \subseteq \mathbb{R}^n$ be a box and $f\colon X \to \mathbb{R}$.

(i) A (single objective) bounding operation is a procedure to specify a feasible point $r(X^*) \in X^*$ and to calculate for any subbox $X^* \subseteq X$ a lower bound $LB(X^*) \in \mathbb{R}$ with

$$LB(X^*) \leq f(x) \text{ for all } x \in X^*.$$

(ii) Furthermore, consider the minimization problem

$$\min_{x \in X} f(x).$$

We say a bounding operation has the *rate of convergence* $p \in \mathbb{N}$ if there exist a fixed constant $C > 0$ such that for all boxes $X^* \subseteq X$, it holds

$$\|f(r(X^*) - LB(X^*))\|_1 \leq C\omega(X^*)^p.$$

For example, using an αBB-underestimator from Section 3.2 for the (scalar-valued) objective f leads to the classical αBB method introduced in [Adj+98]. In fact, $LB(X^*)$ is the minimal value of the underestimator on X^*, and $r(X^*)$ can be chosen as the midpoint of X^* or the minimal solution obtained from the minimization of the underestimator. With this setting and because of Remark 3.13, this bounding operation has a rate of convergence of 2.

Currently, for multiobjective bounding procedures there have been defined no further rates yet. For MOPBB it is not possible to show whether the used multiobjective bounding procedure is convergent in the sense of Definition 4.19. There are two reasons for that: first, we do not compute upper bounds $UB_i(X^*)$, and second, the lower bounds are in general not singletons. However, it is possible to get an upper bound $UB_i(X^*)$, for example, by maximizing a concave overestimator for each objective (see Section 7.2.1). Then $UB(X^*)$ is given by the anti-ideal point (see (3.6)) of the concave overestimator on X^*. Together with the ideal point for $LB(X^*)$ this multiobjective bounding operation is convergent, which can be shown by using Remark 3.13. In fact, the procedure for MOPBB described in Section 4.1.1 is at least as good as using only the ideal point of the underestimators, because we improve this by supporting hyperplanes to get an outer approximation as a lower bound.

4.5.2 Heuristic Algorithms

There are plenty of heuristically motivated algorithms which aim to find globally efficient solutions of multiobjective optimization problems. Here, we mention two of them. The first uses a *direct search approach*. The second is the most frequently used evolutionary algorithm for multiobjective problems — NSGA-II. Recently, there are (heuristical) improvements for NSGA-II, for example, [DJ12; Han+14]. In this work, we focus on the basic version.

MultiGLODS A first heuristic algorithm is MultiGLODS (Multiobjective Global and Local Optimization using Direct Search) which was introduced in [CM18]. It is based on the direct search approach. Thus, it does not require any information about derivatives. However, to prove some convergence results, this method requires certain kinds of differentiability. The single objective variant was developed at first and is known under the name of GLODS (Global and Local Optimization using Direct Search). Because of direct search and under some certain assumptions, the algorithm finds sequences of feasible points converging to *Pareto-Clarke critical points*, see [CM18, Def. 6] for a definition. By using a multistart strategy, the procedure aims for globally efficient points. MultiGLODS works by alternating between new searches, multistarts and exploring subregions of the feasible set. Hence, this procedure seeks sets of globally and locally nondominated points. Constraints are handled by using an extreme barrier approach, i. e., the function values of infeasible points are set to ∞. This is possible since no derivatives are needed.

MultiGLODS and MOPBB differ significantly. First, MultiGLODS is basically a local solver which is also able to obtain globally efficient and nondominated points by using multistarts. However, there is no guarantee of getting globally efficient points. MOPBB works with a B&B approach and tries to discard subregions of the feasible set as early as possible. Most of all, it guarantees a certain accuracy of the computed points.

NSGA-II A popular algorithm which aims for globally efficient solutions and nondominated points of multiobjective optimization problems is the evolutionary algorithm NSGA-II (Nondominated Sorting Genetic Algorithm-II), see [Deb+02a] or [Deb01, Section 6.2]. Evolutionary algorithms work with heuristic strategies to generate "good solutions" randomly. In what follows, NSGA-II is explained in detail because it will be compared numerically with MOPBB in Section 5.3. Words like *population* and *generation* are very common in evolutionary algorithms and are explained and used here as well.

The procedure of the t-th iteration of NSGA-II is the following: For a given set of feasible points P_t — the current *population* — with a fixed size N, another new population is generated. For this, some common operations of evolutionary algorithms like selection, recombination and mutation are applied. Now a population $P_t \cup Q_t$ of

the size $2N$ is present and is sorted by a fast *nondominating sorting algorithm*: The exact sorting procedure is one of the key features of NSGA-II and sorts the population into so-called *nondominated fronts*. The points which are not dominated by any other point of the population belong to the first nondominated front F_1. The nondominated points of the finite set of remaining points are identified as well and build the second nondominated front F_2. The procedure continues and adds the nondominated fronts successively to the new population P_{t+1} until at least N points were added. If more than N points are found, the last nondominated front is considered separately. There, a measure, called *crowding distance*, is assigned to every point of this set. The crowding distance is based on some certain distances in the objective space to estimate the density of points surrounding another point of the same nondominated front. A lower crowding distance for one point means that this point is more crowded by other found image points. After the assignment of the crowding distance, the points with the highest crowding distance are selected until the new population consists of exactly N points.

Different parameters can be chosen by the user. The most common are the population size N, and the total number of iterations, also known as *generations*. Moreover, it is possible to set some internal parameters like selection strategies, crossover and mutation functions as well as a crossover fraction. The crossover fraction is the ratio of new feasible points which are created by the crossover function, [MAT18b]: A crossover fraction of 1 means that only crossover is performed without any mutations. Otherwise, in case of a crossover fraction of 0, there is no crossover, only mutation.

NSGA-II and MOPBB cannot be compared theoretically because both are based on significantly different approaches. MOPBB is deterministic and gives a guarantee to compute an (ε, δ)-efficient set, while NSGA-II works heuristically and cannot guarantee any accuracies of the computed solutions. It is possible to get only locally efficient points, which are not globally efficient. By variation of parameters and performing multiple runs globally efficient points can be found, but a guarantee of that is not given.

4.6 Conclusions

In this chapter, we have combined the ideas of convex underestimators with techniques from multiobjective convex optimization and the idea of local upper bounds from multiobjective combinatorial optimization to obtain an efficient discarding test. In contrast to other global multiobjective optimization algorithms our algorithm guarantees the (ε, δ)-efficiency of the output $\mathcal{A}$ in a finite time. Because of this, the introduced algorithm is not comparable with existing state-of-the-art algorithms which are usually heuristics and do not provide accuracy of their computed solutions. Numerical experiments will be presented in the next chapter. They will show that good results can be obtained even with large values for ε and δ. This is due to the fact that the estimations in some proofs are quite rough.

The current implementation does not use other, simpler, discarding tests as already proposed in the literature. Therefore, further time savings can be expected if these existing discarding tests are used additionally. An example of these tests are the monotonicity tests as proposed in [FT09; Sch12]. Moreover, in [ŽŽ16] bounds based on Lipschitz constants are derived and it would be of interest to explore possible combinations. Also the handling of nonconvex constraints should be considered. However, the same difficulties as for global single objective optimization algorithms will arise. These difficulties are discussed, for instance, in [KSS15].

5 Numerical Results for MOPBB

In this chapter, we test the newly developed algorithm MOPBB on some chosen test problems and on a design problem, which occurs in engineering. This gives a brief overview on different aspects of the algorithm and illustrates that it runs satisfyingly. The test results and their interpretation can be found in Section 5.1

Moreover, we compare MOPBB with the state-of-the-art evolutionary algorithm for multiobjective optimization problems called NSGA-II, explained in Section 4.5.2. For that, in Section 5.3, performance indicators are introduced briefly. Then we present and discuss the results for some test instances.

MOPBB has been implemented in MATLAB R2018a [MAT18a] and uses the optimization toolbox [MAT18c] as well as the toolbox Intlab [Rum99] for interval arithmetic. The experiments of Sections 5.1 and 5.2 have been done on a computer with Intel(R) Core(TM) i3-2015 CPU and 16 Gbytes RAM on operation system Windows 7 Professional. The experiments of Section 5.3 have been done on a computer with Intel(R) Core(TM) i5-7400T CPU and 16 Gbytes RAM on operating system Windows 10 Enterprise.

5.1 Numerical Results for some Test Instances

In this section, we test algorithm MOPBB on some instances from the literature. First, we compare two approaches for a possible discarding test. In fact, we test cases (II) and (III) from the beginning of Section 4.1.1. Recall that (II) uses the ideal point of the convex underestimators to obtain lower bounds and compares them with $\mathcal{L}_{PNS}$ only. Case (III) implements the new discarding test. For a better comparability, we restrict

our algorithm to the first `while`-loop with a termination rule $\omega(X^l) < \delta$. Therefore, there is no dependency on ε in this test run.

Test instance T.1 [FF] This test instance is based on [FF95] and the dimension of the preimage space $n \in \mathbb{N}$ can be chosen arbitrarily.

$$f_1(x) = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right),\ f_2(x) = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right)$$

with $-2 \leq x_i \leq 2$ for all $i = 1, \ldots, n$. It should be mentioned that in [FF95] the domain was originally given by $-4 \leq x_i \leq 4$ for all $i = 1, \ldots, n$.

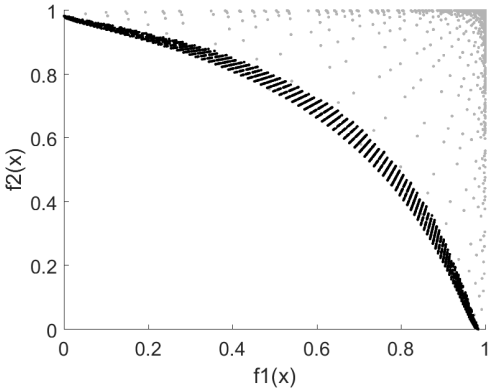
For the number of iterations, the computation time t, and the number of boxes in the solution list $\mathcal{L}_S$, we obtain the results presented in Table 5.1.

Table 5.1. Results for Test instance T.1 with $\delta = 0.1$.

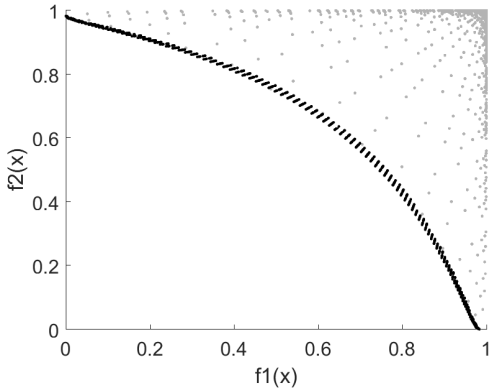
	case (II) only ideal point			case (III) new discarding test		
n	# iterations	t [s]	$\lvert\mathcal{L}_S\rvert$	# iterations	t [s]	$\lvert\mathcal{L}_S\rvert$
1	41	3.60	34	41	4.19	34
2	456	39.43	262	359	38.14	210
3	6283	626.73	3434	3055	364.11	1268
4	78965	10014.12	42540	20966	2958.71	7644

The plots in Figure 5.1 show the results in the image space for $n = 3$. The image set is represented by some image points in gray which are obtained by a discretization of the feasible set X. The black points are the images of the midpoints of the boxes of the list $\mathcal{L}_S$.

It can be seen that we can decrease the number of iterations with the new approach (III), which is clear, because computing the ideal point by minimizing convex underestimators is also a part of (III). Additionally, for $n \geq 2$ approach (III) is faster even though we have to solve more optimization problems on each subbox. In the $n = 3$ case the approximation of the nondominated set obtained with the new discarding test is much tighter than the one which only uses the ideal point. This can be seen in Figure 5.1. In the other cases for n we obtained similar results.



a: Image set with discarding test based on case (II).



b: Image set with discarding test based on case (III).

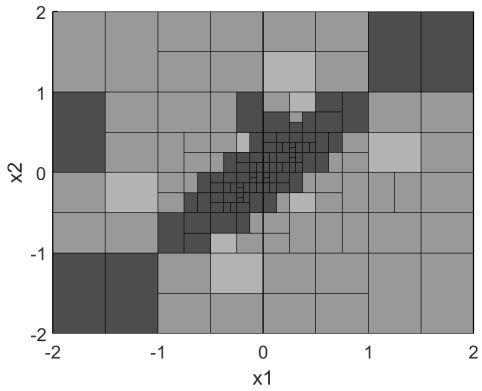
Figure 5.1. Test instance T.1 with $n = 3, \delta = 0.1$

To illustrate MOPBB with the new discarding test and with all three `while`-loops, we choose $\varepsilon = 0.05$ and $\delta = 0.1$. The plots in Figures 5.2a and 5.2b show the partitioning of the feasible set after the second and third `while`-loops. The different shades of gray indicates in which `while`-loop a box was discarded or whether it still belongs to a solution list: Medium gray boxes are those which have been discarded in the first `while`-loop, the light gray boxes were discarded in the second `while`-loop. Furthermore, the new light gray boxes compared to Figure 5.2a were discarded within the third `while`-loop. The dark gray boxes were not discarded after the second and third `while`-loops. In Figure 5.2c the boxes of $\mathcal{L}_{S,3}$ are shown together with some black points which are the points from the (ε, δ)-efficient set $\mathcal{A}$. Figure 5.2d shows the image set of the test function. The black points are the images of the approximation set $\mathcal{A}$. Additionally, in Figures 5.2e and 5.2f the obtained set T is shown.

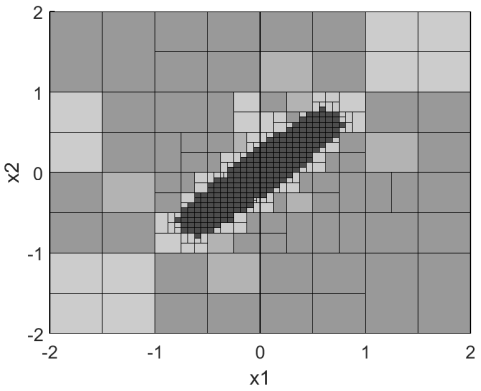
The results of the next test instance show that our new algorithm is also able to find globally efficient and nondominated points if there are also locally efficient points which are not globally efficient.

Test instance T.2 [Deb41] This test instance was proposed in [Deb99]:

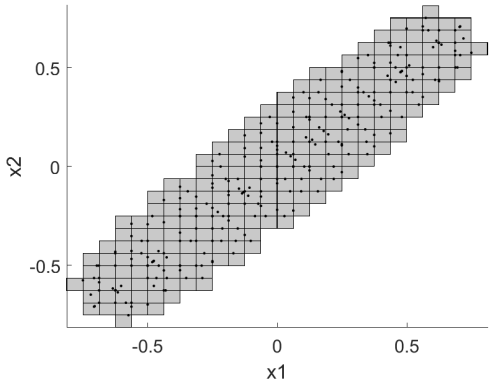
$$f_1(x) = x_1,$$
$$f_2(x) = \frac{1}{x_1}\left(2 - \exp\left(-\left(\frac{x_2 - 0.2}{0.004}\right)^2\right) - 0.8\exp\left(-\left(\frac{x_2 - 0.6}{0.4}\right)^2\right)\right)$$



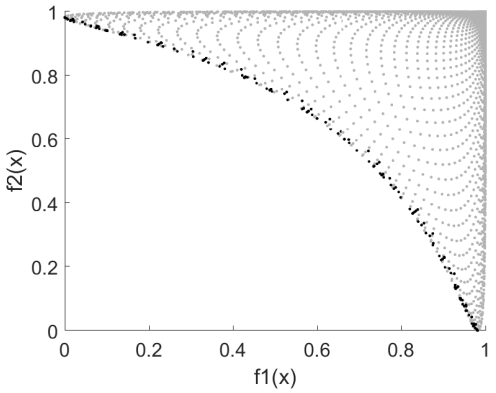
a: Partition of the feasible set after second `while`-loop, 66 discarded boxes.



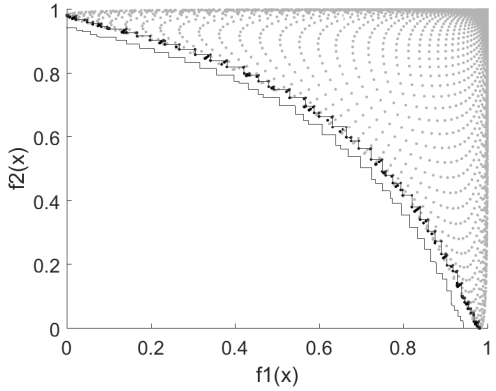
b: Partition of the feasible set after third `while`-loop, 151 discarded boxes.



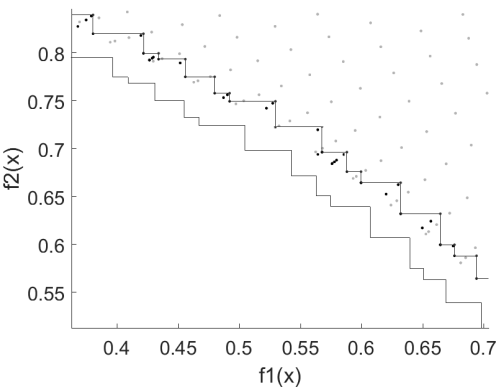
c: 222 boxes of the solution list $\mathcal{L}_{S,3}$, 279 points in $\mathcal{A}$.



d: Image set and images of the (ε, δ)-efficient set $\mathcal{A}$.



e: Image set, images of the (ε, δ)-efficient sets A and T.

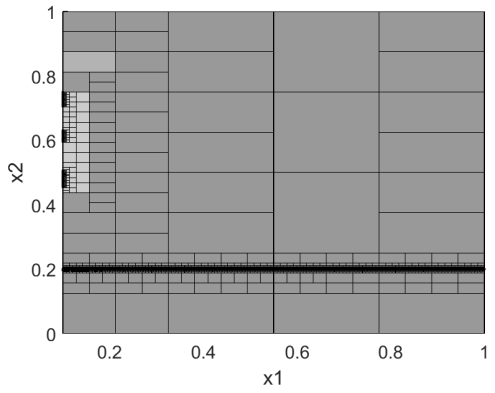


f: Part of Figure 5.2e magnified.

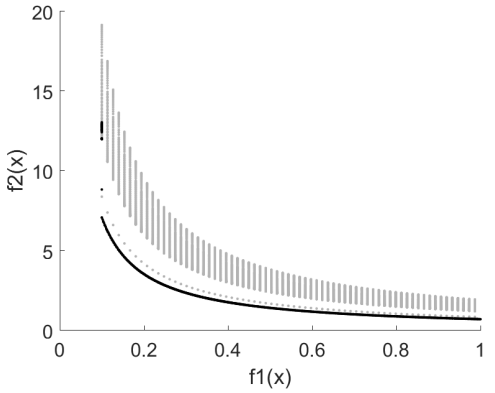
Figure 5.2. Test instance T.1 with $n = 2,\ \varepsilon = 0.05$ and $\delta = 0.1$

with $0.1 \leq x_1 \leq 1,\ 0 \leq x_2 \leq 1$. The globally efficient points are $(\tilde{x}_1, \tilde{x}_2)$ with $\tilde{x}_2 \approx 0.2$ and $\tilde{x}_1 \in [0.1, 1]$. This test instance has also locally (non globally) efficient points with $\tilde{x}_2 \approx 0.6$ and $\tilde{x}_1 \in [0.1, 1]$.

Figure 5.3 shows the results of our algorithm on this test instance. In total 648 ε-efficient points are found, where 608 are in a δ-neighborhood of a globally efficient point. The other 40 points are the ones with $x_1 \approx 0.1$ and $x_2 \not\approx 0.2$ whose images are located on the right boundary of the image set above the nondominated set. Even while those points are not close to the efficient set, they are still ε-efficient. Recall that the images of these ε-efficient points are visualized as black points in Figure 5.3b. It can be seen that `MOPBB` computed an approximation of the set of globally efficient points and did not aim for locally (non globally) efficient points. For example, a weighted sum approach with a local optimization solver such as sequential quadratic programming (SQP) may be able to find the locally efficient points and does not ensure to find globally efficient points.



a: Partition of the feasible set after third `while`-loop, 2039 discarded boxes, 648 points in $\mathcal{A}$.



b: Image set and images of the (ε, δ)-efficient set $\mathcal{A}$.

Figure 5.3. Test instance T.2 with $\varepsilon = 0.01$ and $\delta = 0.01$

The next test instance has three objective functions. Moreover, we added a convex constraint g.

Test instance T.3 [ViennetCon] This test instance (without the convex constraint) was introduced in [VFM96]:

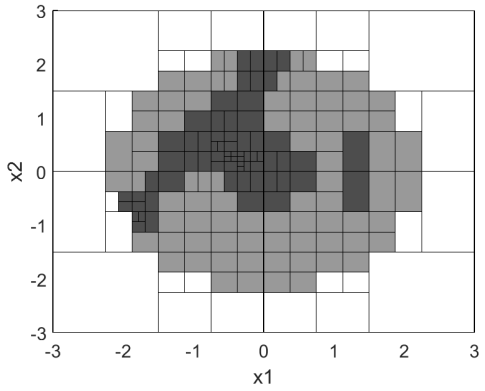
$$
\begin{aligned}
f_1(x) &= 0.5(x_1^2 + x_2^2)^2 + \sin(x_1^1 + x_2^2) \\
f_2(x) &= \frac{(3x_1 - 2x_2 + 4)^2}{8} + \frac{(x_1 - x_2 + 1)^2}{27} + 15 \\
f_3(x) &= \frac{1}{x_1^2 - x_2^2 - 1} - 1.1 \exp(-x_1^2 - x_2^2)
\end{aligned}
$$

with $-3 \leq x_i \leq 3,\ i = 1, 2$ and in addition $g(x) = x_1^2 + x_2^2 - 4$.

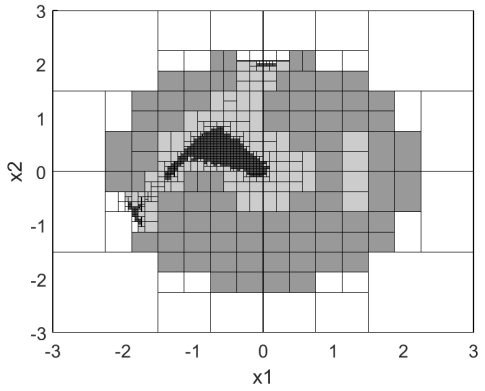
Figure 5.4 shows the results of this test instance. In addition to the meanings of gray shades in the former graphical results, the white boxes are ones which were discarded because of infeasibility. In this example no box was discarded in the second `while`-loop. Figure 5.4c shows the boxes of the solution list $\mathcal{L}_{S,3}$ and the computed ε-efficient points. Moreover, we can see in Figures 5.4b and 5.4d that, in the case of disconnected areas of efficient or nondominated points, our algorithm is also able to find all these areas.

5.2 Application in Lorentz Force Velocimetry

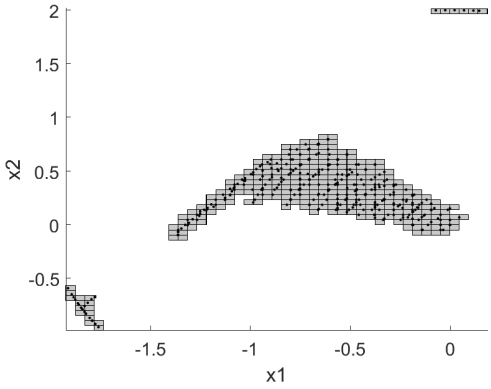
We have applied MOPBB to a problem which arises in the optimization of a measure technique known as Lorentz force velocimetry (LFV). In LFV the aim is to measure, for example, the velocity of fluids. The technique is based on measures of the Lorentz force that occurs due to the flow of an electric conductive fluid under the influence of a variable magnetic field. The following setting was also considered in [BTE18]. For generating a measure system in our problem setting n dipoles have to be arranged around a cylinder. An electric conductive fluid will flow through this cylinder during the measures. Figure 5.5 shows a sketch of the setting with one dipole. We assume that the dipoles are placed at equidistant positions $r^i = (0, \cos\gamma_i, \sin\gamma_i)^T,\ i = 1, \ldots, n$, $\gamma_i = (i-1)\frac{2\pi}{n}$ fixed, around the cylinder and aim to find the optimal magnetic orientation. We assume that all dipole moments have the same magnitude m. The orientation of an individual dipole i is represented in terms of polar and azimuthal angles θ_i and



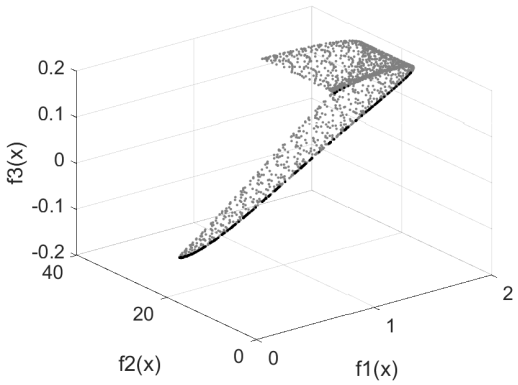
a: Partition of the feasible set after second `while`-loop, 108 discarded boxes.



b: Partition of the feasible set after third `while`-loop, 315 discarded boxes, 360 points in $\mathcal{A}$.



c: 299 boxes of the solution list $\mathcal{L}_{S,3}$.



d: Image set and images of the (ε, δ)-efficient set $\mathcal{A}$.

Figure 5.4. Test instance T.3 with $\varepsilon = 0.1$ and $\delta = 0.1$

φ_i, where φ_i is split into the two angles β_i and fixed γ_i. Thus, the magnetic moment vector m^i is $m^i(\theta_i, \beta_i) = (m\cos\theta_i, m\sin\theta_i\cos(\beta_i+\gamma_i), m\sin\theta_i\sin(\beta_i+\gamma_i))$.

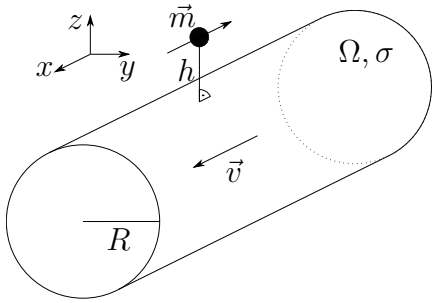


Figure 5.5. Sketch of the problem geometry, taken from [BTE18].

The first objective is to maximize the absolute value of the axial force component as in [BTE18]. Since the dipoles are on a circle of radius H in the plane $x = 0$, the force can be expressed analytically. The self-interaction term for a dipole i and the mutual interaction term between dipoles i and j are

$$F_s(\beta_i, \theta_i) = \frac{45\pi}{4096} \cdot \frac{v\sigma R^4 \mu_0^2 m^2}{128\pi H^7}[355 + 25\cos(2\theta_i) + 266\cos(2\beta_i)\sin^2\theta_i] \text{ and}$$

$$\begin{aligned} F_m(\beta_i, \beta_j, \theta_i, \theta_j) = \; & \tfrac{45\pi}{1024} \cdot \tfrac{v\sigma R^4\mu_0^2 m^2}{128\pi H^7}[10(5 + 14\cos(\gamma_i - \gamma_j))\cos\theta_i\cos\theta_j \\ & + \sin\theta_i\sin\theta_j\,(49\cos(\gamma_i - \gamma_j - \beta_i - \beta_j) + 35\cos(\beta_i - \beta_j) \\ & + 25\cos(\gamma_i - \gamma_j + \beta_i - \beta_j) + 105\cos(\gamma_i - \gamma_j - \beta_i + \beta_j) \\ & + 35\cos(\beta_i + \beta_j) + 49\cos(\gamma_i - \gamma_j + \beta_i + \beta_j))], \end{aligned}$$

respectively. For the resulting interaction force, we obtain

$$F_x(\beta, \theta) = \sum_{i=1}^{n} F_s(\beta_i, \theta_i) + \sum_{i<j} F_m(\beta_i, \beta_j, \theta_i, \theta_j).$$

The constants are the velocity of the fluid v, the electric conductivity σ, the radius of the cylinder R and the vacuum permeability $\mu_0 = 4\pi \cdot 10^{-7}$Vs/Am.

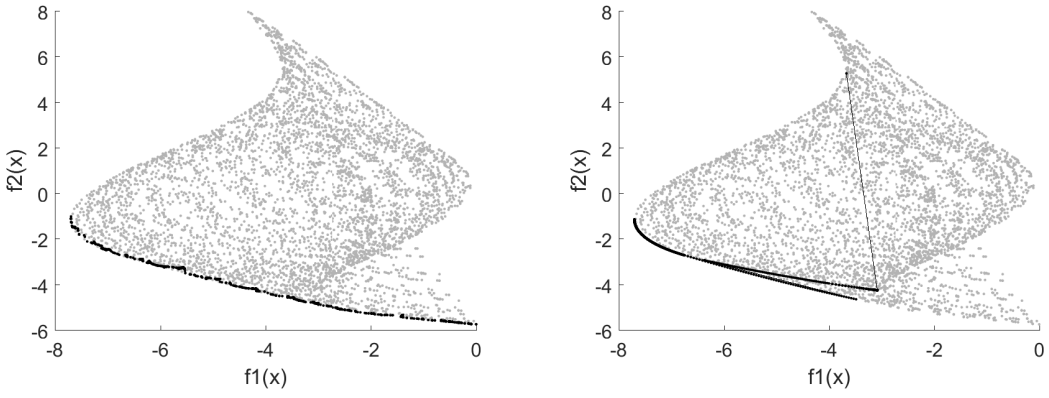
The second goal is the minimization of the interaction potential energy between the dipoles, because arrangements with a high interaction potential energy are more difficult to realize. The vector $r_{i,j}$ represents the vector between the positions of both

dipoles: $r_{i,j} = r^j - r^i$. The function is the sum of all energies between every pair of dipoles:

$$V(\beta, \theta) = \sum_{i<j} \frac{\mu_0}{4\pi |r_{i,j}|^5} [|r_{i,j}|^2 m_i(\theta_i, \beta_i) m_j(\theta_j, \beta_j) - 3(m_i(\theta_i, \beta_i) r_{i,j})(m_j(\theta_j, \beta_j) r_{i,j})].$$

To reduce the dimension of the preimage space, we fix $\theta_i = \frac{\pi}{2}$ for all $i = 1, \dots n$. Because of this and the symmetry of the arrangements, the feasible set of interesting angles β is given by $X = [(-\pi/2, -\pi, \dots, -\pi)^T, (\pi/2, \pi, \dots, \pi)^T] \in \mathbb{IR}^n$. All constant coefficients are set to 1 and we scale the objective functions by -0.1 (to switch from maximization to minimization) and 100, respectively, to obtain different accuracies for both objective functions.

Figure 5.6a shows the image set after executing the algorithm with $\varepsilon = \delta = 0.5$ for three dipoles. The results after using a weighted sum approach with a standard SQP solver for the scalarized problems are shown in Figure 5.6b. The black points connected by black lines are the minimal solutions of minimizing a weighted sum of the objectives and are used as the starting point for the next weighted sum. Obviously, this approach was able to find locally efficient points first and after some iterations some of the globally efficient points. Moreover, the lower-right part of the nondominated set was not found. Even for different starting points, we never obtained an approximation of the whole nondominated set, as was achieved with our algorithm.



a: Image set after MOPBB.

b: Image set after a weighted sum approach with a random starting point.

Figure 5.6. Graphical results for 3 dipoles on a circle around a cylinder.

5.3 Numerical Comparison with NSGA-II

The topic of this section is the numerical comparison of MOPBB with the evolutionary algorithm NSGA-II. This algorithm was introduced and described in Section 4.5.2 and is a popular algorithm for multiobjective optimization problems. However, it works heuristically and cannot guarantee a certain accuracy after a finite amount of time. In contrast to this, the new algorithm MOPBB ensures an (ε, δ)-efficient set, after finite time, see Lemmas 4.5 to 4.7 and Theorem 4.17. Therefore, the comparison of both algorithms is done by test runs of MOPBB, and test runs of NSGA-II depending on the results of MOPBB.

To perform NSGA-II in order to compare it with the algorithm MOPBB, we use the function `gamultiobj` from the global optimization toolbox of MATLAB [MAT18b]. This function is a variant of NSGA-II. In particular, it executes a controlled version of NSGA-II, see [DG01]. This means that the number of points in the first nondominated front is restricted as well as in the other nondominated fronts [Deb01]. An advantage of the controlled version is that the total number of nondominated fronts is retained in order to keep some variance and diversity in the found population. The controlling part is done via a positive scalar from 0 to 1, which specifies the fraction of points belonging to the first nondominated front.

The algorithm `gamultiobj` calculates an internal measure which indicates the movement of the found nondominated set in dependence of the crowding distance. If the change of this value is less than a predefined tolerance for a predefined number of generations, the algorithm `gamultiobj` stops.

In the following, we introduce three performance indicators, which enable the comparison of algorithms. Then, the test instances are introduced and the settings for both algorithms are stated. In the remaining subsection, we present the numerical results and discuss these.

5.3.1 Performance Indicators

There are many performance indicators known for comparing outputs of multiobjective algorithms, see [Aud+18] for an extensive overview. Usually, two sets, which both approximate the nondominated set, are compared by considering a certain measure. Some of the performance indicators need the information of the nondominated set.

First, to apply a measure to the output $\mathcal{A}$ of MOPBB we have to adapt the output $\mathcal{A}$: One requirement is that the approximate set in the objective space has to be stable. The set $\mathcal{A}$ is a subset of the preimage space. Therefore, the function values of all points of $\mathcal{A}$ are calculated and those which are dominated by other points of $f(\mathcal{A})$ are removed. We denote the new (stable) set of points of the objective space by $\mathcal{F}$.

Hypervolume The hypervolume (or S-metric) is the volume of the area which is dominated by an approximation of the nondominated set. For the calculation a reference point is required which is greater than all objective values of the efficient points. Choosing a good reference point can already be a difficult task. For example, this can be the nadir point, which is defined by the maximum of each objective function over all points from X_E, see Remark 3.17. Computing the nadir point is not simple as the whole efficient set has to be known. Thus, the anti-ideal point of f on X or any other upper bound can be a suitable choice. For simplicity, we use $\sup(F(X))$ here, i. e., the upper right vertex of the image box which is obtained by the natural interval extension. Within the numerical tests, it appears that this is not a good choice in each case. Thus, for some special instances, where the hypervolume was too large to compare the algorithms properly, we chose a reference point by hand. Note that the reference point has to be the same for each test instance to compare two algorithms.

There are different algorithms for computing the hypervolume [Bra11]: The inclusion-exclusion principle is one of the most simple methods. However, this method is not very time-efficient as it is exponential in the number of points. Other algorithms, which identify disjoint boxes which contribute to the hypervolume, work in exponential time according to the numbers of objectives. There are also algorithms which

were developed for specific dimensions of the objective space. In this thesis, we compute the hypervolume only for biobjective problems. In this case, this indicator can be obtained by sorting the points of the approximation of the nondominated set and by adding the volume of rectangular slices, see Figure 5.7 for an illustration. The higher the hypervolume, the better the computed set approximates the nondominated set.

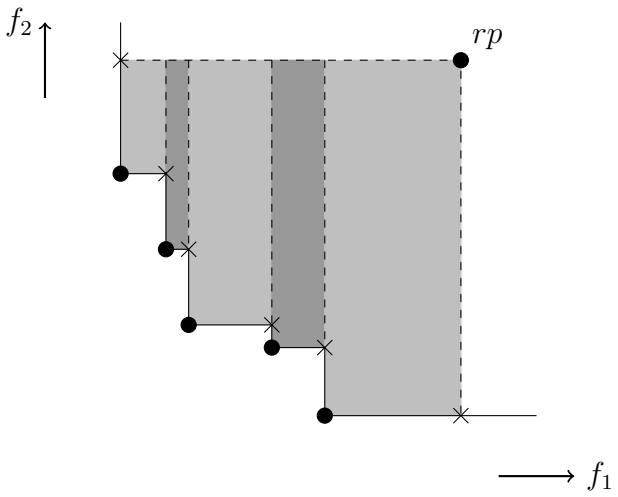


Figure 5.7. Computation of the hypervolume for $m = 2$ by adding the volume of each rectangle. rp is the reference point.

Spacing Metric The spacing metric was proposed in [Sch95, Sect. 13.2]. For a stable set $\mathcal{F} = \{y_1, \ldots, y_{|\mathcal{F}|}\}$, it can be computed by

$$SP(\mathcal{F}) = \sqrt{\frac{1}{|\mathcal{F}| - 1} \sum_{i=1}^{|\mathcal{F}|} (d_i - \overline{d})^2},$$

where $d_i = \min_{y_j \in \mathcal{F}, j \neq i} \|y_i - y_j\|$, i. e., the minimal distance from the i-th point of the approximation $\mathcal{F}$ of the nondominated set to another point of $\mathcal{F}$. The value $\overline{d}$ is the mean of all d_i. Originally, the distance described by d_i was chosen to be the l_1 distance. As far as the Euclidean distance was only used in this work, we will use it here as well. Here, a smaller spacing metric more suitable, because this means that the approximation is more uniformly distributed than the one with a larger spacing metric.

Generational Distance The generational distance was introduced in [Van99, Sect. 6.3.4.2] and requires the knowledge of the nondominated set. For simplicity, let $\mathcal{P}$ be a discrete representation of the nondominated set $f(X_E)$. Then this indicator is computed by

$$GD_p(\mathcal{F}, \mathcal{P}) = \frac{1}{|\mathcal{F}|} \left(\sum_{s \in \mathcal{F}} \min_{r \in \mathcal{P}} \|s - r\|^p \right)^{\frac{1}{p}}.$$

It measures the mean of the distance of the approximation $\mathcal{F}$ to the real nondominated set in dependence of $p > 0$. We use here $p = 2$. The method which produces a small generational distance is the preferable one. If two algorithms should be compared by the generational distance, it is recommended to ensure that both found approximations have an almost equal cardinality. The reason for this is, for example, that an approximation which consists of one point of the true nondominated set has a generational distance of 0. On the other hand, an approximation with many well-distributed points close to the nondominated set has a generational distance greater than 0 although it approximates the true nondominated set much better.

5.3.2 New Test Instances and Settings

Table 5.2 gives an overview of the chosen test instances. For these, we use common abbreviations from the literature as well as an internal numbering. The description of Test instance T.4 to Test instance T.22 can be found in Appendix A. All others (Test instance T.1 to Test instance T.3) are already stated in Section 5.1. In Table 5.2, "s" means that the instance is scalable according to the number of variables or objectives. The sixth column states whether an analytical form of the efficient set X_E or of the nondominated set $f(X_E)$ is known. The next column describes whether there are constraints other than box constraints. The last column describes the geometry of the nondominated set. Note that "convex" means that the upper image, i. e., $f(M) + \mathbb{R}^m_+ \left(= f(X_E) + \mathbb{R}^m_+\right)$, is convex. "Concave" means that $f(X_E) - \mathbb{R}^m_+$ is convex.

The geometry of the nondominated set of DTLZ5 and DTLZ6 is not completely known. The authors of [Deb+02b] claim that the set is degenerated.For example, a nondominated set in the three-dimensional space with a lower dimension than two (e.g., a

Table 5.2. Overview about chosen test instances.

reference	name	T.X	n	m	X_E	constraints	geometry
[Deb+02b]	DTLZ1	T.4	s	s	yes	no	linear
	DTLZ2	T.5	s	s	yes	no	concave
	DTLZ3	T.6	s	s	yes	no	concave
	DTLZ4	T.7	s	s	yes	no	concave
	DTLZ5	T.8	s	s	(yes)[1]	no	(concave)[1]
	DTLZ6	T.9	s	s	(yes)[1]	no	(concave)[1,2]
	DTLZ7	T.10	s	s	yes	no	disconnected
[ZDT00]	ZDT1	T.11	s	2	yes	no	convex
	ZDT2	T.12	s	2	yes	no	concave
	ZDT4	T.13	s	2	yes	yes	convex[2]
[FF95]	FF	T.1	s	2	yes	no	concave
[Deb99]	Deb41	T.2	2	2	no[3]	no	convex[2]
	Deb41Con	T.14	2	2	no	yes[4]	convex[2]
	Deb513	T.15	2	2	no	no	disconnected
	Deb521a	T.16	2	2	no	no	concave[2]
	Deb521b	T.17	2	2	no	no	concave
[VFM96]	Viennet	T.18	2	3	no	no	disconnected
	ViennetCon	T.3	2	3	no	yes[4]	disconnected
[Deb01]	SRN	T.19	2	2	yes	yes	convex
[PMC96]	Pol	T.20	2	2	no	no	disconnected
[Far02]	Far1	T.21	2	2	no	no	mixed[5]
[KW05]	KW2	T.22	2	2	no	no	disconnected

[1] only known for $m = 2$ and $m = 3$; more complicated for higher dimensions
[2] locally (but not globally) nondominated sets possible
[3] globally efficient set only roughly known
[4] constraints added by this work's author
[5] the nondominated set is connected, but consists of convex and nonconvex parts

curve or a line) is degenerated. Studies of [Hub+06] showed that this is not true for instances with more than three objective functions. The instances ZDT3, ZDT5 and ZDT6 (which belong to the ZDT test suite) are not used in this work, because the variables for ZDT5 are binary, and ZDT3 and ZDT6 are not differentiable in all points of the preimage space, especially not in the efficient points. The instance Far1 has some typographic mistakes in the original paper by [Far02]. Thus, we used the corrected version from [Hub+06]. Instance Deb521a is not twice continuously differentiable in the suggested interval. However, we adapt the box constraints to exclude the points in which the objective function is not differentiable.

The following experiments have been done with MATLAB R2018a [MAT18a] on a computer with Intel(R) Core(TM) i5-7400T CPU and 16 Gbytes RAM on operating system Windows 10 Enterprise. For the numerical testing, MOPBB is performed for maximal six hours (21600 seconds). Depending on the number of points in $\mathcal{F}$ and calls of the discarding test, the population size and the maximal number of generations for NSGA-II are chosen. If MOPBB did not deliver any results within the time limit, `gamultiobj` was also not executed.

The settings for NSGA-II are stated next. Thereby, DT is the number of executed discarding tests by MOPBB (in the first and third `while`-loop), and $|\mathcal{F}|$ the number of points of the stable set $\mathcal{F}$ obtained by MOPBB. Thus, we used the following:

- `PopulationSize` $= [|\mathcal{F}|/$`ParetoFraction`$]$
- `Generations` $=$ DT
- `TimeLimit` $= 21600$
- `TolFun` $= 10^{-4}$ (or $= 10^{-5}$ in some special cases)
- all others by default:
  - `CrossoverFcn` $=$ `'crossoverintermediate'`
  - `CrossoverFraction` $= 0.8$
  - `MaxStallGenerations` $= 100$
  - `ParetoFraction` $= 0.35$
  - `SelectionFcn` $=$ `'selectiontournament'`

`TolFun` and `MaxStallGenerations` are parameters to stop `gamultiobj` before the maximum number of generations is reached. As mentioned at the beginning of this subsection, `gamultiobj` stops if an internal measure does not change significantly. More precisely and with the default parameters, `gamultiobj` stops if the changes of this measure are below 10^{-4} (`TolFun`) for 100 generations (`MaxStallGenerations`). The parameter `ParetoFraction` gives the percentage of points in the first front according to the size of the whole population. The other default parameters are important for creating a new population from a former one.

The algorithm `gamultiobj` is run for ten times since it is an evolutionary algorithm and produces different populations in each run. In Section 5.3.3, we present the mean of the chosen performance indicators as this is an estimation of values which can be expected. We also discuss a test instance for which the performance indicators differ significantly. There are some instances for which NSGA-II was faster than MOPBB but delivered worse values for at least one performance indicator. To give NSGA-II a chance to improve, we run it again for ten times for these instances with `TolFun` $= 10^{-5}$ and added those results as well.

Tables 5.3 and 5.4 show the additional settings of the scalable test instances as well as analytical descriptions of the nondominated set. For the non-scalable instances the values of ε and δ are mentioned within Table 5.8. The chosen values for ε and δ were set after separate test runs in order to find reasonable computational times and approximations of the nondominated set. The tables list only those parameters for which MOPBB got results within the time limit.

Since the nondominated set of Deb41 is not known more precisely, this instance will be handled like an instance with unknown nondominated set. Moreover, for SRN [Deb01] suggests an analytical description of the efficient and, thus, of the nondominated set. But this description is not complete as it considers not all efficient points.

Table 5.3. Settings for the DTLZ-instances which are scalable in n and m with a known nondominated set.

name	n	m	ε	δ	nondominated set $Y = f(X_E)$
DTLZ1	2	2	0.1	0.1	$Y = \{y \in \mathbb{R}^2 \mid y_1 \in [0, 0.5], y_2 = 0.5 - y_1\}$
DTLZ2	2	2	0.01	0.1	$Y = \{y \in \mathbb{R}^2 \mid y_1 \in [0,1], y_2 = \sqrt{1 - y_1^2}\}$
DTLZ2	3	2	0.01	0.1	
DTLZ3	2	2	0.1	0.1	$Y = \{y \in \mathbb{R}^2 \mid y_1 \in [0,1], y_2 = \sqrt{1 - y_1^2}\}$
DTLZ4	2	2	0.1	0.1	$Y = \{y \in \mathbb{R}^2 \mid y_1 \in [0,1], y_2 = \sqrt{1 - y_1^2}\}$
DTLZ5	2	2	0.01	0.1	$Y = \{y \in \mathbb{R}^2 \mid y_1 \in [0,1], y_2 = \sqrt{1 - y_1^2}\}$
DTLZ5	3	2	0.01	0.1	
DTLZ6	2	2	0.01	0.1	for $m = 2$: $Y = \{y \in \mathbb{R}^2 \mid y_1 \in [0,1], y_2 = \sqrt{1 - y_1^2}\}$ for $m = 3$: $Y = \left\{y \in \mathbb{R}^3 \mid y_1 = y_2 \in [0, \frac{1}{\sqrt{2}}], y_3 = \sqrt{1 - y_1^2 - y_2^2}\right\}$
DTLZ6	3	2	0.01	0.1	
DTLZ6	3	3	0.01	0.1	
DTLZ6	4	2	0.01	0.1	
DTLZ6	4	3	0.01	0.1	
DTLZ7	2	2	0.01	0.1	$Y = \left\{ y \in \mathbb{R}^2 \middle\| \begin{array}{l} y_1 \in [0, 0.251] \cup [0.633, 0.859], \\ y_2 = 4 - y_1(1 + \sin(3\pi y_1)) \end{array} \right\}$
DTLZ7	3	2	0.01	0.1	

Table 5.4. Settings for the instances with a known nondominated set, $m = 2$.

name	n	ε	δ	nondominated set $Y = f(X_E)$
ZDT1	2	0.01	0.1	$Y = \{y \in \mathbb{R}^2 \mid y_1 \in [0,1], y_2 = 1 - \sqrt{y_1}\}$
ZDT1	3	0.01	0.1	
ZDT1	4	0.01	0.1	
ZDT2	2	0.01	0.1	$Y = \{y \in \mathbb{R}^2 \mid y_1 \in [0,1], y_2 = 1 - y_1^2\}$
ZDT2	3	0.01	0.1	
ZDT4	2	0.01	0.1	$Y = \{y \in \mathbb{R}^2 \mid y_1 \in [0,1], y_2 = 1 - \sqrt{y_1}\}$
ZDT4	3	0.01	0.1	
FF	2	0.05	0.1	$Y = \left\{ y \in \mathbb{R}^2 \middle\| y = \begin{pmatrix} 1 - \exp(-4(t-1)^2) \\ 1 - \exp(-4t^2) \end{pmatrix}, t \in [0,1] \right\}$
FF	3	0.05	0.1	
FF	4	0.05	0.1	
Deb41	2	0.01	0.01	$Y = \left\{ y \in \mathbb{R}^2 \middle\| \begin{array}{l} y_1 \in [0.1, 1], \\ y_2 = f_2(x), x_1 \in [0.1, 1], x_2 \approx 0.2 \end{array} \right\}$

5.3.3 Results

In Tables 5.5 to 5.8 all results are stated. As already mentioned, MOPBB was performed first and the number of points in the stable set $\mathcal{F}$ and the number DT of calls of the discarding test are counted. Moreover, the overall computational time for all `while`-loops was recorded. Afterwards, the performance indicators hypervolume $HV(\mathcal{F})$, spacing metric $SP(\mathcal{F})$ and generational distance $GD(\mathcal{F}, \mathcal{P})$ are computed if possible.

Then the controlled version of NSGA-II was performed with the settings mentioned in Section 5.3.2. Thereby, the first row for each test instance states the results of `gamultiobj` with `TolFun`$= 10^{-4}$. If a second row is present with no entries for MOPBB, this row states the results with `TolFun`$= 10^{-5}$. $|PF|$ denotes the number of found solutions of the first front, i. e., the approximation of the nondominated set. Because of the setting for the population size, the number in the $|PF|$-column is nearly equal to $|\mathcal{F}|$. Since NSGA-II is performed for ten times, the next columns of the tables mention the average values of needed generations, needed time, hypervolume, spacing metric and generational distance.

Bold numbers indicate the best results for each performance indicator or the computational time. Recall that a higher hypervolume is preferable. For all other indicators, spacing, generational distance and time, a smaller value is desirable. The number of generations needed in average are italic if the maximal number of generations (bounded by the number of calls of discarding tests of MOPBB) was reached.

Consider the row of Table 5.5 for DTLZ2 with the settings $n = m = 2$ exemplary. The first line, column 4 to 9 shows the results of MOPBB, and column 10 to 15 the ones obtained by `gamultiobj`. In average, `gamultiobj` needed about 1.99 seconds and 108.6 generations to compute an approximation of the nondominated set consisting of 209 points. The average hypervolume was 0.7736 which is less than the one obtained by MOPBB (0.7750). For that reason, we decreased `TolFun` to 10^{-5} and executed ten additional runs of `gamultiobj`. The results for those ten runs can be found in the second line of the current row. With that lower function tolerance the average computational time increased to 27.86 seconds. Each run needed 1652 generations which is the maximal number of allowed generations because MOPBB executed 1652 discarding

Table 5.5. Results for the DTLZ-instances (1-5) which are scalable in n and m.

			MOPBB						gamultiobj					
name	n	m	$\lvert\mathcal{F}\rvert$	DT	$t\,[s]$	HV	SP	GD	$\lvert PF\rvert$	∅ Gen	$\varnothing t\,[s]$	$\varnothing HV$	$\varnothing SP$	$\varnothing GD$
DTLZ1	2	2	365	5762	708.29	8.8676	**0.0595**	**0.0132**	366	110.1	**3.77**	**8.8743**	0.1274	0.0495
									366	2831.9	93.80	**8.8744**	**0.0026**	**0.0018**
DTLZ2	2	2	209	1652	159.13	**0.7750**	0.0043	**5.8e-05**	209	108.6	**1.99**	0.7736	**0.0026**	1.9e-04
									209	*1652.0*	27.86	0.7737	**0.0023**	1.7e-04
DTLZ2	3	2	213	14282	1384.56	**1.4624**	0.0043	**5.6e-05**	214	105.7	**2.07**	1.4611	**0.0037**	8.0e-04
									214	12098.1	22.65	1.4613	**0.0024**	1.7e-04
DTLZ3	2	2	640	13744	1730.53	8.2052	**0.0404**	**0.0032**	641	114.4	**7.55**	**8.2131**	0.1624	0.0079
									641	1051.9	67.96	**8.2135**	**0.0008**	**0.0015**
DTLZ4	2	2	100	5580	1584.24	0.7696	0.0239	**0.0005**	101	111.2	**1.09**	**0.7701**	**0.0058**	0.0024
									101	*5580.0*	51.60	**0.7704**	**0.0053**	**0.0003**
DTLZ5	2	2	205	1650	373.95	**0.7749**	0.0044	**5.8e-05**	206	106.2	**3.06**	0.7735	**0.0028**	4.4e-04
									206	*1650.0*	46.47	0.7737	**0.0025**	1.7e-04
DTLZ5	3	2	213	14266	3358.42	**1.4624**	0.0043	**5.6e-05**	214	105.7	**3.53**	1.4611	**0.0037**	8.0e-04
									214	11276.4	371.04	1.4612	**0.0024**	1.7e-04

For DTLZ1 and DTLZ3 we used another reference point than $\sup(F(X))$. We used here $(3,3)^T$

Table 5.6. Results for the DTLZ-instances (6,7) which are scalable in n and m.

			MOPBB						gamultiobj					
name	n	m	$\lvert\mathcal{F}\rvert$	DT	$t\,[s]$	HV	SP	GD	$\lvert PF\rvert$	∅ Gen	$\varnothing t\,[s]$	$\varnothing HV$	$\varnothing SP$	$\varnothing GD$
DTLZ6	2	2	93	288	8.85	**3.2052**	0.0195	**6.1e-05**	94	115.7	**1.64**	2.8736	**0.0085**	0.0234
									94	*288.0*	4.09	3.0204	**0.0081**	0.0168
DTLZ6	3	2	45	1020	17.86	**8.1977**	0.0423	**0.0004**	46	125.6	**1.39**	7.1171	**0.0242**	0.0920
									46	*1020.0*	11.71	7.9259	**0.0165**	0.0280
DTLZ6	3	3	1799	5326	301.79	-	**0.0027**	**1.6e-05**	1799	110.2	**50.04**	-	0.0044	0.0048
									1799	391.8	156.08	-	0.0038	0.0039
DTLZ6	4	2	67	13680	279.25	**15.2026**	0.0379	**0.0001**	67	124.6	**2.24**	13.0292	**0.0222**	0.1266
									67	*13680.0*	235.92	14.8368	0.0105	0.0251
DTLZ6	4	3	281	15378	418.01	-	**0.0201**	**0.0001**	282	113.8	**9.93**	-	0.0250	0.0353
									282	1430.5	126.11	-	**0.0197**	0.0278
DTLZ7	2	2	235	502	39.27	**18.7261**	0.0046	**0.0003**	235	127.2	**2.73**	18.7194	**0.0043**	0.0005
									235	*502.0*	10.76	18.7255	0.0119	0.0090
DTLZ7	3	2	240	3078	209.30	**18.7261**	**0.0014**	**0.0004**	241	142.3	**3.29**	18.7011	0.0049	0.0023
									241	3068.6	67.61	18.7257	0.0028	**0.0003**

Table 5.7. Results for the instances with known nondominated set, $m = 2$.

		MOPBB						gamultiobj					
name	n	$\lvert\mathcal{F}\rvert$	DT	$t\ [s]$	HV	SP	GD	$\lvert PF\rvert$	∅ Gen	$\varnothing t\ [s]$	$\varnothing HV$	$\varnothing SP$	$\varnothing GD$
ZDT1	2	282	598	16.50	**9.6646**	**0.0011**	**0.0002**	283	107.8	**2.27**	9.6626	0.0062	0.0013
								283	*598.0*	12.40	9.6633	0.0016	**0.0001**
ZDT1	3	283	1996	34.25	**9.6646**	0.0055	**0.0001**	284	117.4	**2.82**	9.6610	**0.0047**	0.0014
								284	1592.8	35.58	9.6632	**0.0016**	**0.0001**
ZDT1	4	283	26828	518.52	**9.6646**	**0.00543**	**0.0001**	284	127.6	**3.16**	9.6565	0.00537	0.0026
								284	7608.1	172.40	9.6630	**0.0016**	0.0001
ZDT2	2	492	508	35.37	**9.3319**	**0.0013**	**9.8e-05**	493	111	**4.43**	9.3312	0.0174	9.4e-04
								493	462.0	17.52	9.3314	**0.0009**	**9.6e-05**
ZDT2	3	419	4810	276.99	**9.3316**	**0.0022**	**7.8e-05**	419	108.3	**3.92**	9.3264	0.0045	8.8e-04
								419	2666.8	91.05	9.3309	**0.0011**	**4.0e-05**
ZDT4	2	297	2018	46.44	**3.6648**	**0.0016**	**3.4e-05**	298	108.1	**2.45**	3.6633	0.0134	1.2e-03
								298	2009.2	44.56	3.6639	0.0017	9.3e-05
ZDT4	3	311	151436	7481.58	**3.6647**	**0.0041**	**3.6e-05**	312	110.5	**3.02**	3.6629	0.0119	1.4e-03
								312	7301.3	191.34	3.6640	**0.0015**	1.2e-04
FF	2	124	744	44.42	**0.3362**	0.0064	**5.4e-05**	124	105.6	**0.88**	0.3361	**0.0037**	1.4e-04
								124	*744.0*	6.10	0.3360	0.0038	1.5e-04
FF	3	233	8636	521.81	0.3376	0.0044	**4.7e-05**	234	103.5	**1.64**	**0.3387**	**0.0018**	8.4e-05
								234	5070.3	80.11	0.3386	0.0020	8.3e-05
FF	4	380	77626	5769.81	0.3386	0.0033	**3.6e-05**	381	105.7	**2.89**	**0.3399**	**0.0011**	5.0e-05
								381	2613.3	68.96	0.3399	0.0012	5.1e-05

For ZDT4 we used another reference point than $\sup(F(X))$. We used here $(2, 2)^T$

Table 5.8. Results for the instances which are not scalable and with analytically unknown nondominated set.

			MOPBB					gamultiobj				
name	ε	δ	$\lvert\mathcal{F}\rvert$	DT	$t\,[s]$	HV	SP	$\lvert PF\rvert$	∅ Gen	$\varnothing t\,[s]$	$\varnothing HV$	$\varnothing SP$
Deb41	0.01	0.01	657	4734	355.05	15.6098	0.2327	657	131	**6.55**	**15.6100**	**0.0656**
Deb41Cons	0.01	0.01	469	3067	23.95	14.1060	0.0199	469	110.0	**4.91**	**14.1117**	**0.0137**
Deb513	0.01	0.1	142	406	35.99	**21.6640**	**0.0068**	143	106.8	**1.30**	21.6604	0.0201
								143	*406*	4.85	**21.6643**	**0.0041**
Deb521a	0.1	0.1	8167	4068	844.02	1.0886	3.3e-04	8167	110.7	**286.73**	**1.1044**	**7.1e-05**
Deb521b	0.01	0.1	258	556	34.54	**1.3313**	**0.0013**	258	107.3	**1.94**	1.3293	0.0030
								258	*556*	10.01	1.3298	0.0030
Viennet	0.1	0.1	1587	2084	458.53	-	0.0079	1587	104.2	**18.47**	-	**0.0050**
ViennetCons	0.1	0.1	288	1205	246.09	-	0.0095	289	104.9	**2.76**	-	**0.0089**
SRN	1	0.5	2239	2610	331.10	3.43e+05	0.0861	2239	106.3	**39.52**	**3.44e+05**	**0.0475**
Pol	0.1	0.1	282	928	134.48	**535.88**	0.0358	283	124.0	**2.74**	535.74	**0.0227**
KW2	0.1	0.1	1216	2990	779.42	155.92	0.0098	1216	122.8	**13.69**	**155.97**	**0.0042**
Far1	0.1	0.1	600	1856	471.44	22.0258	0.0046	600	102.5	**4.23**	**22.0420**	**0.0022**

For Pol and KW2 we used another reference point than $\sup(F(X))$. We used here $(20, 30)$ and $(7, 7)^T$, respectively.

tests in the first and third `while-loop`. The performance indicators improved with this smaller function tolerance, but the hypervolume and the generational distance did not improve in comparison to the ones obtained by MOPBB.

For some instances, $\sup(F(X))$ was too large for a reference point to compute comparable values for the hypervolume. Another reference point was chosen in those cases. The chosen reference points are stated below the certain tables.

General Results Overall, it cannot be stated that one of the both algorithms outperforms the other one regarding the chosen performance indicators. This is expectable since different test instances with diverse difficulties were tested. NSGA-II is well developed and implemented, and widely used, besides it does not guarantee any accuracies. Nevertheless, there are some instances for which MOPBB obtained better results, for example, for the ZDT-instances. In many cases, NSGA-II was performed with a smaller tolerance additionally to see whether this smaller tolerance for `gamultiobj` leads to better results than those from MOPBB. Even with higher precision, NSGA-II could not improve some parameters in few cases (for example, FF), or the improvement were just small. The results for smaller tolerances also showed that the multiple runs of NSGA-II differ very much. This is also typical for evolutionary algorithms and a reason why one should run NSGA-II more than once. Even in dimension $n = 2$, MOPBB was sometimes as fast as NSGA-II. In higher dimensions ($n \geq 3$) this is not the case, because the computational time of MOPBB increases significantly in higher dimensions.

In most cases, `gamultiobj` delivers a smaller value for the spacing metric than MOPBB. The reason for that is the way how points for a new population are selected. This depends on the crowding distance which estimates the density of points surrounding another point. If a point is less crowded, it is a good choice for the new population. Therefore, NSGA-II tries to find a well distributed approximation of the nondominated set which leads to a smaller spacing metric. In contrast to this, MOPBB does not take the distribution of the points into account. However, it aims for finding points close to the efficient and nondominated set. Thus, the hypervolume and generational distance obtained by MOPBB are in most cases better than the ones obtained by `gamultiobj`.

The instances for which NSGA-II delivers better performance indicators than MOPBB are, in particular, "simple" instances (Viennet, Pol, Far1, KW2). This means that they are defined for low dimensions, the image points are relatively well distributed in the image space and there are no locally efficient points which are not globally efficient. Conversely, the DTLZ- and ZDT-instances are made for testing primarily evolutionary algorithms and have diverse properties which are difficult to handle by evolutionary algorithms.

In the next paragraphs, some individual cases and issues are considered, which are more noticeable than others.

DTLZ-instances The first five DTLZ-instances (DTLZ1-DTLZ5) require a lot of computational time for the performance of MOPBB, which is the reason why instances with larger values for n and m exceed the time limit. The remaining two DTLZ-instances were solved faster and could be scaled into higher dimensions, i. e., DTLZ6 could be scaled up to $n = 4$ and $m = 3$. Nevertheless, `gamultiobj` was always faster and would also run for higher dimensions. In every case, MOPBB gives better values for the generational distances than `gamultiobj`. Often, also the hypervolume and spacing metric are better, although MOPBB does not aim at a large hypervolume or a small spacing metric directly. Here, the smaller tolerances for NSGA-II could improve the indicators sometimes. However, the improvements of the hypervolume were not that significant.

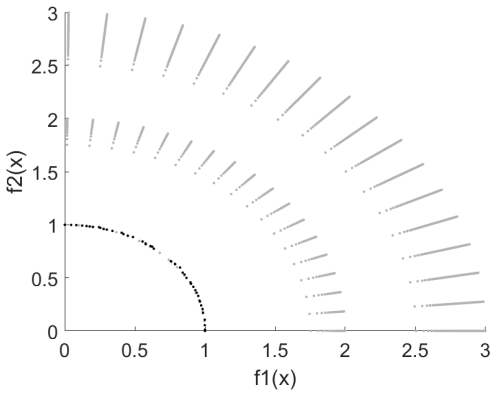
ZDT-instances and FF The ZDT-instances are a good example for which MOPBB works well. In low dimensions, i. e., for $n = 2$, the deterministic algorithm is very fast but not faster than the evolutionary one, and has better performance indicators in all cases. Actually, decreasing the tolerance for NSGA-II could not give better values for the performance indicators than MOPBB on some instances. Noteworthily, `gamultiobj` even delivered worse values for the FF instance when requiring a higher tolerance. This emphasizes how unpredictable evolutionary algorithms are. Even smaller tolerances do not lead to better results in every case.

Aiming for Globality To evaluate the ability of finding global solutions of a multiobjective optimization problem, we take a closer look on DTLZ6, ZDT4, Deb41 and Deb521a. Those instances have locally efficient and nondominated points which are not globally efficient and nondominated, respectively.

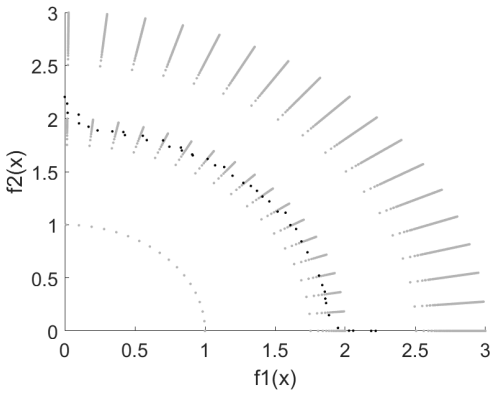
First, we consider the test instance DTLZ6. Since the globally nondominated set is known, the generational distance is a suitable indicator for how well the globally nondominated set is approximated. The high average values for the generational distance obtained by `gamultiobj` already indicate that the evolutionary algorithm cannot always find points close to the globally nondominated set. In fact, the individual generational distances for $n = 3,\ m = 2, \texttt{TolFun} = 10^{-4}$ differ from $8.6 \cdot 10^{-4}$ to $1.4 \cdot 10^{-1}$. Thus, there are runs of `gamultiobj` which found points close to the nondominated set, and there are other runs which could not reach it. Figure 5.8 shows the image space and the obtained approximations of the nondominated set of different runs of `gamultiobj`.

The upper pictures show the best ($HV = 8.1939,\ GD = 8.6 \cdot 10^{-4}$) and the worst ($HV = 6.0968,\ GD = 0.1414$) run for $n = 3,\ m = 2$. In all illustrations of this section, the gray points are some function values obtained by applying the objective function to a discretized feasible set. The black points are the points of PF or $\mathcal{F}$ obtained by `gamultiobj` or MOPBB, respectively. Obviously, the approximation in Figure 5.8b is very far away from the actual nondominated set: The image points are near to the circle line of the quarter circle with radius 2 instead of the quarter circle with radius 1, where the nondominated set is located. The lower two pictures Figures 5.8c and 5.8d are for the instance DTLZ6 with $n = m = 3$. On the left, we present the approximation obtained by MOPBB, and on the right, one approximation obtained by one run of `gamultiobj` is shown. MOPBB is able to approximate the nondominated set, which is a curve here, while in this run `gamultiobj` did not find any image point which is close to the nondominated set.

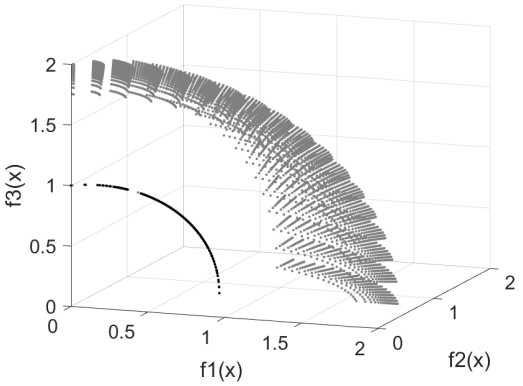
The instance ZDT4 has many locally nondominated points. Here, MOPBB delivers a value for the generational distance of $3.5 \cdot 10^{-5}$. The approximations by `gamultiobj` have an average generational distance of $1.3 \cdot 10^{-3}$ which is nearly 40 times more than the one by MOPBB. Figure 5.9 visualizes the obtained approximation by MOPBB and the worst (i. e., the one with the highest generational distance) approximation by



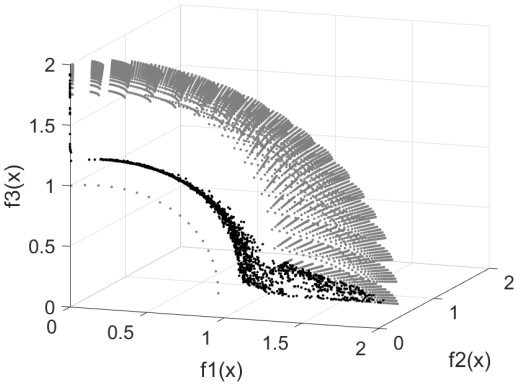
a: Approximation of the nondominated set of DTLZ6, $n = 3,\ m = 2$, obtained by `gamultiobj` — best run.



b: Approximation of the nondominated set of DTLZ6, $n = 3,\ m = 2$, obtained by `gamultiobj` — worst run.



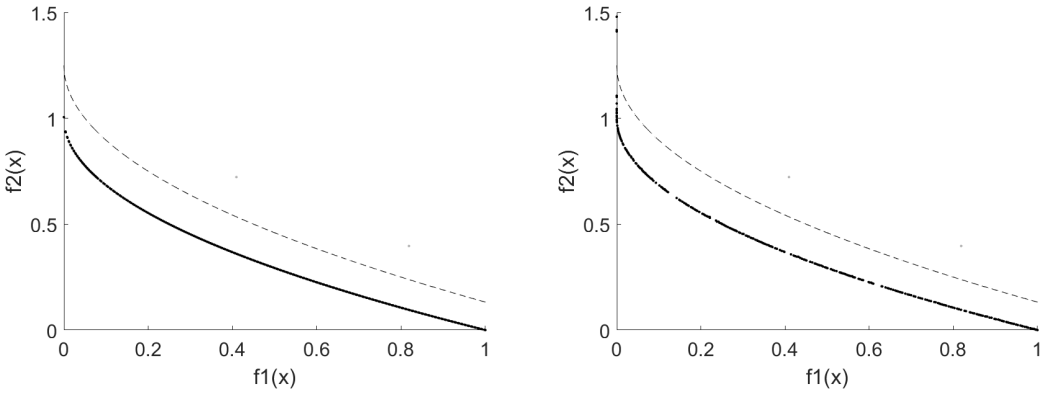
c: Approximation of the nondominated set of DTLZ6, $n = 3,\ m = 3$, obtained by MOPBB.



d: Approximation of the nondominated set of DTLZ6, $n = 3,\ m = 3$, obtained by `gamultiobj`.

Figure 5.8. Globally vs. locally nondominated sets of DTLZ6.

gamultiobj. The dashed line is the best locally nondominated set, which is not globally nondominated. Although the generational distances differ between MOPBB and gamultiobj, both algorithms could find a suitable approximation of the nondominated set.

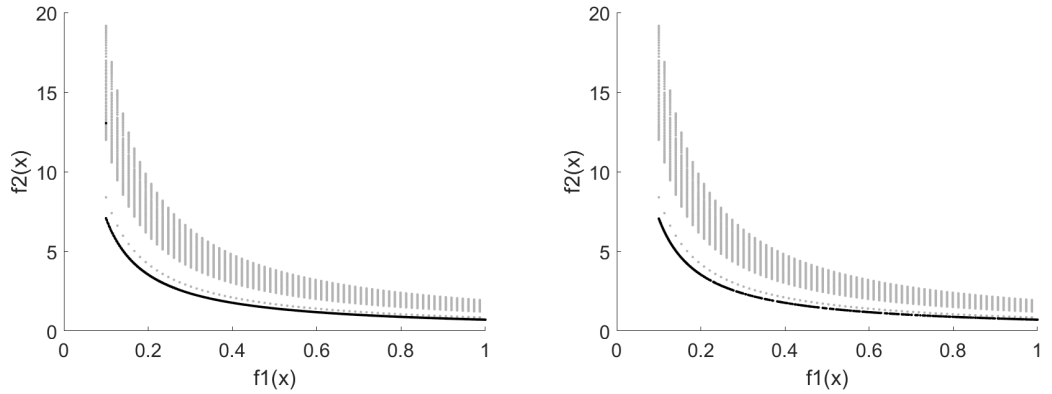


a: Approximation of the nondominated set of ZDT4 obtained by MOPBB.

b: Approximation of the nondominated set of ZDT4 obtained by gamultiobj.

Figure 5.9. Globally vs. locally nondominated sets of ZDT4 with $n = 3$.

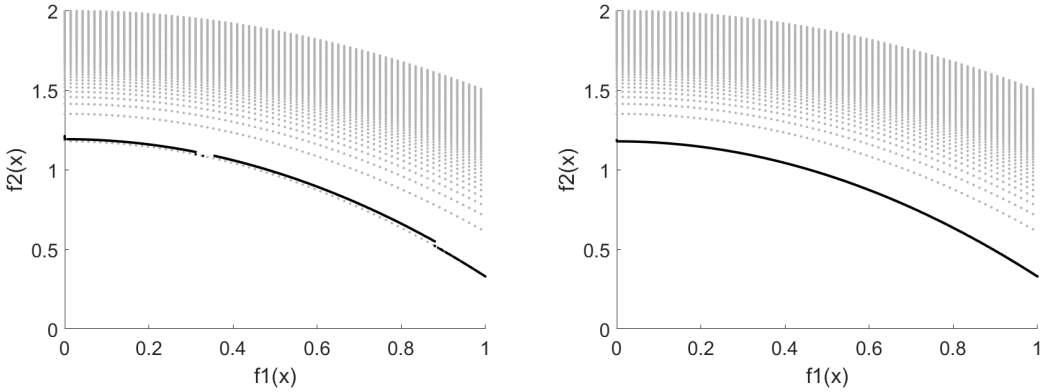
Furthermore, in Figures 5.10 and 5.11 it can be seen that both algorithms could reach the globally nondominated set of the test instances Deb41 and Deb521a.



a: Approximation of the nondominated set of Deb41 obtained by MOPBB.

b: Approximation of the nondominated set of Deb41 obtained by gamultiobj.

Figure 5.10. Globally vs. locally nondominated sets of Deb41.



a: Approximation of the nondominated set of Deb521a obtained by MOPBB.

b: Approximation of the nondominated set of Deb521a obtained by `gamultiobj`.

Figure 5.11. Globally vs. locally nondominated sets of Deb521a.

For test instance Deb521a, we have to remark that we slightly changed the feasible set in the test instance definition. The reason for that is the non-differentiability of the second objective function in $x_2 = 0$ although there are efficient points of Deb521a with $x_2 = 0$. We restricted the feasible set for x_2 to $[0.001, 1]$ for both algorithms, MOPBB and `gamultiobj`. This might be the cause why MOPBB was a little bit worse than `gamultiobj` in this case. However, it seems that `gamultiobj` had also some troubles to get appropriate solutions because in average it needed a higher amount of time (more than 4 minutes) than usual.

All those figures except Figure 5.8 suggest that both algorithms are able to approximate the globally nondominated set. Nevertheless, there are instances (like DTLZ6) designed to mislead evolutionary algorithms. For those the evolutionary algorithm resulted in points which are far away from the nondominated set.

6 Solving Multiobjective Mixed Integer Convex Optimization Problems

When a problem involves both continuous and integer variables we are in the context of multiobjective mixed integer programming. In this chapter, we focus on multiobjective mixed integer convex optimization problems (MOMICPs), namely problems of the following form:

$$\begin{aligned} \min \quad & f(x) = \begin{pmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{pmatrix} \\ \text{s.t.} \quad & g_k(x) \leq 0 \text{ for all } k = 1, \dots, p, \\ & x \in X := [l, u], \\ & x_i \in \mathbb{Z} \qquad \forall i \in I, \end{aligned} \tag{$MOMIC$}$$

where $f_j, g_k \colon X \to \mathbb{R},\ j = 1, \dots, m,\ k = 1, \dots, p$ are convex and continuously differentiable functions. The vectors $l, u \in \mathbb{R}^n$ are lower and upper bounds on the variables $x \in \mathbb{R}^n$ and define a box X. The index set $I \subseteq \{1, \dots, n\}$ specifies which variables have to take integer values. We assume without loss of generality that $l_i, u_i \in \mathbb{Z}$ holds for all $i \in I$.

Multiobjective mixed integer optimization problems arise in many application fields such as location or production planning, finance, manufacturing, and emergency management (see [Ehr+09; PY14; XMP10]). As an example, we can think of the uncapacitated facility location problem, studied in the single objective case in [GLW07]. The first objective hereby is to decide which facilities to build in order to minimize costs. As a second objective function one could consider the total negative impact on

the environment with the building plan for the facilities, e. g., the carbon emissions, [Roc18].

It is well-known that mixed integer nonlinear optimization is NP-hard and its solution typically requires dealing with enormous search trees, [Bel+13]. Handling more than one objective function adds an additional difficulty: assume there is only one binary variable, i. e., $I = \{1\}$ with $x_1 \in \{0, 1\}$, and we have just one objective function, i. e., $m = 1$. Then for solving $(MOMIC)$ only two convex optimization problems have to be addressed, one with x_1 fixed to 0 and one with x_1 fixed to 1. Clearly, the smallest minimal value is the optimal value of the original problem. In case of two or more objective functions already this simple setting is much more challenging. Solving the problems with fixed values for x_1 would mean to determine the whole efficient set of a multiobjective convex optimization problem, which is in general infinite. After computing two sets of nondominated points, one has to compare them and to determine the "smallest" values. See Figure 6.1 on page 107 for an illustration of this observation for four choices of the integer variables.

6.1 Definitions and Notations for MOMICPs

Given a vector $x \in \mathbb{R}^n$ and an index set $I \subseteq \{1, \dots, n\}$, we denote by x_I the subvector with components x_i, $i \in I$.

By X^g, $X^{\mathbb{Z}}$ and $X^{g,\mathbb{Z}}$ we denote the following sets related to the constraints of the optimization problem $(MOMIC)$:

$$\begin{aligned} X^g &:= \{x \in X \mid g(x) \leq 0\}, \\ X^{\mathbb{Z}} &:= \{x \in X \mid x_i \in \mathbb{Z} \text{ for all } i \in I\}, \\ X^{g,\mathbb{Z}} &:= X^g \cap X^{\mathbb{Z}}. \end{aligned} \tag{6.1}$$

Using these sets, we can write $(MOMIC)$ in short form as

$$\begin{aligned} \min \quad & f(x) \\ \text{s. t.} \quad & x \in X^{g,\mathbb{Z}}. \end{aligned}$$

As mentioned before, we are going to define a B&B method based on partitioning the feasible set of $(MOMIC)$. Our branching rule is based on bisections of the box X. Let $\tilde{X}$ be a subbox of X. By $\tilde{X}^g$, $\tilde{X}^{\mathbb{Z}}$ and $\tilde{X}^{g,\mathbb{Z}}$ we denote the sets defined according to (6.1), where the set X is replaced by $\tilde{X}$ (i. e., "$x \in \tilde{X}$" in all the set definitions).

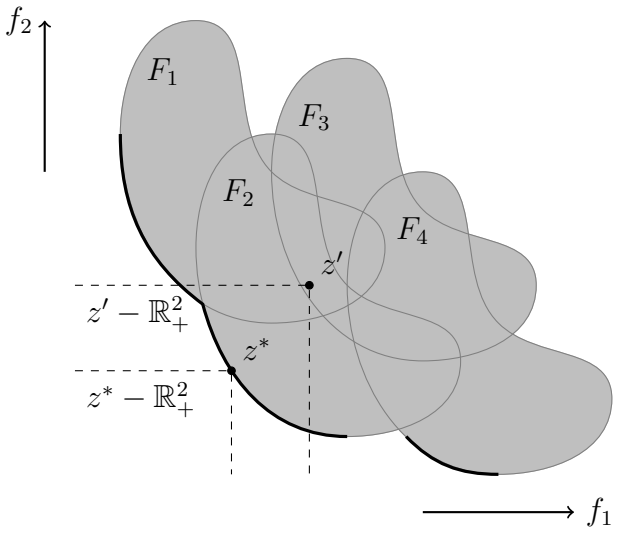


Figure 6.1. Image set of a biobjective instance of $(MOMIC)$.

In Figure 6.1, we plot the image set of a biobjective mixed integer convex optimization problem. Here, we assume that $\{x_I \mid x \in X^{g,\mathbb{Z}}\} =: \{y^1, y^2, y^3, y^4\}$ and we show the sets $F_j := \{f(x) \mid x \in X^{g,\mathbb{Z}},\ x_I = y^j\},\ j = 1, \dots, 4$. Then the union of all F_j describes the whole image set, i. e., $\bigcup_{j=1,\dots,4} F_j = \{f(x) \mid x \in X^{g,\mathbb{Z}}\}$. The point $z^* \in f(X^{g,\mathbb{Z}})$ is nondominated and the preimage of z^* is an efficient point. On the other hand, $z' \in f(X^{g,\mathbb{Z}})$ is dominated because of $z^* \leq z'$ and $z^* \neq z'$. In fact, all the points $z \in F_3$ are dominated, as points $w \in f(X^{g,\mathbb{Z}})$ exist such that $w < z$. The nondominated set of the problem is visualized as the thick boundary of the image set. The efficient set is made of all preimages of the nondominated set. Figure 6.1 shows that the nondominated set of a multiobjective mixed integer nonlinear programming problem is in general a disconnected set. From an algorithmic point of view, this makes the detection of the efficient set of $(MOMIC)$ an extremely challenging problem. Furthermore, there is the necessity of comparing sets of points: this is a crucial difference with respect to single objective mixed integer nonlinear optimization.

6.2 An Outer Approximation Based B&B Algorithm for MOMICPs

The algorithm we propose is a B&B method that seeks the efficient set of $(MOMIC)$ by partitioning the box X. At every node of the B&B tree, a subbox $\tilde{X} \subseteq X$ is selected and lower and upper bounds on the nondominated set of $(MOMIC)$ are derived. When considering the subbox $\tilde{X}$, a lower bound is any set $L_{\tilde{X}} \subseteq \mathbb{R}^m$ such that the set $L_{\tilde{X}} + \mathbb{R}^m_+$ contains the image of feasible points $\tilde{X}^{g,\mathbb{Z}}$ through f, namely $f(\tilde{X}^{g,\mathbb{Z}}) \subseteq L_{\tilde{X}} + \mathbb{R}^m_+$. In Figure 6.2, we illustrate the set $f(\tilde{X}^{g,\mathbb{Z}})$ and a lower bound $L_{\tilde{X}}$ for a biobjective purely integer programming problem: note that the image of feasible points in $\tilde{X}$ through f is a set of isolated points in $\mathbb{R}^m$.

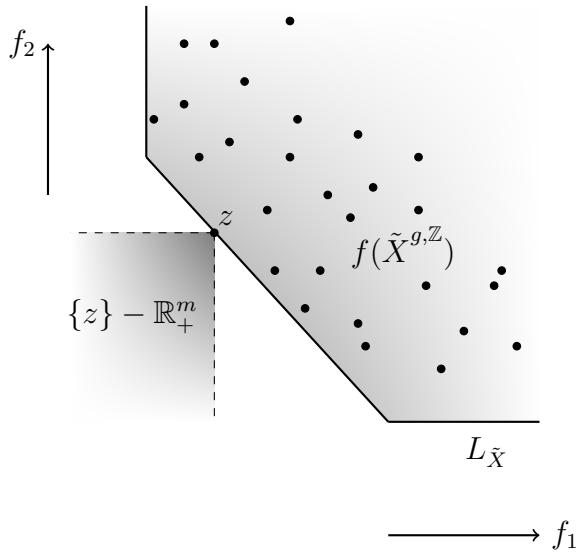


Figure 6.2. Image set of a biobjective purely integer instance of $(MOMIC)$.

In our algorithm we derive lower bounds by building linear outer approximations of $\text{conv}(f(\tilde{X}^{g,\mathbb{Z}}))$. As $f(\tilde{X}^{g,\mathbb{Z}}) \subseteq \text{conv}(f(\tilde{X}^{g,\mathbb{Z}}))$ holds, linear outer approximations of the convex hull of $f(\tilde{X}^{g,\mathbb{Z}})$ are valid lower bounds on $\tilde{X}$. Details on how we compute the hyperplanes to outer approximate $\text{conv}(f(\tilde{X}^{g,\mathbb{Z}}))$ will be given in Section 6.2.2.

Upper bounds are just computed by evaluating the objective functions at feasible points. As soon as an upper bound z exists such that $L_{\tilde{X}} + \mathbb{R}^m_+ \subseteq \{z\} + \mathbb{R}^m_+ \setminus \{0\}$ we can discard the subbox $\tilde{X}$. Or, in other words, we can avoid to go on partitioning $\tilde{X}$, as we have an evidence that it cannot contain any efficient point for $(MOMIC)$. Our discarding procedure is in fact using a list of upper bounds and it will be detailed in Section 6.2.1 and Section 6.2.2.

In Figure 6.2, the point $z \in f(\tilde{X}^{g,\mathbb{Z}})$ is an upper bound for the nondominated set of the problem, as it is the image of an feasible point. All the points that belong to $\mathbb{R}^m \setminus (\{z\} + \mathbb{R}^m_+ \setminus \{0\})$ are candidates for belonging to the nondominated set (note that it is not enough to consider $\{z\} - \mathbb{R}^m_+$).

Let $\delta > 0$ be a positive scalar, which is the input parameter of our B&B method. As the output of our algorithm, we will have a list of subboxes $\tilde{X}$ with $\omega(\tilde{X}) < \delta$, containing the set of efficient points, and a list of upper bounds approximating the nondominated set.

Algorithm 7 is a basic scheme of our B&B procedure: $\mathcal{L}_W$ denotes the working list and contains boxes that still have to be examined. $\mathcal{L}_S$ is the list of boxes that fulfill the termination criteria, i. e., those subboxes $\tilde{X}$ that were not discarded and satisfy $\omega(\tilde{X}) < \delta$. In Section 6.2.3 we will prove that $\mathcal{L}_S$ represents a cover of the efficient set X_E, namely $X_E \subseteq \bigcup_{\tilde{X} \in \mathcal{L}_S} \tilde{X}$. The list $\mathcal{L}_{PNS}$ denotes a set of upper bounds (as in MOPBB) and it will be defined in Section 6.2.1. Note that the flag $\mathcal{D}$ is used in order to decide if a box should be discarded, and it is an output of Algorithm 8. As a final step in Algorithm 7, we filter the list $\mathcal{L}_S$ by a postprocessing phase. Further details will be given in Section 6.2.2.

6.2.1 Computation of Upper Bounds

In order to compute upper bounds of the nondominated set of $(MOMIC)$, we evaluate the objective functions at feasible points $x \in X^{g,\mathbb{Z}}$. It is well-known in mixed integer optimization that determining feasible points of a mixed integer set is an NP-hard problem. In the literature, several heuristic methods have been proposed. The most popular one is the Feasibility Pump [FGL05] and some of its enhancements, among

Algorithm 7 MOMIX: A $(MOMIC)$ Solver

INPUT: $(MOMIC)$, $\delta > 0$

OUTPUT: $\mathcal{L}_S$, $\mathcal{L}_{PNS}$

1: $\mathcal{L}_S \leftarrow \emptyset$ $\mathcal{L}_W \leftarrow \{B\}$ $\mathcal{L}_{PNS} \leftarrow \emptyset$
2: **while** $\mathcal{L}_W \neq \emptyset$ **do**
3: Select a box $\tilde{X}$ of $\mathcal{L}_W$ and update $\mathcal{L}_W := \mathcal{L}_W \setminus \tilde{X}$
4: Bisect $\tilde{X}$ into subboxes $\tilde{X}^1$ and $\tilde{X}^2$
5: **for** $k = 1, 2$ **do**
6: Apply Algorithm 8 to $\tilde{X}^k$ and obtain $\mathcal{D}$ and an updated $\mathcal{L}_{PNS}$
7: **if** $\mathcal{D} =$ `true` **then** Discard $\tilde{X}^k$
8: **else**
9: **if** $\omega(\tilde{X}^k) < \delta$ **then** Add $\tilde{X}^k$ to $\mathcal{L}_S$
10: **else** Add $\tilde{X}^k$ to $\mathcal{L}_W$
11: Postprocessing($\mathcal{L}_S, \mathcal{L}_{PNS}$)

them [Bol+12; Bon+09; DLR13; Gei+17]. Within our algorithm, we either detect feasible points by addressing specific single objective mixed integer convex programming problems (see Section 6.2.2) or we try to build feasible points simply by rounding the integer components of points $x \in \tilde{X}$, which are generated in our discarding test. Let round(x) be the point defined by

$$(\text{round}(x))_i = \begin{cases} [x_i], & i \in I \\ x_i, & \text{otherwise,} \end{cases} \quad \text{for each } i \in \{1, \dots, n\}.$$

If round$(x) \in X^{g,\mathbb{Z}}$ holds, $f(\text{round}(x))$ is a valid upper bound.

Upper bounds are needed in order to discard boxes or, in other words, to prune nodes in the B&B tree. For that, we need to introduce two finite sets of points, namely the list of potentially nondominated solutions $\mathcal{L}_{PNS} \subseteq f(X^{g,\mathbb{Z}})$ and the list of local upper bounds $\mathcal{L}_{LUB} \subseteq \mathbb{R}^m$. For a repetition of local upper bounds, see Section 3.4.

In our algorithm the list of potentially nondominated solutions $\mathcal{L}_{PNS}$ is initialized as the empty set. Then every time an upper bound $z \in f(X^{g,\mathbb{Z}})$ is computed, we check whether it is dominated by any point in $\mathcal{L}_{PNS}$. If this is the case, z is not added to $\mathcal{L}_{PNS}$. Otherwise, we update the list by adding z to $\mathcal{L}_{PNS}$ and by removing all the

upper bounds dominated by z from $\mathcal{L}_{PNS}$. This is the same procedure as described in Algorithm 3. By doing this, we ensure that $\mathcal{L}_{PNS}$ is a stable set of points. Recall that a set $\mathcal{N} \subseteq \mathbb{R}^m$ is said to be stable if there are no $x, y \in \mathcal{N}$ with $x \leq y$ and $x \neq y$, see also the last point of Definition 2.6.

Similar to Lemma 4.2 from Section 4.1.2, we can prove our main result for the pruning of nodes:

Theorem 6.1 *Consider a subbox $\tilde{X} \subseteq X$. Let $\mathcal{L}_{PNS} \subseteq f(X^{g,\mathbb{Z}})$ be a finite and stable set. Let $\mathcal{L}_{LUB}$ be the local upper bound set w. r. t. $\mathcal{L}_{PNS}$. If*

$$p \notin f(\tilde{X}^{g,\mathbb{Z}}) + \mathbb{R}^m_+ \quad \textit{holds for all } p \in \mathcal{L}_{LUB}, \tag{6.2}$$

$\tilde{X}$ does not contain any efficient point for $(MOMIC)$.

Proof. Assume by contradiction that an efficient point $x^* \in \tilde{X}^{g,\mathbb{Z}}$ for $(MOMIC)$ exists. From (6.2), we have

$$f(x^*) \nleq p \text{ for all } p \in \mathcal{L}_{LUB}. \tag{6.3}$$

Since $\mathcal{L}_{LUB}$ is a local upper bound set w. r. t. $\mathcal{L}_{PNS}$, it follows from (6.3) and Proposition 3.22 (ii), (a) and (b), that $f(x^*)$ does not belong to the search region S. Hence, there exists a point $z \in \mathcal{L}_{PNS}$ with $z \leq f(x^*)$. As $z \in \mathcal{L}_{PNS}$, a point $x' \in X^{g,\mathbb{Z}}$ exists such that $z = f(x')$. Since x^* is efficient for $(MOMIC)$, it follows $z = f(x') = f(x^*)$. Lemma 3.24 implies that there is a point $p' \in \mathcal{L}_{LUB}$ with $f(x^*) \leq p'$, which is a contradiction to (6.3) and the theorem is proved. □

From Theorem 6.1, since $\tilde{X}^{g,\mathbb{Z}} \subseteq \tilde{X}^g$ and $f(\tilde{X}^{g,\mathbb{Z}}) \subseteq \text{conv}(f(\tilde{X}^{g,\mathbb{Z}}))$ hold, we obtain the following corollary.

Corollary 6.2 *Let $\tilde{X}$ be a subbox of X. Let $\mathcal{L}_{PNS} \subseteq f(X^{g,\mathbb{Z}})$ be a finite and stable set and let $\mathcal{L}_{LUB}$ be the local upper bound set w. r. t. $\mathcal{L}_{PNS}$.*

(i) If

$$p \notin f(\tilde{X}^g) + \mathbb{R}^m_+ \quad \textit{holds for all } p \in \mathcal{L}_{LUB},$$

$\tilde{X}$ does not contain any efficient point for $(MOMIC)$.

(ii) If

$$p \notin \operatorname{conv}(f(\tilde{X}^{g,\mathbb{Z}})) + \mathbb{R}^m_+ \quad \textit{holds for all } p \in \mathcal{L}_{LUB},$$

$\tilde{X}$ *does not contain any efficient point for* $(MOMIC)$.

The following remark clarifies how the assumptions of Corollary 6.2 are related. Furthermore, it gives the basis of the hierarchy of lower bounds in our bounding procedure.

Remark 6.3 Note that due to the convexity of the objective functions $f_j,\ j = 1, \ldots, m$ and of $\tilde{X}^g$ the following holds

$$\operatorname{conv}(f(\tilde{X}^{g,\mathbb{Z}})) + \mathbb{R}^m_+ \subseteq f(\tilde{X}^g) + \mathbb{R}^m_+.$$

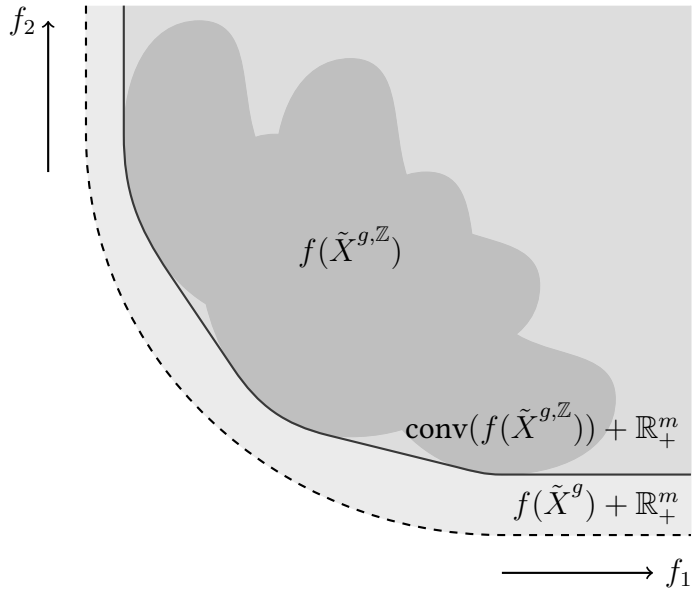


Figure 6.3. Discarding a box $\tilde{X}$ by Theorem 6.1 and Corollary 6.2.

Figure 6.3 illustrates Remark 6.3. The thick curve which touches some points of the boundary of $f(\tilde{X}^{g,\mathbb{Z}})$ is the boundary of $\operatorname{conv}(f(\tilde{X}^{g,\mathbb{Z}})) + \mathbb{R}^m_+$, while the dashed curve is the boundary of $f(\tilde{X}^g) + \mathbb{R}^m_+$.

6.2.2 Determining Lower Bounds and Pruning Nodes

The theoretical results introduced in the previous section, namely Corollary 6.2, give the basis of the discarding procedure in our B&B algorithm: For every subbox $\tilde{X}$ we check whether $p \notin L_{\tilde{X}} + \mathbb{R}^m_+$ holds for all $p \in \mathcal{L}_{LUB}$, being $L_{\tilde{X}}$ a valid lower bound for $\tilde{X}^{g,\mathbb{Z}}$.

As $f(\tilde{X}^g)$ is a valid lower bound, a straightforward way to verify if a box should be discarded would be to check whether a local upper bound $p \in \mathcal{L}_{LUB}$ belongs to this lower bound given by the convex relaxation. This would mean to check whether $p \in f(\tilde{X}^g) + \mathbb{R}^m_+$ holds. This can be done by addressing a simple single objective continuous convex problem.

From a computational point of view, this means that we would need to solve $|\mathcal{L}_{LUB}|$ single objective continuous convex problems, at every node of the branching tree.

In our algorithm, in order to reduce this numerical effort, we check instead whether a local upper bound belongs to a linear outer approximation of $f(\tilde{X}^g) + \mathbb{R}^m_+$, i. e., we only need to check whether a local upper bound satisfies linear inequalities. Furthermore, our linear outer approximations are built in a smart way: the supporting hyperplanes computation is adaptively driven by some "meaningful" local upper bounds $p \in \mathcal{L}_{LUB}$.

Additionally, if we want to improve our lower bound, we compute further hyperplanes to outer approximate $\text{conv}(f(\tilde{X}^{g,\mathbb{Z}})) + \mathbb{R}^m_+$. Again this computation is done in an adaptive way, and the supporting hyperplanes computation is steered by some specific local upper bounds $p \in \mathcal{L}_{LUB}$.

In what follows, we give details on how the supporting hyperplanes are computed and how the discarding procedure works.

At an arbitrary node of our branching tree we select a subbox $\tilde{X} \subseteq X$. In order to compute valid lower bounds on $\tilde{X}$, we build linear outer approximations $L_{\tilde{X}}$ of $\text{conv}(f(\tilde{X}^{g,\mathbb{Z}}))$, such that

$$f(\tilde{X}^{g,\mathbb{Z}}) \subseteq \text{conv}(f(\tilde{X}^{g,\mathbb{Z}})) \subseteq L_{\tilde{X}} + \mathbb{R}^m_+.$$

In order to discard the subbox $\tilde{X}$ and prune the current node, we check whether

$$p \notin L_{\tilde{X}} + \mathbb{R}^m \quad \text{holds for all } p \in \mathcal{L}_{LUB}.$$

Then from Corollary 6.2, $\tilde{X}$ does not contain any efficient point for ($MOMIC$) and the current node can be pruned. As we will deal with linear outer approximations of sets, recall the definition of supporting hyperplane

$$H^{\lambda,z} := \{y \in \mathbb{R}^m \mid \lambda^T y = \lambda^T z\}$$

of a set by Definition 3.15.

We propose two versions of the new B&B algorithm. The difference lies in the lower bounds computation. The first version of our algorithm, named MOMIX$_{light}$, computes valid lower bounds by addressing only single objective continuous convex optimization problems. The second version, named MOMIX, tries to define stronger lower bounds by dealing also with single objective mixed integer convex programming problems. In our algorithm we use a flag $light$ to distinguish between the two versions of the method.

Both, MOMIX$_{light}$ and MOMIX, start by computing linear outer approximations of the convex set $f(\tilde{X}^g) + \mathbb{R}^m_+$ by solving a family of single objective continuous convex optimization problems. As $\text{conv}(f(\tilde{X}^{g,\mathbb{Z}})) + \mathbb{R}^m_+ \subseteq f(\tilde{X}^g) + \mathbb{R}^m_+$ holds by Remark 6.3, we have that linear outer approximations of $f(\tilde{X}^g) + \mathbb{R}^m_+$ are valid lower bounds for $\text{conv}(f(\tilde{X}^{g,\mathbb{Z}}))$ as well (see Figure 6.4 on page 117). If the linear outer approximation of $f(\tilde{X}^g) + \mathbb{R}^m_+$ does not allow to discard the box $\tilde{X}$, MOMIX tries to improve it by addressing properly defined single objective mixed integer convex programming problems.

As a first step for the outer approximation, we compute the ideal point $a \in \mathbb{R}^m$ of $f(\tilde{X}^g)$, see also Definition 3.16:

$$a_j := \min_{x \in \tilde{X}^g} f_j(x) \quad j = 1, \ldots, m. \tag{6.4}$$

We denote by $\tilde{x}^j \in \tilde{X}^g$ a minimal solution of (6.4). Let u^j be the j-th unit vector, then $H^{u^j,a}$ is a supporting hyperplane of $f(\tilde{X}^g)$. As a first linear outer approximation of $f(\tilde{X}^g)$ or, in other words, as a first lower bound for $f(\tilde{X}^{g,\mathbb{Z}})$ we consider

$$L_{\tilde{X}} := \text{bd}\left(\bigcap_{j\in\{1,\dots,m\}} (H^{u^j,a} + \mathbb{R}^m_+)\right) = \{a\} + \text{bd}(\mathbb{R}^m_+). \tag{6.5}$$

Note that building $L_{\tilde{X}}$ requires to solve m single objective continuous convex optimization problems for the computation of a.

Once $L_{\tilde{X}}$ is computed, we enter in a loop. For every $p \in \mathcal{L}_{LUB}$ we check whether $p \in L_{\tilde{X}} + \mathbb{R}^m_+$ holds. If this is the case, we try to improve the current linear outer approximation $L_{\tilde{X}}$ by computing a further hyperplane based on $p \in \mathcal{L}_{LUB}$. This is done by addressing the following single objective continuous convex programming problem (see also [ESS11; LRU14] or Section 3.3)

$$\begin{aligned} \min \quad & t \\ \text{s.t.} \quad & f(x) \leq p + te \\ & x \in \tilde{X}^g \\ & t \in \mathbb{R}, \end{aligned} \tag{$P_p(\tilde{X}^g)$}$$

where $e = (1,\dots,1)^T \in \mathbb{R}^m$ is the m-dimensional all-ones vector.

Note that $(P_p(\tilde{X}^g))$ needs to be addressed *only* in case of $p \in L_{\tilde{X}} + \mathbb{R}^m_+$. In other words, in our lower bound computation, we do not necessarily address $(P_p(\tilde{X}^g))$ for all $p \in \mathcal{L}_{LUB}$, as it would be the case if we would only rely on the convex relaxation $f(\tilde{X}^g)$.

Under regularity assumptions, we have that any minimal solution $(\hat{x},\hat{t}) \in \tilde{X}^g \times \mathbb{R}$ of Problem $(P_p(\tilde{X}^g))$ admits Lagrange multipliers. We refer to [ESS11] in case of non existing Lagrange multipliers. Let $(\hat{x},\hat{t}) \in \tilde{X}^g \times \mathbb{R}$ be a minimal solution of $(P_p(\tilde{X}^g))$ and let $\hat{\lambda} \in \mathbb{R}^m_+$ be a Lagrange multiplier for the constraint $f(x) \leq p+te$. Then, the hyperplane $H^{\hat{\lambda},\hat{y}(p)}$ with $\hat{y}(p) := p + \hat{t}e$ is a supporting hyperplane of $f(\tilde{X}^g)$, cf. [LRU14] or Theorem 3.18.

There are two possibilities:

(i) If $\hat{t} > 0$ holds, then $p \notin f(\tilde{X}^g) + \mathbb{R}^m_+$, we improve the outer approximation by $H^{\hat{\lambda},\hat{y}(p)}$, and consider the next local upper bound.

(ii) If $\hat{t} \leq 0$ holds, then $p \in f(\tilde{X}^g) + \mathbb{R}^m_+$ and the assumption of Corollary 6.2 (i) is not satisfied.

If case (ii) occurs, so far we cannot discard $\tilde{X}$ based on Corollary 6.2 (i) as it may contain efficient points for ($MOMIC$). Then, in case we apply `MOMIX`, i. e., $light = 0$, we try to apply Corollary 6.2 (ii) and thus we try to improve our linear outer approximation by addressing a single objective mixed integer convex programming problem. Let $\hat{\lambda} \in \mathbb{R}^m$ be a Lagrange multiplier for the constraint $f(x) \leq p + te$ for the solution of ($P_p(\tilde{X}^g)$). We define the following problem

$$\begin{aligned} \min \quad & \hat{\lambda}^T f(x) \\ \text{s. t.} \quad & x \in \tilde{X}^{g,\mathbb{Z}}. \end{aligned} \tag{$MICP_p(\hat{\lambda}, \tilde{X})$}$$

Let $\hat{x} \in \tilde{X}^{g,\mathbb{Z}}$ be a minimal solution of ($MICP_p(\hat{\lambda}, \tilde{X})$). Then the hyperplane $H^{\hat{\lambda},f(\hat{x})}$ is a supporting hyperplane of $\text{conv}(f(\tilde{X}^{g,\mathbb{Z}}))$ and $\text{conv}(f(\tilde{X}^{g,\mathbb{Z}})) + \mathbb{R}^m_+ \subseteq H^{\hat{\lambda},f(\hat{x})} + \mathbb{R}^m_+$ holds. Furthermore, $f(\hat{x})$ is a valid upper bound for ($MOMIC$). Note that in case we are at a node where all integer variables are fixed, we do not need to perform this step, because $\tilde{X}^{g,\mathbb{Z}} = \tilde{X}^g$ holds.

Again two situations occur:

(i) If $\hat{\lambda}^T p < \hat{\lambda}^T f(\hat{x})$ holds, we improve the outer approximation by $H^{\hat{\lambda},f(\hat{x})}$ and consider the next local upper bound.

(ii) If $\hat{\lambda}^T p \geq \hat{\lambda}^T f(\hat{x})$ holds, the local upper bound p lies above the hyperplane $H^{\hat{\lambda},f(\hat{x})}$.

If we are in case (ii), we do not continue with improving the linear outer approximation and we branch the current node by bisecting $\tilde{X}$ in a later iteration.

Algorithm 8 is reporting our lower bound computation in detail.

As soon as feasible points of $\tilde{X}^{g,\mathbb{Z}}$ are found, both $\mathcal{L}_{PNS}$ and $\mathcal{L}_{LUB}$ are updated. This is the reason why in Algorithm 8 we make use of the list $\mathcal{L}^*_{LUB}$ which does not change

along the discarding test: We need a fixed set of local upper bounds in order to ensure the termination of the loop starting in line 8.

Note that in line 14 we update the linear outer approximation even if the subbox $\tilde{X}$ is further kept either in the working list $\mathcal{L}_W$ or in the solution list $\mathcal{L}_S$. This is done in order to perform the postprocessing phase in Algorithm 7: Let $\tilde{X} \in \mathcal{L}_S$ and let $\mathcal{H}$ be the linear outer approximation of $f(\tilde{X}^{g,\mathbb{Z}})$ built by Algorithm 8. This subbox $\tilde{X}$ is removed from $\mathcal{L}_S$ if for all local upper bounds p belonging to the final list $\mathcal{L}_{LUB}$ we have that a hyperplane $H^{\lambda,z'} \in \mathcal{H}$ exists such that $\lambda^T p \geq \lambda^T z'$ holds.

Example 6.4 In Figure 6.4 we illustrate our lower bounding procedure on a biobjective purely integer convex programming instance. Note that in this case the image of feasible points is a set of isolated points in $\mathbb{R}^2$. The first outer approximation considered is based on the ideal point a. Then, considering the local upper bound $p \in \mathcal{L}_{LUB}$, the supporting hyperplane $H^{\hat{\lambda},\hat{y}(p)}$ for $f(\tilde{X}^g)$ is built by solving Problem $(P_p(\tilde{X}^g))$ and added to the linear outer approximation. Finally, in case MOMIX (and not MOMIX_{light}) is applied, the linear outer approximation is further refined by considering $H^{\hat{\lambda},f(\hat{x})}$, being $\hat{x}$ a solution of $(MICP_p(\hat{\lambda}, \tilde{X}))$.

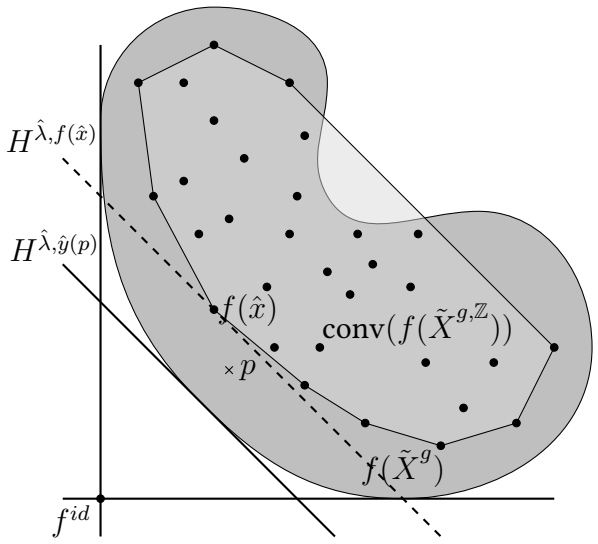


Figure 6.4. Illustration of our lower bounding procedure on a biobjective purely integer convex programming instance.

Algorithm 8 Lower bounding procedure

INPUT: ($MOMIC$), a subbox $\tilde{X} \subseteq B$, $\mathcal{L}_{PNS}$, $\mathcal{L}_{LUB}$, $light \in \{0, 1\}$

OUTPUT: $\mathcal{L}_{PNS}$, $\mathcal{L}_{LUB}$, $\mathcal{D}$, where $\mathcal{D} = \texttt{true}$ means "Discard $\tilde{X}$"

Set $\mathcal{D} \leftarrow \texttt{true}$
for $j \in \{1, \ldots, m\}$ **do**
    Compute f_j^{id} and obtain $\tilde{x}^j \in \tilde{X}^g$
    if round$(\tilde{x}^j) \in B^{g,\mathbb{Z}}$ **then**
        Update $\mathcal{L}_{PNS}$ by $f(\text{round}(\tilde{x}^j))$ and update $\mathcal{L}_{LUB}$ by Algorithms 1 and 3
Set $\mathcal{L}^*_{LUB} \leftarrow \mathcal{L}_{LUB}$
Set $\mathcal{H} \leftarrow \{H^{u^j,a} \mid j \in \{1, ..., m\}\}$
for $p \in \mathcal{L}^*_{LUB}$ **do**
    if $\lambda^T p \geq \lambda^T z'$ for all $H^{\lambda,z'} \in \mathcal{H}$ **then**
        Solve ($P_p(\tilde{X}^g)$) and get $(x^*, t^*) \in \tilde{X}^g \times \mathbb{R}$, $\hat{\lambda} \in \mathbb{R}^m$ Lagrange multiplier for the constraint $f(x) \leq p + te$
        if round$(x^*) \in B^{g,\mathbb{Z}}$ **then**
            Update $\mathcal{L}_{PNS}$ by $f(\text{round}(x^*))$ and update $\mathcal{L}_{LUB}$ by Algorithms 1 and 3
        if $t^* \leq 0$ and $light = 1$ **then**
            Set $\mathcal{H} \leftarrow \mathcal{H} \cup \{H^{\hat{\lambda},p+t^*e}\}$
            Set $\mathcal{D} \leftarrow \texttt{false}$ and **break for-loop**
        else if $t^* \leq 0$ and $light = 0$ **then**
            Solve Problem ($MICP_p(\hat{\lambda}, \tilde{X})$)
            if ($MICP_p(\hat{\lambda}, \tilde{X})$) is infeasible **then**
                Set $\mathcal{D} \leftarrow \texttt{true}$ and **break for-loop**
            else
                Let $\hat{x} \in \tilde{X}^{g,\mathbb{Z}}$ be a solution of Problem ($MICP_p(\hat{\lambda}, \tilde{X})$)
                Update $\mathcal{L}_{PNS}$ by $f(\hat{x})$ and update $\mathcal{L}_{LUB}$ by Algorithms 1 and 3
                Set $\mathcal{H} \leftarrow \mathcal{H} \cup \{H^{\hat{\lambda},f(\hat{x})}\}$
            if $\hat{\lambda}^T p \geq \hat{\lambda}^T f(\hat{x})$ **then**
                Set $\mathcal{D} \leftarrow \texttt{false}$ and **break for-loop**
        else
            Set $\mathcal{H} \leftarrow \mathcal{H} \cup \{H^{\hat{\lambda},p+t^*e}\}$.

In the following lemma, we prove the exactness of our lower bounding procedure: we show that Algorithm 8 returns $\mathcal{D} = \texttt{false}$ if $\tilde{X}$ contains an efficient point of $(MOMIC)$. Thus, it will be further partitioned.

Lemma 6.5 *Let $\tilde{X}$ be a subbox of X that contains an efficient point $x^* \in \tilde{X}^{g,\mathbb{Z}}$ of $(MOMIC)$. Then Algorithm 8 returns $\mathcal{D} = \texttt{false}$.*

Proof. We distinguish two cases for which Algorithm 8 returns $\mathcal{D} = \texttt{true}$: either $\mathcal{D} = \texttt{true}$ because $(MICP_p(\hat{\lambda}, \tilde{X}))$ is infeasible for any $p \in \mathcal{L}^*_{LUB}$ or because lines 15 or 25 are never reached for any $p \in \mathcal{L}^*_{LUB}$. The first case cannot occur as $x^* \in \tilde{X}^{g,\mathbb{Z}}$. The second case may occur if for all $p \in \mathcal{L}^*_{LUB}$ one of the conditions in line 9, 13 or 24 is not satisfied. In all three cases we get that $p \notin f(\tilde{X}^g) + \mathbb{R}^m_+$ holds for all $p \in \mathcal{L}^*_{LUB}$. Corollary 6.2 then implies that $\tilde{X}$ does not contain any efficient point for $(MOMIC)$. □

6.2.3 Correctness of MOMIX

We already mentioned that our algorithm stops as soon as the working list $\mathcal{L}_W$ is empty, and we get a list of subboxes $\tilde{X}$ of width less than a prescribed value $\delta > 0$, i. e., $\omega(\tilde{X}) < \delta$. In this section, we first prove the exactness of Algorithm 7, namely we prove that it returns the set $\mathcal{L}_S$ which is a cover of the efficient set X_E of $(MOMIC)$. In order to do that, we need to make the following assumption related to the branching rule adopted.

Assumption A.1 Let $\tilde{X} \subseteq X$. Let the branching rule in Algorithm 7 be in such a way that

$$\tilde{X}^{g,\mathbb{Z}} \subseteq \tilde{X}^1 \cup \tilde{X}^2$$

holds for the subboxes $\tilde{X}^1$ and $\tilde{X}^2$ derived from $\tilde{X}$, and that the algorithm performs a finite number of branching steps before stopping.

Note that Assumption A.1 implies that the set of efficient points for $(MOMIC)$ belonging to $\tilde{X}$ is a subset of $\tilde{X}^1 \cup \tilde{X}^2$. In Section 6.3 we propose and compare two branching rules which both satisfy Assumption A.1.

From Assumption A.1 and Lemma 6.5 we directly get the following:

Theorem 6.6 *Let X_E be the efficient set of $(MOMIC)$. Let $\mathcal{L}_S$ be the output of Algorithm 7. Then $\mathcal{L}_S$ is a cover of X_E, namely $X_E \subseteq \bigcup_{\tilde{X} \in \mathcal{L}_S} \tilde{X}$.*

We underline that when MOMIX (and not MOMIX_{light}) is applied, the list $\mathcal{L}_S$ is built in a way such that every subbox $\tilde{X}$ belonging to $\mathcal{L}_S$ admits at least one feasible point, i. e., $\tilde{X}^{g,\mathbb{Z}} \neq \emptyset$. Note that a feasible point $\hat{x} \in \tilde{X}^{g,\mathbb{Z}}$ is computed in line 21 in Algorithm 8.

We further prove that the points in the final list $\mathcal{L}_{PNS}$ are images of some points in the cover of the efficient set of $(MOMIC)$.

Proposition 6.7 *Let $\mathcal{L}_{PNS}$ and $\mathcal{L}_S$ be the output of Algorithm 7. Then, for every $z \in \mathcal{L}_{PNS}$ there exists a subbox $\tilde{X} \in \mathcal{L}_S$ such that $z \in f(\tilde{X}^{g,\mathbb{Z}})$.*

Proof. Assume by contradiction that the preimage x of $z \in \mathcal{L}_{PNS}$ belongs to a discarded subbox $\tilde{X}^{g,\mathbb{Z}}$. Then, at a certain node of our branching tree, a lower bound $L_{\tilde{X}}$ was computed such that for all $p \in \mathcal{L}'_{LUB}$

$$p \notin L_{\tilde{X}} + \mathbb{R}^m_+$$

holds, where $\mathcal{L}'_{LUB}$ is the list of local upper bounds at that node. Hence,

$$p \notin f(\tilde{X}^{g,\mathbb{Z}}) + \mathbb{R}^m_+ \text{ holds for all } p \in \mathcal{L}'_{LUB}. \tag{6.6}$$

Let $\mathcal{L}_{LUB}$ be the final list of local upper bounds related to $\mathcal{L}_{PNS}$. By Lemma 3.24, a local upper bound $\hat{p} \in \mathcal{L}_{LUB}$ exists such that $z \leq \hat{p}$ holds, i. e., $\hat{p} \in f(\tilde{X}^{g,\mathbb{Z}}) + \mathbb{R}^m_+$. If $\hat{p} \in \mathcal{L}'_{LUB}$ holds, we directly get a contradiction to (6.6). Otherwise, from Remark 3.26 it follows that a local upper bound $\bar{p} \in \mathcal{L}'_{LUB}$ exists such that $\hat{p} \leq \bar{p}$ holds. Hence, $\bar{p} \in f(\tilde{X}^{g,\mathbb{Z}}) + \mathbb{R}^m_+$ holds, which contradicts (6.6). □

We now show that, in case MOMIX is applied, $\mathcal{L}_{PNS}$ is a "good" approximation of the nondominated set. This means that the distance of the image of efficient points from $\mathcal{L}_{PNS}$ is bounded by a quantity that depends on $\delta > 0$, which is the input parameter of Algorithm 7. For this, we exploit the Lipschitz continuity of the objective functions

f_j, $j = 1, \ldots, m$, which holds as the functions are continuously differentiable and the feasible sets are compact. Let $L_j \geq 0$ be the Lipschitz constant for function f_j, $j = 1, \ldots, m$.

Theorem 6.8 *Let $\delta > 0$ be the input parameter and $\mathcal{L}_{PNS}$, $\mathcal{L}_S$ be the output of Algorithm 7 where* MOMIX *is applied, i. e., $light = 0$. Let $\mathcal{L}_{LUB}$ be the local upper bound set w. r. t. $\mathcal{L}_{PNS}$ and $X_E \subseteq X^{g,\mathbb{Z}}$ be the efficient set of $(MOMIC)$. Set $L := \max_{j=1,\ldots,m} L_j$. Then*

$$f(X_E) \subseteq \left(\bigcup_{p \in \mathcal{L}_{LUB}} (\{p\} - \mathbb{R}^m_+) \right) \bigcap \left(\bigcup_{z \in \mathcal{L}_{PNS}} (\{z - L\delta e\} + \mathbb{R}^m_+) \right)$$

holds, where $e = (1, \ldots, 1)^T \in \mathbb{R}^m$.

Proof. Let $x \in X_E$. In order to prove $f(x) \in \bigcup_{p \in \mathcal{L}_{LUB}} (\{p\} - \mathbb{R}^m_+)$, we distinguish two cases. Assume first that $f(x)$ belongs to the search region S related to $\mathcal{L}_{PNS}$ (see (3.18) and Proposition 3.22). Then, a local upper bound $p \in \mathcal{L}_{LUB}$ exists such that $f(x)$ belongs to the search zone $C(p)$ related to p. It follows that $f(x) < p$.

On the other hand, if $f(x) \notin S$, by (3.18) a point $z \in \mathcal{L}_{PNS}$ exists such that $z \leq f(x)$. Since $f(x)$ is nondominated and $z \in \mathcal{L}_{PNS}$ is the image of a feasible point, we necessarily have $f(x) = z$. From Lemma 3.24 we obtain that a $p \in \mathcal{L}_{LUB}$ exists such that $f(x) = z \leq p$ holds.

Next, we prove $f(x) \in \bigcup_{z \in \mathcal{L}_{PNS}} (\{z - L\delta e\} + \mathbb{R}^m_+)$: For this, we choose a box $\tilde{X} \in \mathcal{L}_S$ with $x \in \tilde{X}^{g,\mathbb{Z}}$ which exists by Theorem 6.6. From Algorithm 8, if $light = 0$, a feasible point $\hat{x} \in \tilde{X}^{g,\mathbb{Z}}$ is computed for $\tilde{X}$ (see line 21). The point $f(\hat{x})$ is an upper bound for $(MOMIC)$ and then a candidate to belong to $\mathcal{L}_{PNS}$. Then, either $f(\hat{x})$ is an element of $\mathcal{L}_{PNS}$ or $z \in \mathcal{L}_{PNS}$ exists such that $z \leq f(\hat{x})$. Since $\omega(\tilde{X}) < \delta$ holds, we have $\|x - \hat{x}\| < \delta$ and, by Lipschitz continuity of f_j we obtain for all $j = 1, \ldots, m$ $|f_j(x) - f_j(\hat{x})| \leq L_j\delta \leq L\delta$. Therefore, since $L\delta \geq 0$ holds, it follows $f_j(x) \geq f_j(\hat{x}) - L\delta \geq z_j - L\delta$ for all $j = 1, \ldots, m$, and the theorem is proved. □

An illustration of Theorem 6.8 on a test instance solved by $(MOMIC)$ is given in Figure 6.7 in Section 6.3.

6.3 Numerical Results

In this section, we present our numerical experience on different test instances of ($MOMIC$). Next to some results for biobjective quadratic instances, we show results for an instance with $m = 3$ objective functions and results for a mixed integer convex non-quadratic instance.

In our implementation of Algorithm 7, at line 3, in order to select a subbox $\tilde{X} \in \mathcal{L}_W$, we consider the ideal point a computed according to (6.4). We pick at first those subboxes with the lexicographic smallest ideal point a, with the idea that boxes with small a may lead to good upper bounds. Concerning the branching rule, we adopted two different strategies detailed in Section 6.3.1. For all runs we set $\delta = 0.1$ if it is not stated otherwise.

For the solution of the mixed integer convex programming problem used to define the hyperplane $H^{\hat{\lambda},f(\hat{x})}$ that enriches the linear outer approximation of $f(\tilde{X}^{g,\mathbb{Z}})$ (line 17 of Algorithm 8), we can adopt any solver which is able to deal with convex mixed integer linear problems as e. g. SCIP [Gle+18]. In our numerical experiments, we mainly used quadratic instances and within our implementation of MOMIX we adopted the mixed integer quadratic solver of GUROBI, [Gur18].

Both versions of Algorithm 7, MOMIX and MOMIX$_{light}$, have been implemented in MATLAB R2018a, [MAT18a]. All experiments have been performed on a computer with Intel(R) Core(TM) i5-7400T CPU and 16 Gbytes RAM on operating system Windows 10 Enterprise.

6.3.1 Branching Rules

In our numerical experiments, we make use of two different branching rules. Both rules are based on the idea of partitioning boxes considering first the largest edges, giving priority to the integer variables in two different ways.

Let $\tilde{X} = [\tilde{l}, \tilde{u}]$ be a subbox of X. We consider the following two sets of indices in order to identify the branching variable $\hat{\imath} \in \{1, \dots, n\}$:

(br1) $J_1 = \text{argmax}\{\tilde{u}_i - \tilde{l}_i \mid i \in I\}$. If $\tilde{u}_i - \tilde{l}_i = 0$ for all $i \in I$, i. e., if all the integer variables are fixed, define $J_1 = \text{argmax}\{\tilde{u}_i - \tilde{l}_i \mid i \in \{1, \ldots, n\} \setminus I\}$. Choose $\hat{\imath} \in J_1$.

(br2) $J_2 = \text{argmax}\{\tilde{u}_i - \tilde{l}_i \mid i \in \{1, ..., n\}\}$. If $J_2 \cap I \neq \emptyset$ holds, choose $\hat{\imath} \in J_2 \cap I$. Otherwise, choose $\hat{\imath} \in J_2$.

The first strategy is standard in mixed integer procedures: the integer variables are fixed at first. The second strategy aims to reduce the largest edge of the boxes, no matter whether it is related to an integer variable or not. Only if there is more than one largest edge and one of them belongs to an integer variable, we prefer to branch at this variable. We will show that this second non-standard branching rule performs better on some of the test instances.

Once the branching variable $\hat{\imath} \in \{1, \ldots, n\}$ has been selected, we partition the box $\tilde{X}$ into two boxes $\tilde{X}_1, \tilde{X}_2$ as follows: We set for $c_1, c_2 \in [\tilde{l}_{\hat{\imath}}, \tilde{u}_{\hat{\imath}}]$:

$$\begin{aligned} \tilde{X}_1 &:= \left[\tilde{l}, \left(\tilde{u}_1, ..., \tilde{u}_{\hat{\imath}-1}, c_1, \tilde{u}_{\hat{\imath}+1}, ..., \tilde{u}_n\right)^T\right] \\ \text{and} \quad \tilde{X}_2 &:= \left[\left(\tilde{l}_1, ..., \tilde{l}_{\hat{\imath}-1}, c_2, \tilde{l}_{\hat{\imath}+1}, ..., \tilde{l}_n\right)^T, \tilde{u}\right]. \end{aligned}$$

Thereby, we differentiate between $\hat{\imath} \in I$ and $\hat{\imath} \notin I$. If $\hat{\imath} \notin I$, we set $c_1 = c_2 = (\tilde{l}_{\hat{\imath}} + \tilde{u}_{\hat{\imath}})/2$. If $\hat{\imath} \in I$, we set $c_1 = \lfloor (\tilde{l}_{\hat{\imath}} + \tilde{u}_{\hat{\imath}})/2 \rfloor$ and $c_2 = \lceil (\tilde{l}_{\hat{\imath}} + \tilde{u}_{\hat{\imath}})/2 \rceil$. If $\tilde{l}_{\hat{\imath}} + \tilde{u}_{\hat{\imath}}$ is an even number, in order to avoid $\tilde{X}_1 \cap \tilde{X}_2 \neq \emptyset$, we split considering $c_1 = (\tilde{l}_{\hat{\imath}} + \tilde{u}_{\hat{\imath}})/2$ and $c_2 = 1 + (\tilde{l}_{\hat{\imath}} + \tilde{u}_{\hat{\imath}})/2$. Note that such a bisection excludes the infeasible part between c_1 and c_2.

As already mentioned in the introduction, we assume $X = [l, u] \subset \mathbb{R}^n$ with $l_i, u_i \in \mathbb{Z}$ for all $i \in I$. Then, for all subboxes $\tilde{X}$ obtained by any of the branching rules presented, it holds $\tilde{l}_i, \tilde{u}_i \in \mathbb{Z}$ for all $i \in I$. Furthermore, it is easy to see that both branching rules adopted in `MOMIX` and $\texttt{MOMIX}_{light}$ satisfy Assumption A.1.

In order to clarify the differences between the two rules (br1) and (br2), we present the results obtained by `MOMIX` applied to the following:

Test instance T.23 We study the biobjective mixed integer instance with two variables:

$$\begin{aligned} \min \quad & \begin{pmatrix} x_1 + x_2 \\ x_1^2 + x_2^2 \end{pmatrix} \\ \text{s.t.} \quad & (x_1 - 2)^2 + (x_2 - 2)^2 \leq 36 \\ & x_1 \in [-2, 2] \\ & x_2 \in [-4, 4] \cap \mathbb{Z}. \end{aligned}$$

In Figures 6.5b and 6.5d, we show in gray the image of $X^{g,\mathbb{Z}}$ under the objective functions. In black we give the set $\mathcal{L}_{PNS}$ obtained by applying MOMIX with (br1) and (br2), respectively. Note that MOMIX is able to find in both cases a good approximation of the non-connected nondominated set of the instance.

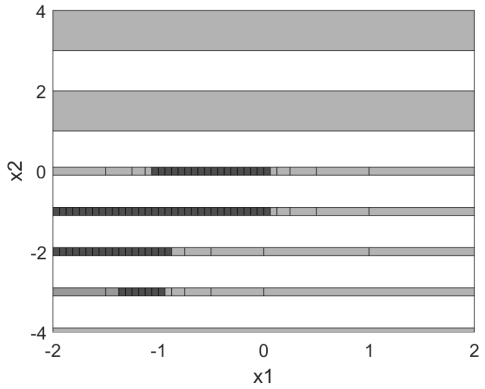
Both branching rules explore the whole feasible set of Test instance T.23. Even while they partition the box X in different ways, the outputs of MOMIX are very similar, i. e., with (br1) and (br2) the boxes in the solution list $\mathcal{L}_S$ and the list of upper bounds $\mathcal{L}_{PNS}$ are nearly the same.

6.3.2 Results on Scalable Instances

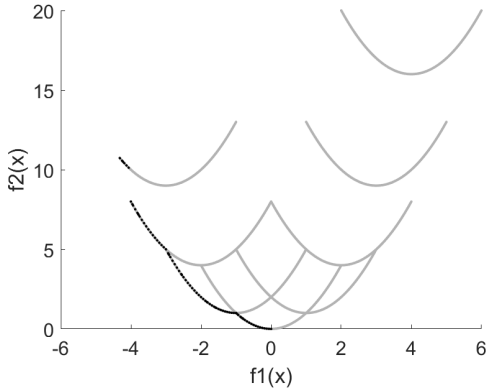
In this section, we show results on three different test instances of ($MOMIC$), all scalable in the number of variables. We apply MOMIX and MOMIX$_{light}$ in combination with (br1) and (br2) on all instances. We analyze the impact of the branching rules as well as the difference between MOMIX and MOMIX$_{light}$. Recall that MOMIX uses stronger lower bounds but these require to solve single objective mixed integer convex optimization problems.

Test instance T.24 This instance has quadratic objective functions and the number of integer variables can be set to different values. Let the matrices Q_1 and Q_2 be defined as follows:

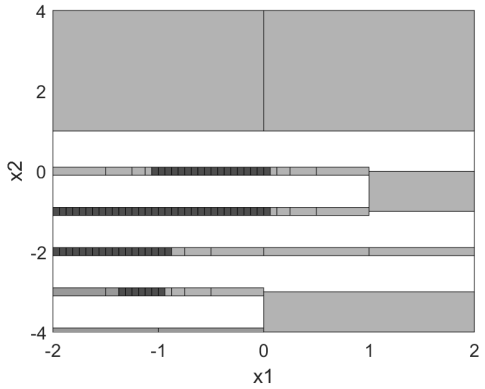
$$(Q_1)_{i,j} = \begin{cases} 3 & \text{if } i = j = 1 \\ 4 & \text{if } i = j = n \\ 1 & \text{else} \end{cases} \qquad \text{and } (Q_2)_{i,j} = \begin{cases} 2 & \text{if } i = j = 1 \text{ or } i = j = n \\ 4 & \text{if } i = j \text{ and } i \notin \{1, n\} \\ 1 & \text{else.} \end{cases}$$



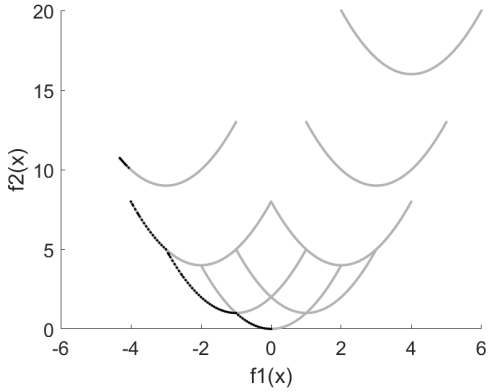
a: Partition of the box X obtained by applying MOMIX with (br1).



b: $\mathcal{L}_{PNS}$ obtained by applying MOMIX with (br1).



c: Partition of the box X obtained by applying MOMIX with (br2).



d: $\mathcal{L}_{PNS}$ obtained by applying MOMIX with (br2).

Figure 6.5. Results for Test instance T.23 obtained by applying MOMIX with different branching rules.

Then the optimization problem is stated by

$$\begin{aligned} \min \quad & \begin{pmatrix} x^T Q_1^T Q_1 x + (1, 2, \ldots, 2, 1)x \\ x^T Q_2^T Q_2 x + (-1, -2, \ldots, -2, 5)x \end{pmatrix} \\ \text{s. t.} \quad & x_i \in [-5, 5] \text{ for all } i = 1, \ldots, n \\ & I = \{3, \ldots, n\}. \end{aligned}$$

Note that $Q_1^T Q_1$ and $Q_2^T Q_2$ are positive semidefinite matrices, and hence f_1 and f_2 are convex functions.

Test instance T.25 This instance is also scalable in the number of integer variables.

$$\begin{aligned} \min \quad & \begin{pmatrix} x_1 \\ x_2 + \sum_{i=3}^{n} 10(x_i - 0.4)^2 \end{pmatrix} \\ \text{s. t.} \quad & \sum_{i=1}^{n} x_i^2 \leq 4 \\ & x_i \in [-2, 2] \text{ for all } i = 1, \ldots, n \\ & I = \{3, \ldots, n\} \end{aligned}$$

Here, we can explicitly give the set of all efficient points by

$$X_E = \{x \in \mathbb{R}^n \mid x_1^2 + x_2^2 = 4, x_1 \in [-2, 0], x_2 \in [-2, 0], x_i = 0 \text{ for all } i \geq 3\}.$$

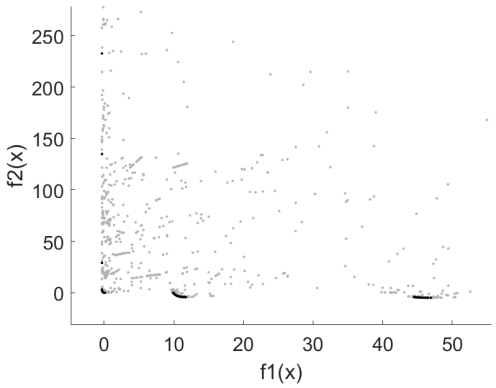
Test instance T.26 In this instance both the number of continuous and integer variables can be set to different values, with the restriction that $k^c = |\{1, \ldots, n\} \setminus I|$ has to be even.

$$\begin{aligned} \min \quad & \begin{pmatrix} \sum_{i=1}^{k^c/2} x_i + \sum_{i=k^c+1}^{n} x_i \\ \sum_{i=k^c/2+1}^{k^c} x_i - \sum_{i=k^c+1}^{n} x_i \end{pmatrix} \\ \text{s. t.} \quad & \sum_{i=1}^{k^c} x_i^2 \leq 1 \\ & x_i \in [-2, 2] \text{ for all } i = 1, \ldots, n \\ & I = \{k^c + 1, ..., n\} \end{aligned}$$

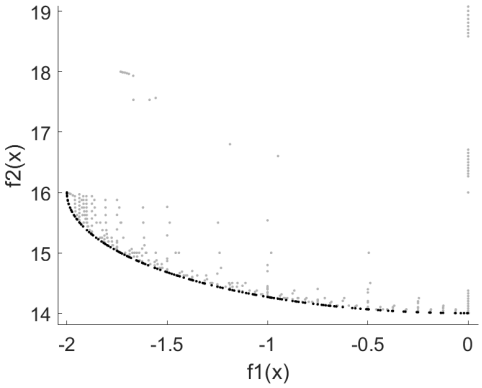
For both objective functions the Lipschitz constant is $L = \sqrt{k^c/2 + |I|}$.

For all instances but Test instance T.26 we set half an hour (1800 seconds) as a time limit. For Test instance T.26 we set the time limit to one hour (3600 seconds).

In Figures 6.6a, 6.6b and 6.7, we show our results in the image space. As the set $\mathcal{L}_{PNS}$ is similar for all versions of MOMIX and choices of the branching rule within one test instance, we present only the results for MOMIX with (br2) within the figures. In black we plot the points of $\mathcal{L}_{PNS}$. The gray points are the images of the feasible points, i. e., the upper bounds, computed along the algorithm. The parameter for the set from Theorem 6.8 applied to Test instance T.26 with $k^c = 2$ and $|I| = 1$ is $L\delta = 0.1\sqrt{2}$. Hence, the set described by $\bigcup_{z \in \mathcal{L}_{PNS}} (\{z - L\delta e\} + \mathbb{R}^m_+)$ is just a rough lower bound of the nondominated set. From a practical point of view, in all our test runs, the points from the lists $\mathcal{L}_{PNS}$ deliver a good approximation of the nondominated sets.



a: The set $\mathcal{L}_{PNS}$ of Test instance T.24 for $|I| = 5, n = 7$.



b: The set $\mathcal{L}_{PNS}$ of Test instance T.25 for $|I| = 10, n = 12$.

Figure 6.6. Results for Test instances T.24 and T.25 obtained by applying MOMIX.

The numerical results on all instances are shown in Table 6.1. In the first two columns we state the number of integer variables ($|I|$) and the number of continuous variables ($|C|$) for each instance. For both MOMIX and MOMIX$_{light}$ we report the total computational time in seconds ($t\,[s]$) and the number of considered boxes in the branching tree (#nod). For MOMIX we additionally show the total time needed by Gurobi to address the single objective mixed integer quadratic problems (MIQP). Failures, i. e., instances for which the time limit was exceeded, are marked with "-".

Table 6.1. Numerical results for Test instances T.24 to T.26.

		MOMIX						MOMIX$_{light}$			
		(br1)			(br2)			(br1)		(br2)	
$\|I\|$	$\|C\|$	$t\ [s]$	#nod	MIQP	$t\ [s]$	#nod	MIQP	$t\ [s]$	#nod	$t\ [s]$	#nod
Test instance T.24 - time limit 1800s											
1	2	40.1	757	2.3	38.7	765	2.3	849.9	609	524.5	669
2	2	30.8	537	1.6	31.6	575	1.7	667.2	555	563.0	641
3	2	31.0	535	1.5	30.8	521	1.5	1381.2	1127	814.4	917
4	2	34.7	567	1.7	65.6	1095	3.0	-	-	1134.9	1285
5	2	38.5	587	1.6	81.5	1259	3.2	-	-	-	-
10	2	350.3	2707	9.5	-	-	-	-	-	-	-
Test instance T.25 - time limit 1800s											
1	2	15.5	301	0.5	14.6	299	0.4	1045.4	299	1025.6	299
10	2	36.5	413	1.2	27.1	353	0.7	-	-	-	-
20	2	-	-	-	46.9	411	0.9	-	-	-	-
30	2	-	-	-	80.4	471	1.1	-	-	-	-
50	2	-	-	-	-	-	-	-	-	-	-
Test instance T.26 - time limit 3600s											
1	2	41.5	749	1.3	44.3	771	1.3	296.3	747	225.6	801
2	2	226.2	3683	6.3	240.5	3761	6.2	-	-	3090.4	3701
3	2	1354.9	19127	32.3	1321.5	18451	31.1	-	-	-	-
1	4	2199.5	23935	53.5	2246.6	24399	53.8	-	-	-	-

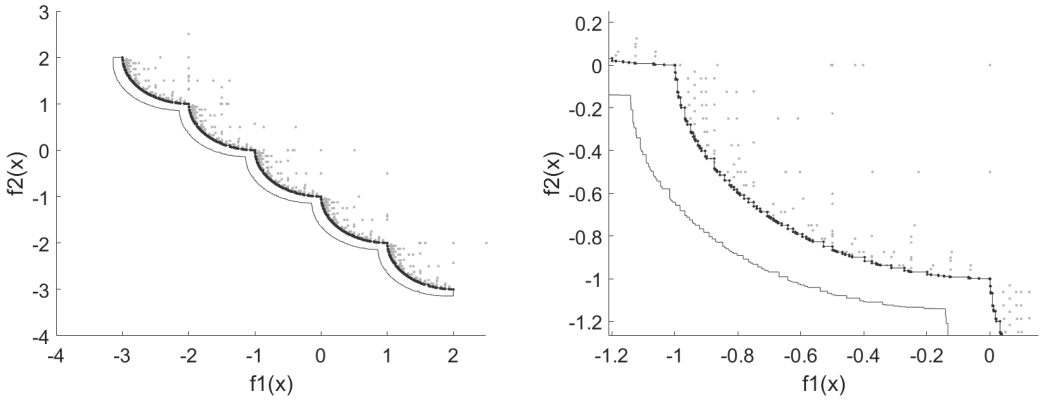


Figure 6.7. The set $\mathcal{L}_{PNS}$ of Test instance T.26 for $|I| = 1, n - |I| = 2$ and the boundary of the set from Theorem 6.8, $L\delta = 0.1\sqrt{2}$. Right picture shows a detail of the left one.

We observe that MOMIX outperforms MOMIX$_{light}$ on all test instances. MOMIX is able to solve a higher number of instances within the time limit. This seems to indicate that the improved lower bounding procedure of MOMIX and the effort in solving single objective mixed integer convex problems pays off.

We notice that the time Gurobi needs to address the single objective mixed integer subproblems is a small percentage of the whole computational time. By using the MATLAB profiler on our code, we figured out that the bottleneck in our implementation is `fmincon`: Most of the computational time was spent to solve the single objective continuous convex problems. In fact, for high dimensional test instances `fmincon` was not able to solve some of the single objective continuous convex problems. This was the case for, e. g., Test instance T.25 with $|I| = 50$. Note that `fmincon` can be replaced by any solver for convex problems within both, MOMIX and MOMIX$_{light}$.

Regarding the two branching rules, we can notice some differences as soon as the dimension of the instances grows. In most cases, MOMIX and MOMIX$_{light}$ with (br2) could solve instances with a larger number of integer variables.

6.3.3 Results on a Triobjective Instance

Our implementation of MOMIX and MOMIX$_{light}$ can handle instances of $(MOMIC)$ with a general number of objective functions $m \geq 2$. In the following, we present the results obtained by applying MOMIX with branching rule (br2).

Test instance T.27 We consider the triobjective mixed integer instance

$$\begin{aligned} \min \quad & \begin{pmatrix} x_1 + x_4 \\ x_2 - x_4 \\ x_3 + x_4^2 \end{pmatrix} \\ \text{s.t.} \quad & \sum_{i=1}^{3} x_i^2 \leq 1 \\ & x_i \in [-2, 2] \text{ for all } i = 1, \ldots, 4 \\ & x_4 \in \mathbb{Z}. \end{aligned}$$

We set $\delta = 0.5$ in MOMIX. In order to detect $\mathcal{L}_S$, the cover of the efficient set of Test instance T.27, MOMIX needed to explore 1237 nodes. This was done within 190 seconds.

In Figure 6.8 the points in $\mathcal{L}_{PNS}$ are plotted in black, giving an approximation of the nondominated set of Test instance T.27. In gray we plot the images of the feasible points computed along the algorithm.

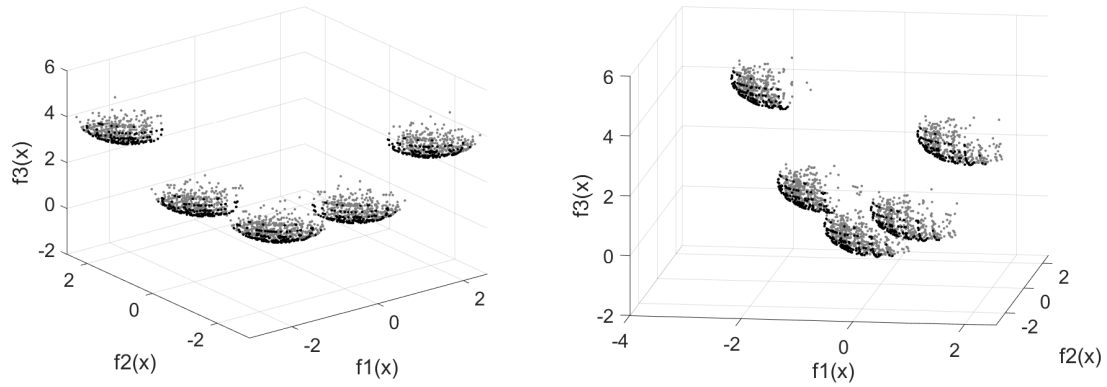


Figure 6.8. The set $\mathcal{L}_{PNS}$ for Test instance T.27 from two different perspectives.

6.3.4 Results on a Non-Quadratic Convex Instance

As a further example, we report the results obtained applying MOMIX$_{light}$ with branching rule (br1) on the following non-quadratic instance:

Test instance T.28

$$\begin{aligned} \min \quad & \begin{pmatrix} x_1 + x_3 \\ x_2 + \exp(-x_3) \end{pmatrix} \\ \text{s.t.} \quad & x_1^2 + x_2^2 \leq 1 \\ & x_i \in [-2, 2] \text{ for all } i = 1, \dots, 3 \\ & x_3 \in \mathbb{Z} \end{aligned}$$

Note that the second objective of Test instance T.28 is a convex non-quadratic function.

As already mentioned at the beginning of the section, in our implementation of MOMIX we use GUROBI, [Gur18] as mixed integer quadratic solver and we did not include any other solver within it. Therefore, in order to solve Test instance T.28 we applied MOMIX$_{light}$ setting $\delta = 0.1$. MOMIX$_{light}$ was able to detect $\mathcal{L}_S$ by addressing 1105 nodes within 20 seconds. In Figure 6.9, we plot the obtained approximation of the nondominated set of Test instance T.28.

Assume that Test instance T.28 is solved by using the ε-constraint method. The ε-constraint scalarization of Test instance T.28 for some $\varepsilon \in \mathbb{R}$ is then defined by

$$\begin{aligned} \min \quad & f_1(x) = x_1 + x_3 \\ \text{s.t.} \quad & f_2(x) = x_2 + \exp(-x_3) \leq \varepsilon \\ & x_1^2 + x_2^2 \leq 1 \\ & x_i \in [-2, 2] \text{ for all } i = 1, \dots, 3 \\ & x_3 \in \mathbb{Z}. \end{aligned} \tag{6.7}$$

Considering the gap in the nondominated set of Test instance T.28 (see Figure 6.9), solving (6.7) for all values ε in the interval $[3, 6]$ would lead to the same solution. The

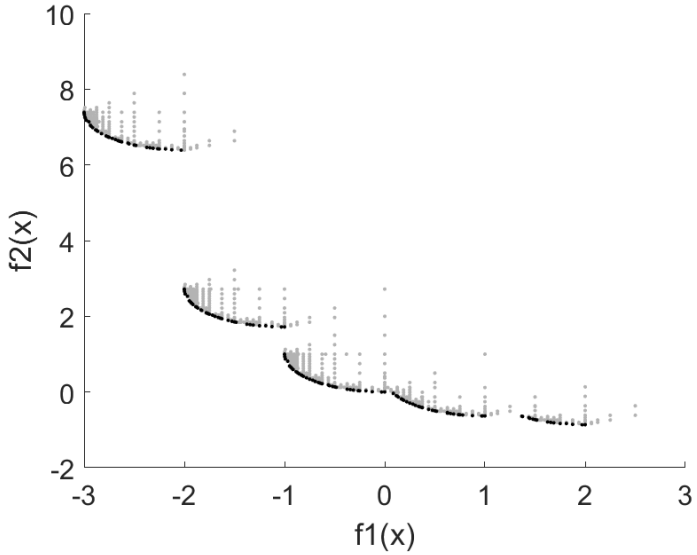


Figure 6.9. The set $\mathcal{L}_{PNS}$ of Test instance T.28 obtained by $\texttt{MOMIX}_{light}$.

significant values for ε are only those in the intervals $[-1, 3]$ and $[6, 7.5]$. Clearly, the significant intervals are not known in advance and this is a big issue when applying the ε-constraint method on ($MOMIC$), as the nondominated set of a multiobjective mixed integer convex problem may have huge gaps.

6.4 Conclusions

In this chapter, the first deterministic algorithm for solving multiobjective mixed integer convex programming problems is presented. The presentation is based on the paper [DeS+19] which is available online and submitted for publication. The method of the new algorithm MOMIX is based on linear outer approximations of the image set. We first build linear outer approximations of the convex relaxation of the problem by adaptively computing hyperplanes considering some meaningful local upper bounds. Then in some cases, we compute additional hyperplanes which outer approximate the convex hull of the true image set. This is again done in an adaptive way, taking specific local upper bounds into account. The local upper bound sets are updated as soon as a new upper bound is found and are used both to have a pruning criterion and to approximate the dominated set. Theoretical results related to the correctness of our

algorithm are provided. Numerical examples on biobjective and triobjective instances show the ability of our procedure to detect nondominated points of multiobjective mixed integer convex programming problems. We also explored the possibility of using two different branching rules. Each of these two are suitable for our new approach, but differences depending on the instance, which has to be solved, can be observed.

The algorithm MOMIX is similar to algorithm MOPBB introduced in Chapter 4. Basically, both are B&B algorithms dealing with different issues. While MOPBB has to handle nonconvex objective functions, MOMIX deals with continuous and integer variables. However, both algorithms use a discarding test which is based on the idea of Benson's outer approximation algorithm. Therefore, it might be of interest to combine both approaches to obtain a B&B algorithm which solves multiobjective mixed integer nonconvex optimization problems. The new procedure has to use the convex underestimators of the objective functions instead of taking them directly, because solving $(P_p(\tilde{X}^g))$ and $(MICP_p(\hat{\lambda}, \tilde{X}))$ globally needs too much computational time.

However, in case of an existing "fast" algorithm for mixed integer (single objective) nonconvex optimization problems, it might even be possible to solve $(MICP_p(\hat{\lambda}, \tilde{X}))$ directly in order to improve the hyperplane found by $(P_p(\tilde{X}^g))$ w. r. t. the convex underestimator. Then, it is of interest whether those two types of relaxations, using the convex underestimator and neglecting the integrality constraints in $(P_p(\tilde{X}^g))$, are still worth to obtain approximate solutions with a certain accuracy of this multiobjective mixed integer nonconvex optimization problem.

7 Solving Multiobjective Optimization Problems with Decision Uncertainty

This chapter handles multiobjective optimization problems with decision uncertainty, i. e., we have to deal with uncertainties concerning the realization of the decision variables. It comprises three sections, the first of which provides a brief introduction into decision uncertainty and the relationship to set optimization. In the second section we present the key features for a possible algorithm to solve multiobjective optimization problems with decision uncertainties. The overall algorithm is stated in the last part, together with the presentation and discussion of some numerical results. The results of this chapter are based on [ENR19].

7.1 Specific Preliminaries for Multiobjective Optimization with Decision Uncertainty

Recall the multiobjective optimization problem (P). For that, let a nonempty feasible set $M \subseteq \mathbb{R}^n$ and twice continuously differentiable functions $f_j \colon M \to \mathbb{R},\ j = 1, \ldots, m$ be given.

$$\begin{aligned} \min \quad & f(x) \\ \text{s. t.} \quad & x \in M \end{aligned} \tag{P}$$

Additionally, we assume the objective functions f_j to be factorable for all $j = 1, \ldots, m$, see Definition 3.2. Next, we introduce the relevant notions from multiobjective optimization with decision uncertainty, before we shortly point out the relations to the more general class of set optimization problems.

7.1.1 Decision Uncertainty

As described in the introduction, it is possible that the realization of a decision variable x is associated with uncertainties. These uncertainties will be modeled by a convex and compact set $Z \subseteq \mathbb{R}^n$ with $0 \in Z$. Hence, instead of x it might happen that $x+z$ for some $z \in Z$ is realized due to the uncertainties. We follow the notation as introduced in [EKS17].

In view of robustness, only feasible points $x \in M$ which are feasible for any uncertainty $z \in Z$ are of interest. This reduces the feasible set to the set of *decision robust feasible points* defined by

$$S := \{x \in M \mid x + z \in M \text{ for all } z \in Z\}.$$

It is a well-known challenge in robust optimization to calculate S from M and Z, also in the single objective case. Here, we assume that S is known and nonempty and we only concentrate on those challenges which are due to the multiple objectives. Moreover, we assume that S is convex and that there exists a box $X \in \mathbb{IR}^n$ which contains S. We require these properties because single objective subproblems with a convex objective function over the set S have to be solved. For being able to solve those problems globally and efficiently, we need the assumption of convexity of S. As our algorithm works with subdivisions in the preimage space, we also need the structure of the box.

For defining decision robust efficient solutions, we have to take all possible realizations of each x into account. The set of all these realizations for one x is given by $\{x + z \mid z \in Z\}$. As we are interested in the values of the objective functions, we have to compare the sets

$$f_Z(x) := \{f(x + z) \in \mathbb{R}^m \mid z \in Z\} \text{ for all } x \in S,$$

which define a set-valued map $f_Z \colon S \rightrightarrows \mathbb{R}^m$.

In case of linear functions f_j we have $f_Z(x) = \{f(x)\} + \{f(z) \in \mathbb{R}^m \mid z \in Z\}$. This simplifies the problem significantly, see [EKS17, Theorem 23] or [EG19]. If the

functions f_j and the set Z are convex, the sets $f_Z(x)$ for $x \in S$ do not have to be convex, as the following example shows:

Example 7.1 Let the two functions $f_1, f_2 \colon \mathbb{R}^2 \to \mathbb{R}$ with $f_1(x) = x_1$, $f_2(x) = x_1^2 + x_2^2$, and the set $Z = [-0.1, 0.1] \times [0, 0.1]$ be given. Then for $x = 0$, the set

$$f_Z(0) = \{f(0+z) \mid z \in Z\} = \left\{ \begin{pmatrix} z_1 \\ z_1^2 + z_2^2 \end{pmatrix} \in \mathbb{R}^2 \;\middle|\; z \in Z \right\}$$

contains the two points $y^1 = (-0.1, 0.02)$ and $y^2 = (0.1, 0.02)$, but not the point $0.5y^1 + 0.5y^2 = (0, 0.02)$. This is because $f(0+z) = (0, 0.02)$ only holds for the points $z = (0, -\sqrt{0.02})$ or $z = (0, \sqrt{0.02})$. But in both cases it is $z \notin Z$. Hence, the set $f_Z(0)$ is not convex.

We define the difference of two sets $A, B \subseteq \mathbb{R}^m$ by

$$A - B := \{a - b \mid a \in A, b \in B\}.$$

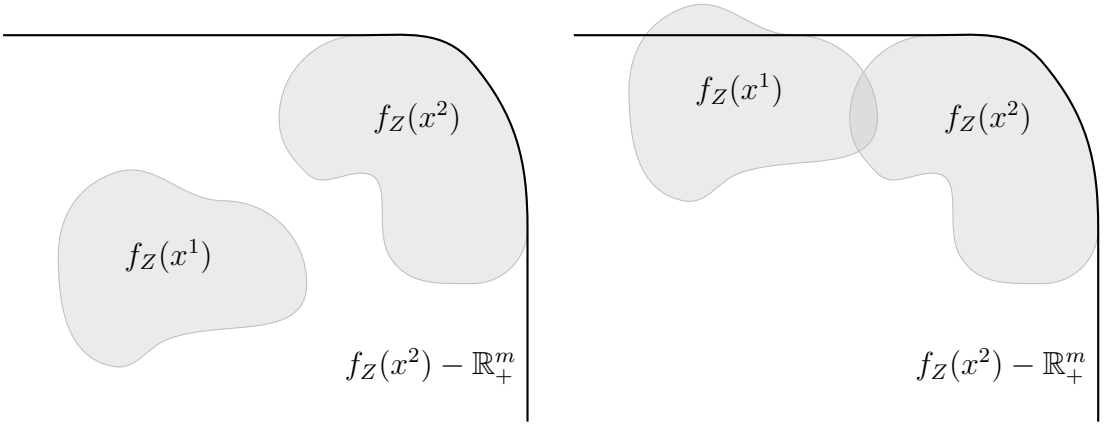
Due to [EKS17], we introduce the following optimality concept for the problem (P) with respect to decision uncertainty given by the set Z. It is motivated by the definition of optimality for single objective decision uncertain optimization as well as by the definition of optimality in parameter uncertain multiobjective optimization.

Definition 7.2 [EKS17] A point $x^* \in S$ is called a *decision robust strictly efficient solution* of (P) w. r. t. Z if there is no $x \in S \setminus \{x^*\}$ with the property

$$f_Z(x) \subseteq f_Z(x^*) - \mathbb{R}^m_+.$$

We illustrate this definition with the following example:

Example 7.3 Assume $S = \{x^1, x^2\}$, and Z as well as $f \colon S \to \mathbb{R}^2$ are in such a way that the sets $f_Z(x^1)$ and $f_Z(x^2)$ look as in Figures 7.1a and 7.1b, respectively. Then for the situation in Figure 7.1a, only x^1 is a decision robust strictly efficient solution w. r. t. Z, while for Figure 7.1b both points, x^1 and x^2, are decision robust strictly efficient solutions.



a: First case. **b:** Second case.

Figure 7.1. Illustration for Example 7.3.

Note that in the case of $Z = \{0\}$, i. e., the case of no uncertainty, there can be efficient solutions of (P) that are not decision robust strictly efficient solutions of (P) w. r. t. Z. This is due to the fact that decision robust strict efficiency is a generalization of strict efficiency for multiobjective optimization problems, see Definition 2.4 (ii). In contrast to efficiency, strict efficiency of a point x^* does not allow the existence of another point x with $f(x) = f(x^*)$. For more details and a more detailed example we refer to [EKS17, Sect. 2].

For applying Definition 7.2 in an algorithm, one has to be able to verify numerically whether $A \subseteq B - \mathbb{R}^m_+$ holds for two sets $A, B \subseteq \mathbb{R}^m$. In case of polyhedral sets this can be done by using a finite number of support functionals, see [Jah15]. In case of arbitrary closed convex sets one might need an infinite number of such linear functionals. Additionally, one has to solve a minimizing problem for each of these functionals and for each of the sets to decide whether the subset condition holds. This is based on the equivalence

$$A \subseteq B - \mathbb{R}^m_+ \qquad \Leftrightarrow \qquad \forall\, \ell \in \mathbb{R}^m_+ \colon \ \sup_{a \in A} \ell^T a \leq \sup_{b \in B} \ell^T b,$$

which was formulated in a more general setting in [Jah13, Theorem 2.1].

7.1.2 Relation to Set Optimization

In set optimization one studies set-valued optimization problems. Solving the following set optimization problem is closely related to determining decision robust strictly efficient solutions.

$$\begin{aligned} \min \quad & f_Z(x) = \{f(x+z) \mid z \in Z\} \\ \text{s.t.} \quad & x \in S \end{aligned} \tag{SOP}$$

There are different approaches to define optimal solutions for problems like (SOP). For an introduction to set optimization, see the book [KTZ15]. In case of the set approach, see [JH11], one uses order relations to compare the sets which are the images of the objective function. As we are minimizing in our multiobjective optimization problem, the "maximal elements" of a set are in some sense the worst elements, which fits to our worst-case approach. Comparing the "maximal elements" of a set corresponds to the upper-less order relation, which we introduce next.

Definition 7.4 [JH11] Let $A, B \subseteq \mathbb{R}^m$ be two nonempty sets. The *upper-type (u-type) less order relation* $\preccurlyeq_u$ is defined by:

$$A \preccurlyeq_u B \;\Leftrightarrow\; A \subseteq B - \mathbb{R}^m_+.$$

For nonempty sets $A, B \subseteq \mathbb{R}^m$, this characterization is equivalent to

$$A \preccurlyeq_u B \Leftrightarrow \forall a \in A \; \exists b \in B : a \leq b.$$

Moreover, the upper-type less order relation is reflexive and transitive, but it is in general not anti-symmetric. However, it holds

$$A \preccurlyeq_u B \text{ and } B \preccurlyeq_u A \Leftrightarrow A - \mathbb{R}^m_+ = B - \mathbb{R}^m_+.$$

Now we can state the definition of an optimal solution of a set-valued optimization problem as (SOP). For this, we use a definition which was formulated in [RS07]:

Definition 7.5 Let a nonempty set $X \subseteq \mathbb{R}^n$ and a set-valued map $H : X \rightrightarrows \mathbb{R}^m$ be given, where $H(x) \neq \emptyset$ holds for all $x \in X$. A point $x^* \in X$ is called a strictly optimal solution of the set optimization problem

$$\min_{x \in X} H(x)$$

w. r. t. $\preccurlyeq_u$ if there exists no $x \in X \setminus \{x^*\}$ with $H(x) \preccurlyeq_u H(x^*)$.

In our case, we have the special set-valued map described by $f_Z \colon S \rightrightarrows \mathbb{R}^m$ with $f_Z(x) = \{f(x+z) \mid z \in Z\}$. Obviously, a point $x^* \in S$ is a decision robust strictly efficient solution of (P) w. r. t. Z if and only if $x^* \in S$ is a strictly optimal solution of the set optimization problem (SOP). Hence, we present in this chapter an algorithm to calculate a covering of the set of strictly optimal solutions of a specific set optimization problem. The proposed techniques can also be used to develop algorithms for more general set optimization problems. We discuss this in Section 7.4 briefly.

A basic technique of our algorithm will be a B&B approach. Lower and upper bounds will be important for the bounding step. The next definition clarifies these terms.

Definition 7.6 Let $A \subseteq \mathbb{R}^m$ be a nonempty set.

(i) A set $U \subseteq \mathbb{R}^m$ is called an upper bound set/upper bound for A if $A \preccurlyeq_u U$.

(ii) A set $L \subseteq \mathbb{R}^m$ is called a lower bound set/lower bound for A if $L \preccurlyeq_u A$.

7.2 Algorithmic Approach

Our algorithm uses the concept of a B&B method. The branching will be in the preimage space $\mathbb{R}^n$. We have assumed that there is a box X which contains the convex feasible set S. This is, for instance, the case when S is given by convex inequality constraints and by lower and upper bounds for each variable. We start with the box X and partition it along the longest edge into two subboxes. See, for instance, Section 4.2 for a more detailed description of such a partitioning. On each subbox X^*, we test whether a sufficient criteria is satisfied which guarantees that $X^* \cap S$ does not contain any decision robust strictly efficient solution of (P) w. r. t. Z. If such a criterion

is satisfied, we do not consider this subbox and the feasible points in $X^* \cap S$ anymore. Otherwise, we partition the box until all boxes are either discarded or smaller than a predefined value.

For such a B&B scheme, a good criterion for discarding a box is essential. These criteria are in general based on lower bounds obtained on the subboxes and on upper bounds obtained within the procedure. In single objective global (i. e., nonconvex) optimization, the upper bounds are function values of feasible solutions. The lower bounds are bounds for all possible values of the objective over a subbox which are determined by interval arithmetic or by other underestimation methods. Hence, just scalars have to be compared. In our setting, already a "function value" $f_Z(x)$ for some feasible point x is a whole set. As these lower and upper bounds have to be compared frequently within such an algorithm, we will present a way to avoid to compare sets as $f_Z(x)$ for some x directly. We will present replacements (i. e., sufficient conditions) using sets which have a very simple structure. To distinguish between upper and lower bound sets in this chapter, the corresponding variables x, z and subboxes or subsets of X and Z are indicated with $\tilde{\cdot}$ or $\cdot^*$, respectively. This means that an upper bound set of $f_Z(\tilde{x})$ is computed with respect to a fixed point $\tilde{x} \in \tilde{X}$ and a lower bound set is determined for all sets $f_Z(x^*) = \{f(x^* + z) \mid z \in Z\}$ with $x^* \in X^*$. Note that determining a lower bound L for all sets $f_Z(x^*)$ with $x^* \in X^*$ (i. e., $L \preccurlyeq_u f_Z(x^*)$ for all $x^* \in X^*$) is not equivalent to simply determining a lower bound for $f_Z(X^*) := \bigcup_{x^* \in X^*} \{f(x^* + z) \mid z \in Z\}$ (i. e., with $L \preccurlyeq_u f_Z(X^*)$).

As the objective functions f_i, and also the sets $f_Z(x)$, are not necessarily convex, we will use the concept of convex underestimators for being able to formulate such replacements and a numerically tractable sufficient condition finally.

7.2.1 Concave Overestimators

As shown in Example 7.1, the sets $f_Z(x)$ for $x \in S$ may be nonconvex, even in case of convex functions $f_i \colon S \to \mathbb{R}$. For that reason, we will make use of the concept of convex underestimators and concave overestimators, respectively, depending on whether we aim at lower or at upper bounds. In Section 3.2 we already got familiar

with convex underestimators. Now, we need the converse — concave overestimators — additionally.

Definition 7.7 Let a function $h\colon \mathbb{R}^n \to \mathbb{R}$ on a box $X = [\underline{x}, \overline{x}] \in \mathbb{IR}^n$ be given. A function $\overline{h}\colon X \to \mathbb{R}$ is called *concave overestimator* for h on X if $\overline{h}$ is a concave function with $\overline{h}(x) \geq h(x)$ for all $x \in X$.

Let a box be defined by $X = [\underline{x}, \overline{x}] \in \mathbb{IR}^n$. Recall that we obtain a convex underestimator $\underline{h}$ of a smooth function h by

$$\underline{h}(x) := h(x) + \frac{\alpha_h}{2}(\underline{x} - x)^T(\overline{x} - x),$$

where $\alpha_h \geq \max\{0, -\min_{x \in X} \lambda_{\min,h}(x)\}$. Here, $\lambda_{\min,h}(x)$ denotes the smallest eigenvalue of the Hessian $H_h(x)$ of h in x, [MF94].

We use convex underestimators to be able to calculate the elements of sets $L \subseteq \mathbb{R}^m$ numerically such that

$$L \subseteq f_Z(x) - \mathbb{R}^m_+ \ (\Leftrightarrow\ L \preccurlyeq_u f_Z(x))$$

holds for all $x \in X^* \cap S$ for some subbox X^* of X.

Also concave overestimators of a function h on X are of interest. If we calculate a convex underestimator of the function $-h$ as described above, i. e.,

$$\underline{-h}(x) := -h(x) + \frac{\alpha_{-h}}{2}(\underline{x} - x)^T(\overline{x} - x),$$

where $\alpha_{-h} \geq \max\{0, -\min_{x \in X} \lambda_{\min,-h}(x)\}$, the function $\overline{h} := -(\underline{-h})$ is such a concave overestimator of h on X. We use such concave overestimators to be able to calculate sets $U \subseteq \mathbb{R}^m$ numerically such that

$$f_Z(\tilde{x}) \subseteq U - \mathbb{R}^m_+ \ (\Leftrightarrow\ f_Z(\tilde{x}) \preccurlyeq_u U)$$

holds for some given $\tilde{x} \in S$. The advantage is that while it might be numerically difficult to compare the lower bound L of all sets $f_Z(x)$ for all $x \in X^* \cap S$ with $f_Z(\tilde{x})$, it might be much easier to compare L with U. Note that we have

$$U \subseteq L - \mathbb{R}^m_+ \;\Rightarrow\; f_Z(\tilde{x}) \subseteq f_Z(x) - \mathbb{R}^m_+ \text{ for all } x \in X^* \cap S.$$

When we use the set order relation as defined in Definition 7.4 we can write this equivalently as

$$\begin{aligned} U \preccurlyeq_u L \;&\Rightarrow\; f_Z(\tilde{x}) \preccurlyeq_u U \preccurlyeq_u L \preccurlyeq_u f_Z(x) \text{ for all } x \in X^* \cap S \\ &\Rightarrow\; f_Z(\tilde{x}) \preccurlyeq_u f_Z(x) \text{ for all } x \in X^* \cap S. \end{aligned} \tag{7.1}$$

The implication holds as $\preccurlyeq_u$ is a transitive set order relation. Thus, if $\tilde{x} \in S \setminus X^*$ holds, we can discard the subbox X^* as it does not contain any decision robust strictly efficient solution of (P) w. r. t. Z.

7.2.2 Upper Bound Sets

First, we describe how to calculate a simple set U with $f_Z(\tilde{x}) \preccurlyeq_u U$, where $\tilde{x}$ is a fixed feasible point of a given box $\tilde{X}$. To begin with, we explain how to construct such a set U which is a singleton. Then we describe how this upper bound can be improved by using outer approximations as known from convex multiobjective optimization.

For deriving a singleton upper bound for a set, we use the anti-ideal point, see also Definition 3.16 (ii). For the set $f_Z(\tilde{x}) = \{f(\tilde{x}+z) \mid z \in Z\}$, this is the point $\overline{a}$ defined by

$$\overline{a}_j := \max_{y \in f_Z(\tilde{x})} y_j = \max_{z \in Z} f_j(\tilde{x}+z) \text{ for all } j = 1, \ldots, m.$$

Hence, the anti-ideal point can easily be determined if f_j is a concave function for $j = 1, \ldots, m$, as Z is assumed to be a convex and compact set and the functions f_j are twice continuously differentiable. One can apply any solution method from single objective constrained optimization, for instance, an algorithm which uses SQP.

However, if f_j is not concave, such a local solution method as SQP might only deliver a locally maximal solution and not a globally maximal one. In that case we use the concave overestimators which were introduced in Section 7.2.1. The result is summarized in the next lemma. This lemma needs a box $\hat{Z}$ with $Z \subseteq \hat{Z}$. Recall that Z was assumed to be a compact convex set and thus such a set $\hat{Z}$ can easily be calculated. The reason for the assumption of a box is that we can determine concave overestimators only over boxes, as explained in Section 7.2.1. With Remark 3.13 it follows that a small box $\hat{Z}$ leads to a tighter concave overestimator. Therefore, $\hat{Z}$ should be chosen as small as possible.

Lemma 7.8 *Let $\hat{Z} \in \mathbb{IR}^n$ be a box with $Z \subseteq \hat{Z}$, and let $\tilde{x} \in S$ be given. Let $\overline{f_j}$ be the concave overestimator of f_j on the box $\{\tilde{x}\} + \hat{Z}$ for all $j = 1, \ldots, m$ as defined in Definition 7.7. The singleton set U with*

$$U := \{\bar{p}\} \text{ with } \bar{p} := (\max_{z \in Z} \overline{f_1}(\tilde{x} + z), \ldots, \max_{z \in Z} \overline{f_m}(\tilde{x} + z))^T \tag{7.2}$$

is an upper bound set for $f_Z(\tilde{x})$, i. e., $f_Z(\tilde{x}) \preccurlyeq_u U$.

Proof. For the proof of this lemma, we show $f_Z(\tilde{x}) \subseteq U - \mathbb{R}^m_+$. Therefore, let $w \in f_Z(\tilde{x})$ be arbitrary, i. e., there is a $z \in Z$ with $w = f(\tilde{x} + z)$. As $\overline{f_j}$ is a concave overestimator of f_j on $\{\tilde{x}\} + \hat{Z}$ and $z \in Z \subseteq \tilde{Z}$, we obtain for every $j = 1, \ldots, m$:

$$w_j = f_j(\tilde{x} + z) \leq \overline{f_j}(\tilde{x} + z) \leq \max_{z \in Z} \overline{f_j}(\tilde{x} + z).$$

Therefore, $w \in U - \mathbb{R}^m_+$ holds. □

The optimization problems in (7.2) have a convex and compact feasible set and twice continuously differentiable concave objective functions, which are maximized. Thus, they can be solved, for instance, with an SQP method. Lemma 7.8 uses that it holds for the set

$$\overline{f_Z}(\tilde{x}) := \{(\overline{f_1}(\tilde{x} + z), \ldots, \overline{f_m}(\tilde{x} + z))^T \mid z \in Z\}$$

that $f_Z(\tilde{x}) \subseteq \overline{f_Z}(\tilde{x}) - \mathbb{R}^m_+$ and that $\overline{f_Z}(\tilde{x}) - \mathbb{R}^m_+$ is a convex set. Thus, the anti-ideal point of $\overline{f_Z}(\tilde{x})$ can be calculated by known local methods for single objective optimization. Figure 7.2 shows how to obtain the upper bound set U.

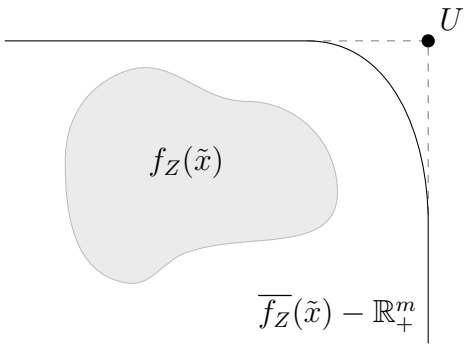


Figure 7.2. The set $f_Z(\tilde{x})$ and its singleton upper bound set U according to Lemma 7.8.

This rough upper bound can be improved by using outer approximation techniques from convex multiobjective optimization. These can be applied as the set $\overline{f_Z}(\tilde{x}) - \mathbb{R}^m_+$ is a convex set. The algorithm which we are using is called Benson's outer approximation algorithm, extensively introduced in Section 3.3. One step depends on one fixed point $\bar{p}$, which can, for instance, be the anti-ideal point. With the idea of Benson's outer approximation algorithm we adapt $(P_{\bar{p},\,M})$ to the single objective convex optimization problems of the following type:

$$\begin{aligned} \min_{(z,t)\in\mathbb{R}^n\times\mathbb{R}} \quad & t \\ \text{s. t.} \quad & z \in Z, \\ & \bar{p} - te \leq \overline{f}(\tilde{x} + z). \end{aligned} \tag{$P_{\tilde{x},\bar{p}}$}$$

Let $(\tilde{z}, \tilde{t})$ be an optimal solution of $(P_{\tilde{x},\bar{p}})$ and let $\tilde{\lambda} \geq 0$ be a Lagrange multiplier to the constraint $\bar{p} - te \leq \overline{f}(\tilde{x} + z)$. Then the set

$$\{y \in \mathbb{R}^m \mid \tilde{\lambda}^T y = \tilde{\lambda}^T(\bar{p} - \tilde{t}e)\}$$

describes a supporting hyperplane of $\overline{f_Z}(\tilde{x})$, see also Section 3.3 for a general proof. If no Lagrange multiplier is available, a single objective linear optimization problem can be solved to calculate a normal vector $\tilde{\lambda}$ of the supporting hyperplane, see [ESS11].

Note that the anti-ideal point of $\overline{f_Z}(\tilde{x})$ gives m supporting hyperplanes of $\overline{f_Z}(\tilde{x})$ by

$$\{y \in \mathbb{R}^m \mid y_j = \max_{z\in Z} \overline{f_j}(\tilde{x} + z)\} \tag{7.3}$$

for every $j = 1, \ldots, m$. Several such supporting hyperplanes can be constructed to various points $\bar{p}$ with the help of $(P_{\tilde{x},\bar{p}})$. In our numerical experiments we limited ourselves to the hyperplane which we obtain by solving $(P_{\tilde{x},\bar{p}})$ for $\bar{p}$ as the anti-ideal point of $\overline{f_Z}(\tilde{x})$ and to those which we obtain directly from the anti-ideal point of $\overline{f_Z}(\tilde{x})$, see (7.3).

Adding more hyperplanes to get a better outer approximation is possible, but then even more single objective convex optimization problems have to be solved. Moreover, the calculation of the intersection of these hyperplanes gets more challenging. Also steering the calculation of the hyperplanes within the algorithm by an adaptive choice of the points $\bar{p}$ in $(P_{\tilde{x},\bar{p}})$ is an interesting approach for further improvements of the proposed algorithmic approach. Such an approach was, for instance, followed for nonconvex multiobjective optimization problems in Chapter 4.

The next lemma gives a summary of the construction of our upper bound set, which is also illustrated in Figure 7.3.

Lemma 7.9 *Let $\hat{Z} \in \mathbb{IR}^n$ be a box with $Z \subseteq \hat{Z}$ and let $\tilde{x} \in S$ be given. Let $\overline{f_j}$ be the concave overestimator of f_j on the box $\{\tilde{x}\} + \hat{Z}$ for all $j = 1, \ldots, m$ as defined in Definition 7.7. Furthermore, let $\bar{p} \in \mathbb{R}^m$ be the anti-ideal point of the concave overestimators $\overline{f} = (\overline{f_1}, \ldots, \overline{f_m})^T$ on $\{\tilde{x}\} + Z$, see Lemma 7.8. Moreover, let $(\tilde{z}, \tilde{t})$ be a minimal solution of $(P_{\tilde{x},\bar{p}})$ with Lagrange multiplier $\tilde{\lambda} \geq 0$ to the constraint $\bar{p} - te \leq \overline{f}(\tilde{x} + z)$. Then the set U with*

$$U := \{y \in \mathbb{R}^m \mid y_j \leq \bar{p}_j,\ j = 1, \ldots, m,\ \tilde{\lambda}^T y = \tilde{\lambda}^T(\bar{p} - \tilde{t}e)\} \tag{7.4}$$

is an upper bound set for $f_Z(\tilde{x})$, i. e., $f_Z(\tilde{x}) \preccurlyeq_u U$.

Proof. We denote the hyperplane to which $\tilde{\lambda} \geq 0$ is the normal vector by

$$U^* := \{y \in \mathbb{R}^m \mid \tilde{\lambda}^T y = \tilde{\lambda}^T(\bar{p} - \tilde{t}e)\}.$$

As $(\tilde{z}, \tilde{t})$ is feasible for $(P_{\tilde{x},\bar{p}})$ and by the definition of $\bar{p}$, we get for each $j = 1, \ldots, m$:

$$\tilde{t} \geq \bar{p}_j - \overline{f_j}(\tilde{x} + \tilde{z}) \geq 0.$$

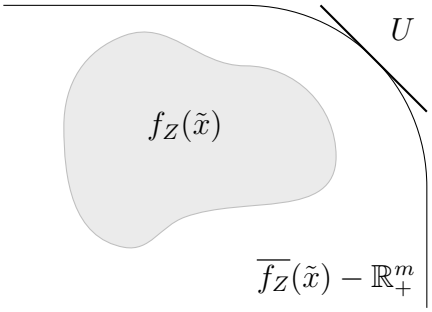


Figure 7.3. The set $f_Z(\tilde{x})$ and its upper bound set U according to Lemma 7.9.

As a consequence, we have $\bar{p} - \tilde{t}e \in U$. In particular, $\bar{p} - \tilde{t}e \in U^*$ holds and thus $\bar{p} \in U^* + \mathbb{R}^m_+$.

Next, we show that $f_Z(\tilde{x}) \subseteq U^* - \mathbb{R}^m_+$. This follows from the theory in Section 3.3 or [LRU14] as $(P_{\tilde{x},\bar{p}})$ is the same optimization problem which is solved to obtain a supporting hyperplane of $-\overline{f_Z}(\tilde{x})+\mathbb{R}^m_+$ with the ideal point $-\bar{p}$ (see Definition 3.16 for the definition of ideal point). There, U^* is by construction a supporting hyperplane of $-\overline{f_Z}(\tilde{x})+\mathbb{R}^m_+$, i. e., $\tilde{\lambda}^T(-\bar{p}+\tilde{t}e) \leq \tilde{\lambda}^T y$ holds for all $y \in -\overline{f_Z}(\tilde{x})+\mathbb{R}^m_+$. By multiplying with -1, this is equivalent to $\tilde{\lambda}^T(\bar{p} - \tilde{t}e) \geq \tilde{\lambda}^T y$ for all $y \in \overline{f_Z}(\tilde{x}) - \mathbb{R}^m_+$. Hence, U^* is a supporting hyperplane of $\overline{f_Z}(\tilde{x}) - \mathbb{R}^m_+$. As $\overline{f_j}$ is a concave overestimator of f_j on the box $\{\tilde{x}\} + \hat{Z}$ for all $j = 1, \ldots, m$, it follows:

$$f_Z(\tilde{x}) \subseteq \overline{f_Z}(\tilde{x}) - \mathbb{R}^m_+ \subseteq U^* - \mathbb{R}^m_+.$$

Now, let $w \in f_Z(\tilde{x})$ be arbitrarily chosen. Then

$$w \in U^* - \mathbb{R}^m_+ = \{y \in \mathbb{R}^m \mid \tilde{\lambda}^T y \leq \tilde{\lambda}^T(\bar{p} - \tilde{t}e)\}$$

and $\bar{p} \in U^* + \mathbb{R}^m_+ = \{y \in \mathbb{R}^m \mid \tilde{\lambda}^T y \geq \tilde{\lambda}^T(\bar{p} - \tilde{t}e)\}$ hold. Thus, there exists some $\mu \in [0, 1]$ with $y := \mu\bar{p} + (1 - \mu)w \in U^*$. By Lemma 7.8 we have $w \leq \bar{p}$. Therefore,

$$w \leq w + \mu(\bar{p} - w) = y = \mu\bar{p} + (1 - \mu)w \leq \bar{p}$$

and $w \leq y$ with $y \in U^* \cap (\{\bar{p}\} - \mathbb{R}^m_+) = U$. Hence, we derive $f_Z(\tilde{x}) \subseteq U - \mathbb{R}^m_+$. □

7.2.3 Lower Bound Sets

Recall, for a subbox X^* of X we denote the feasible set of X^* by $S^* := X^* \cap S$. For applying the implication (7.1), we propose a method to determine a set L with a simple structure, i. e., with

$$L \preccurlyeq_u f_Z(x) \text{ for all } x \in S^* \ (\Leftrightarrow\ L \subseteq f_Z(x) - \mathbb{R}^m_+ \text{ for all } x \in S^*).$$

"Simple structure" means that it should be easy computable and comparable with the obtained upper bounds. Similar to the anti-ideal point from Lemma 7.8, we can use here the ideal point of a set. While one could use the ideal point of the set

$$\{f(x+z) \in \mathbb{R}^m \mid x \in S^*,\ z \in Z\},$$

already the ideal point of any set

$$f(S^* + z^*) := \{f(x+z^*) \in \mathbb{R}^m \mid x \in S^*\}$$

for any $z^* \in Z$ delivers a lower bound:

Lemma 7.10 *Let $z^* \in Z$, X^* be a subbox of X, and let $a \in \mathbb{R}^m$ be defined by*

$$a_j := \min\{f_j(x+z^*) \in \mathbb{R} \mid x \in S^*\},\ j = 1, \dots, m. \tag{7.5}$$

Then the set $L := \{a\}$ is a lower bound set of $f_Z(x)$ for all $x \in S^*$.

Proof. Let $x \in S^*$ be arbitrarily chosen. We have to show that $a \in f_Z(x) - \mathbb{R}^m_+$. As $a_j \leq f_j(x+z^*)$ holds for all $j = 1, \dots, m$, $a \in \{f(x+z^*)\} - \mathbb{R}^m_+ \subseteq f_Z(x) - \mathbb{R}^m_+$ holds as well. □

In case of convex functions f_j, $j = 1, \dots, m$, the single objective optimization problems in (7.5) can be solved by any local solution method as an SQP method. Otherwise, convex underestimators, see Section 3.2, can be used. Let $\underline{f_j}$ be the convex underes-

timator of f_j on the box X^* for all $j = 1, \dots, m$. Define $\underline{f} := (\underline{f_1}, \dots, \underline{f_m})^T$. We can choose one or several points $z^* \in Z$ and determine the ideal point of the set

$$\underline{f}(S^* + z^*) := \{\underline{f}(x + z^*) \in \mathbb{R}^m \mid x \in S^*\}$$

for each chosen z^*. As each ideal point gives a lower bound set of the sets $f_Z(x)$ for all $x \in S^*$, also the set of all ideal points to the various points z^* is a lower bound set of $f_Z(x)$ for all $x \in S^*$:

Lemma 7.11 *Let X^* be a subbox of X, and let $\underline{f_j}$ be the convex underestimator of f_j on X^* for all $j = 1, \dots, m$ as defined in Definition 3.8. Let $Z^* = \{z^1, \dots, z^p\} \subseteq Z$, and for $k = 1, \dots, p$ determine the ideal point $\underline{a}^k$ of $\underline{f}(S^* + z^k)$, i. e., calculate*

$$\underline{a}^k := (\min_{x \in S^*} \underline{f_1}(x + z^k), \dots, \min_{x \in S^*} \underline{f_m}(x + z^k))^T.$$

Then it holds:

$$L := \{\underline{a}^1, \dots, \underline{a}^p\} \preccurlyeq_u f_Z(x) \text{ for all } x \in S^*,$$

i. e., L is a lower bound set for all sets $f_Z(x)$ with $x \in S^*$.

Proof. We have to show that for any $k \in \{1, \dots, p\}$ and for all $x^* \in S^*$ it holds that $a^k \in f_Z(x^*) - \mathbb{R}^m_+$. Thus, let $k \in \{1, \dots, p\}$ and $x^* \in S^*$ be arbitrarily chosen. As

$$\underline{a_j}^k = \min_{x \in S^*} \underline{f_j}(x + z^k) \leq \underline{f_j}(x^* + z^k) \leq f_j(x^* + z^k)$$

holds for all $j = 1, \dots, m$, if follows $\underline{a}^k \in \{f(x^* + z^k)\} - \mathbb{R}^m_+ \subseteq f_Z(x^*) - \mathbb{R}^m_+$. □

A visualization of this lemma can be found in Figure 7.4. The set $f_Z(S^*)$ is defined by $\{f_Z(x + z) \mid x \in S^*, z \in Z\}$ and includes all images of all possible outcomes $x^* + z$ with $x^* \in S^*$ and $z \in Z$. By fixing a point $z^k \in Z$ we only obtain a subset $\{f(x + z^k) \mid x \in S^*\} =: A^k$ of $f_Z(S^*)$ for every $k = 1, \dots, p$. As $A^k + \mathbb{R}^m_+$ is in general nonconvex, a convex underestimator and its ideal point $\underline{a}^k$ are computed. The lower bound set L consists of all ideal points a^k, $k = 1, \dots, p$.

For the algorithm, a large p, i. e., many elements $\{z^1, \dots, z^p\}$, improves the lower bound set. On the other hand, a large p implies that we have to solve a large number

of convex single objective optimization problems on each subbox to determine the ideal points $\underline{a}^k$, $k = 1, \ldots, p$. We explore this issue in Section 7.3.2.

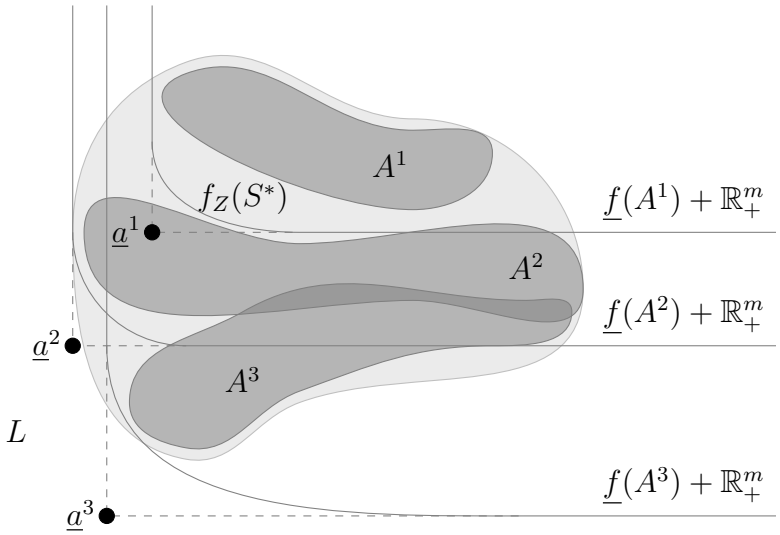


Figure 7.4. The set $f_Z(S^*)$, the sets $A^k = \{f(x + z^k) \mid x \in S^*\},\ k = 1, 2, 3$ and the lower bound set $L = \{\underline{a}^1, \underline{a}^2, \underline{a}^3\}$ according to Lemma 7.11.

7.2.4 Discarding Test

The main idea of our discarding test is the implication given in (7.1) together with the way in which we construct the sets L and U. We summarize our discarding test in the next theorem.

Theorem 7.12 *Let X^* be a subbox of X, $\tilde{x} \in S$ with $\tilde{x} \notin X^*$, and let the set U be defined as in Lemma 7.9 and the set L as in Lemma 7.11. If $U \preccurlyeq_u L$, the subbox X^* can be discarded, i. e., $X^* \cap S$ does not contain any decision robust strictly efficient solution of (P) w. r. t. Z.*

Proof. With U as in Lemma 7.9 and with L as in Lemma 7.11 we obtain

$$f_Z(\tilde{x}) \preccurlyeq_u U \preccurlyeq_u L \preccurlyeq_u f_Z(x) \text{ for all } x \in S^*,$$

i. e., no $x \in S^*$ can be a strictly optimal solution of the set optimization problem $\min_{x \in S} f_Z(x)$. Therefore, no $x \in S$ can be decision robust strictly efficient for (P) w. r. t. Z. □

With our choice for the construction of the sets L and U we aim at two goals. Firstly, we want to be able to easily calculate them. This is achieved since the arising convex single objective optimization problems can easily be solved, as any locally optimal solution is already a globally optimal solution. Secondly, we want the sets L and U to have a simple structure such that they can easily be compared w. r. t. the upper-type less order relation. If L and U are finite sets, this can be done with a pairwise comparison. In case of $m = 2$ and U is not finite, i. e., not just the anti-ideal point, then U is just a line segment and we can easily check whether $U \preccurlyeq_u L$ holds. For $m \geq 3$ and non-finite U such comparisons get already more complicated, as U is then a subset of a hyperplane. This comparison becomes even more difficult if several points $\bar{p}$ are used in $(P_{\tilde{x},\bar{p}})$ to construct improved outer approximations, as it was done in Chapters 4 and 6. In that case the concept of local upper bounds might be helpful as used and explained in detail in Section 4.1.2. As this is an implementation issue and does not add insights to the main idea of the discarding test above, we limit ourselves here to cases which can easily be implemented.

7.3 The Algorithm and Numerical Results

In this section, we derive the algorithm based on the proposed discarding test in Theorem 7.12. We also illustrate and discuss the algorithm on several test instances.

7.3.1 The B&B Algorithm

The base of our algorithm MOPDUBB is a B&B approach in which we partition the box X into subboxes, see also Algorithm 2. Then we try to discard subboxes which do not contain decision robust strictly efficient solutions.

Algorithm 9 MOPDUBB: Algorithm for multiobjective optimization problems with Decision Uncertainty

Input: $f \in \mathcal{C}^2(\mathbb{R}^n, \mathbb{R}^m),\ S \subseteq \mathbb{R}^n,\ X \in \mathbb{IR}^n$ with $S \subseteq X,\ Z \subseteq \mathbb{R}^n$, $\hat{Z} \in \mathbb{IR}^n$ with $Z \subseteq \hat{Z},\ Z^* = \{z^1, \ldots, z^p\}$

Output: $\mathcal{L}_{S,\text{final}}$

$\mathcal{L}_W \leftarrow \{X\}, \mathcal{L}_S \leftarrow \emptyset, \mathcal{L}_{S,\text{final}} \leftarrow \emptyset$
if $\text{mid}(X) \in S$ **then**
    $\tilde{x} \leftarrow \text{mid}(X)$
    Compute U with $f_Z(\tilde{x}) \preccurlyeq_u U$, see Lemmas 7.8 and 7.9
    Store U in $\mathcal{L}_U$
else $\mathcal{L}_U \leftarrow \emptyset$
while $\mathcal{L}_W \neq \emptyset$ **do**
    Select a box X^* from $\mathcal{L}_W$ and delete it from $\mathcal{L}_W$
    Bisect X^* perpendicularly to a direction of maximum width $\rightarrow X^1, X^2$
    for $l = 1, 2$ **do**
        Compute L with $L \preccurlyeq_u f_Z(x)$ for all $x \in X^l \cap S$, see Lemma 7.11, obtain an $\tilde{x} \in X^l \cap S$ or $X^l \cap S = \emptyset$
        if $X^l \cap S = \emptyset$ **then** Discard X^l
        else
            if there is an $U \in \mathcal{L}_U$ with $U \preccurlyeq_u L$ **then** Discard X^l
            else
                Compute U with $f_Z(\tilde{x}) \preccurlyeq_u U$, see Lemmas 7.8 and 7.9
                Store U in $\mathcal{L}_U$ and update $\mathcal{L}_U$
                if $\omega(X^l) < \delta$ **then** Store X^l with L in $\mathcal{L}_S$
                else Store X^l in $\mathcal{L}_W$
while $\mathcal{L}_S \neq \emptyset$ **do**
    Select a box X^* with its lower bound L from $\mathcal{L}_S$ and delete it from $\mathcal{L}_S$
    if there is a $U \in \mathcal{L}_U$ with $U \preccurlyeq_u L$ **then** Discard X^l
    else Store X^l in $\mathcal{L}_{S,\text{final}}$

We are working with several lists. The list $\mathcal{L}_W$ is as usual the working list, which collects those boxes which are still of interest. The upper bound sets for some feasible points $\tilde{x} \in S$ are collected in the list $\mathcal{L}_U$. The list $\mathcal{L}_S$ collects those boxes which deliver a first cover of the set of decision robust strictly efficient solutions. In the second `while`-loop from line 20 we check again if boxes from $\mathcal{L}_S$ can now be discarded, because the set of upper bound sets $\mathcal{L}_U$ changes during the main `while`-loop (lines 7 to 19). The final solution list is denoted by $\mathcal{L}_{S,\text{final}}$.

For obtaining lower and upper bound sets, single objective optimization problems are solved. In lines 4 and 16 the upper bound set U of a set $f_Z(x)$ is computed. If the set U consists of the anti-ideal point only, m optimization problems are solved. If the set U is determined by outer approximations, we need $m+1$ optimization problems. For the lower bound set in line 11, the algorithm solves $mp = m|Z^*|$ additional optimization problems.

For computing upper bound sets, at least one feasible point $\tilde{x} \in S^*$ of a currently considered box X^* is required. One possibility is to use the midpoints of the boxes, i. e., $\tilde{x} := \text{mid}(X^*)$ as far as $\tilde{x} \in S$. Another possibility to obtain feasible points is the following: in line 11, see also Lemma 7.11, for each $z \in Z^*$ and for each objective function f_j an optimization problem is solved with feasible set S^*. The minimal solutions can thus be used as feasible points for computing upper bound sets. In addition, we obtain the information whether S^* is empty.

To get only one upper bound set per considered box and to make numerical experiments comparable, we use the minimal solution of the first (underestimated) objective function without uncertainties, i. e., with $z^* = 0$.

After adding a new upper bound set to $\mathcal{L}_U$, an update procedure is applied in line 17: If there is a set $U' \in \mathcal{L}_U$ with $U' \preccurlyeq_u U$ for some $U \in \mathcal{L}_U$, all those dominated U are removed from $\mathcal{L}_U$. Moreover, U' is not stored in $\mathcal{L}_U$ if $U \preccurlyeq_u U'$ holds for one $U \in \mathcal{L}_U$. This reduces the amount of comparisons for checking the conditions in lines 14 and 22.

We stop MOPDUBB when all boxes are either discarded or put into the solution list, i. e., if the working list is empty. We put a box X^* to the solution list if its box width is small enough, i. e., for given $\delta > 0$ it holds: $\omega(X^*) < \delta$.

Theorem 7.13 *Let $\mathcal{L}_{S,\text{final}}$ be the output of* MOPDUBB*, and $\mathcal{L}_S$ be the solution list after line 19. Moreover, let X_E be the set of decision robust strictly efficient solutions of (P) w. r. t. Z. Then the following holds:*

(i) $X_E \subseteq \bigcup_{X \in \mathcal{L}_{S,\text{final}}} X \subseteq \bigcup_{X \in \mathcal{L}_S} X$

(ii) For every box $X^* \in \mathcal{L}_{S,\text{final}}$ it holds: $\omega(X^*) < \delta$

Moreover, the algorithm terminates after finitely many subdivisions in line 9.

Proof. Property (i) holds because of Theorem 7.12. In line 18 only boxes X^* with $\omega(X^*) < \delta$ are stored in the solution list $\mathcal{L}_S$. The final elimination in the second `while`-loop of the algorithm just discards some boxes from $\mathcal{L}_S$, but the box width of the boxes does not change. This proves (ii).

The boxes are bisected perpendicularly to a direction of maximum width. Boxes are either discarded or partitioned until the subboxes have a width smaller than δ. Hence, even if no box is discarded, all subboxes have a width smaller than δ after a finite number of subdivisions. □

We would like to mention that MOPDUBB does not guarantee to find anything like almost decision robust strictly efficient solutions or that with decreasing δ the covering of X_E gets arbitrarily close. For being able to show something like that, we would need lower and upper bound sets which get closer to the sets which are compared for smaller boxes. With Remark 3.13 it follows that for smaller boxes X^* the lower bounds for $f(S^* + z^*)$ for one $z^* \in Z^*$ are indeed tighter than for larger boxes. For an upper bound set U one has to calculate concave overestimators $\overline{f_j}$ for each f_j on the box $\{\tilde{x}\} + \hat{Z}$. However, the distance between $\overline{f_j}$ and f_j does not decrease, when the boxes X^* become smaller, as it only depends on the size of $\hat{Z}$ (and the non-concavity of f_j). The boxes in the set $\mathcal{L}_{S,\text{final}}$ might contain infeasible points. However, the distance from any point in any box in $\mathcal{L}_{S,\text{final}}$ to the feasible set is bounded from above by δ, because in every box in $\mathcal{L}_{S,\text{final}}$ there exists at least one feasible point. Otherwise, it would have been discarded.

7.3.2 Numerical Results

The proposed algorithm MOPDUBB has been implemented in MATLAB R2018a and uses the toolbox Intlab [Rum99] for interval arithmetic. All experiments have been done on a computer with Intel(R) Core(TM) i5-7400T CPU and 16 Gbytes RAM on operating system Windows 10 Enterprise.

As the notion of decision robust strictly efficient solutions has been introduced in multiobjective optimization recently, there are no test instances from the literature so far. Here, we introduce some test instances where it is possible to calculate the decision robust strictly efficient solution sets analytically. This allows to verify our results. We also discuss the impact of various parameters of the algorithm as, for instance, the choice and the number p of elements of the set Z^* which are used to calculate the lower bounds in Lemma 7.11.

For all instances we used $\delta = 0.05$. Additionally, we used $\delta = 0.01$ for the last two test instances to explore the influence of δ. In Table 7.1 the number of subdivisions in line 9 of MOPDUBB, the time, the number of boxes in the final solution list, and the total number of solved single objective optimization problems are given for each test instance. The last number is split into the number of solved problems to obtain the upper bounds U and the number of solved problems to get the lower bounds L. Also, visualizations of the partitions of the initial box X after the execution of the algorithm are presented. The light gray boxes are the discarded boxes. All boxes from the final solution list $\mathcal{L}_{S,\text{final}}$ are dark gray. In case of convex constraints, boxes can be discarded because of infeasibility. These boxes are white in the figures.

Moreover, we compare the impact of using upper bounds only based on the anti-ideal point, as in Lemma 7.8, and based on an improved outer approximations, as defined in Lemma 7.9.

Test instance T.29 This convex test function is inspired by [BK97]:

$$f(x) = \begin{pmatrix} x_1^2 + x_2^2 \\ (x_1 - 5)^2 + (x_2 - 5)^2 \end{pmatrix}$$

with $S = X = [-5, 5] \times [-5, 5]$ and $\hat{Z} = Z = [-0.3, 0.1] \times [-0.3, 0.1]$.

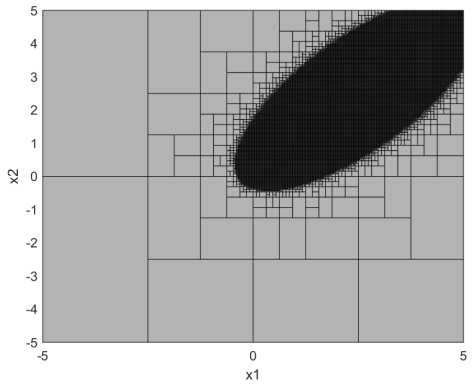
The set of decision robust strictly efficient solutions is a line segment:

$$\mathcal{L} = \{\lambda(0.1, 0.1)^T + (1-\lambda)(5,5)^T \mid \lambda \in [0,1]\}.$$

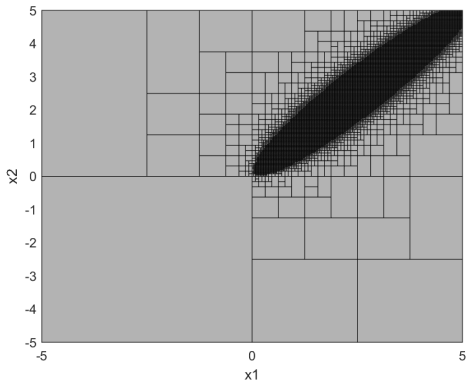
The sets Z^0 to Z^3 are chosen as

$$\begin{aligned}
Z^0 &= \{0\},\\
Z^1 &= \{(z_1, z_2) \in \mathbb{R}^2 \mid z_1, z_2 \in \{-0.3, 0.1\}\} \cup \{0\},\\
Z^2 &= \{(z_1, z_2) \in \mathbb{R}^2 \mid z_1, z_2 \in \{-0.3, -0.1, 0.1\}\} \cup \{0\} \text{ and}\\
Z^3 &= \{(z_1, z_2) \in \mathbb{R}^2 \mid z_1, z_2 \in \{-0.3, -0.2, -0.1, 0, 0.1\}\}.
\end{aligned}$$

The results of MOPDUBB are shown in Figure 7.5 and Table 7.1, first part.



a: $Z^* = Z^0$, upper bounds by anti-ideal points or improved outer approximation.



b: $Z^* = Z^3$, upper bounds by improved outer approximation.

Figure 7.5. Partition of the feasible set of Test instance T.29 after MOPDUBB.

By Table 7.1, first part, and Figure 7.5, we recognize that choosing a larger set Z^* and computing the lower bound set U by an improved outer approximation leads to better results. However, the improvement from the case $Z^* = Z^1$ to $Z^* = Z^2$ or $Z^* = Z^3$ is not as significant as the one from $Z^* = Z^0$ to $Z^* = Z^1$. The best choice for Z^* depends on how tight the covering of the set of decision robust strictly efficient solutions should be. Note that the point $\bar{x} = 0$ is not a decision robust strictly efficient solution while $\bar{x}$ is an efficient solution of the corresponding multiobjective optimization problem without uncertainties. That point is still included in the covering of the efficient set

in case of $Z^* = Z^0$. In all other cases, $\bar{x} = 0$ does not belong to a box of the solution lists. For $Z^* = Z^3$, this can be seen in Figure 7.5.

Test instance T.30 This test instance consists of a nonconvex objective function and a circular decision uncertainty set Z:

$$f(x) = \begin{pmatrix} x_1^2 - x_2^2 \\ x_1/x_2 \end{pmatrix}$$
$$\text{with } S = X = [-1, 2] \times [1, 2] \text{ and } Z = \{(z_1, z_2) \in \mathbb{R}^2 \mid z_1^2 + z_2^2 \leq 0.01\}.$$

A box $\hat{Z}$ which contains Z is $\hat{Z} = [-0.1, 0.1] \times [-0.1, 0.1]$.

The sets Z^0 and Z^1 are chosen as

$$Z^0 = \{0\},$$
$$Z^1 = \{(z_1, z_2) \in \mathbb{R}^2 \mid z_1, z_2 \in \{-0.1, -0.05, 0, 0.05, 0.1\}\} \cap Z.$$

For the results of MOPDUBB on Test instance T.30, see Figure 7.6 and Table 7.1, second part.

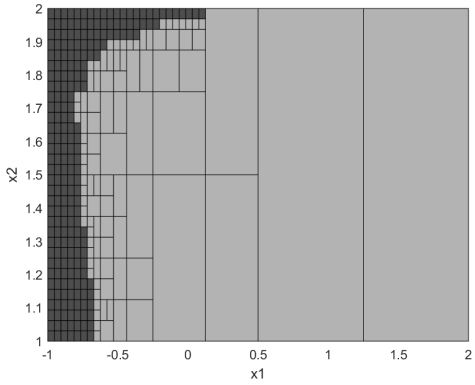
In Figure 7.6 it can be seen that choosing a smaller δ leads to a tightening of the covering of the decision robust strictly efficient solutions. However, all quantities increase because more boxes have to be bisected until the termination rule is fulfilled.

Test instance T.31

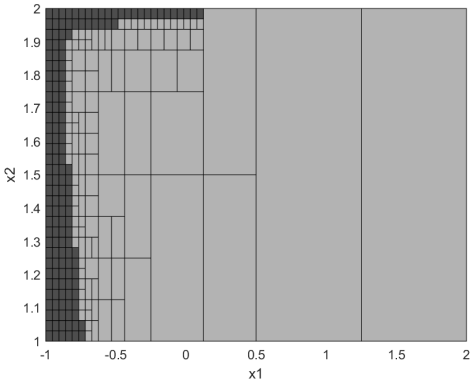
$$f(x) = \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix}$$
$$\text{with } S = \left\{ \left(x_1, x_2\right) \in \mathbb{R}^2 \middle| \begin{array}{ll} x_1^2 + x_2^2 & \leq 0.5 \\ x_1 - x_2 & \leq 0.5 \end{array} \right\} \cap [-1, 1] \times [-1, 1]$$
$$\text{and } \hat{Z} = Z = [-0.1, 0.3] \times [-0.1, 0.3]$$

The set of the decision robust strictly efficient solutions is

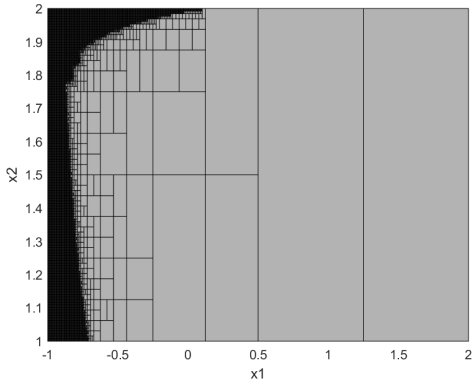
$$\mathcal{L} = \{(-0.1, -0.1)^T\}.$$



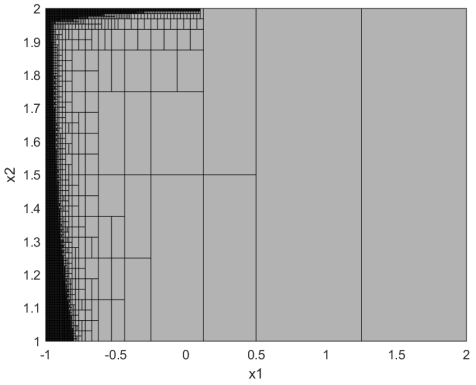
a: $Z^* = Z^1$, upper bounds by anti-ideal point, $\delta = 0.05$.



b: $Z^* = Z^1$, upper bounds by improved outer approximation, $\delta = 0.05$.



c: $Z^* = Z^1$, upper bounds by anti-ideal point, $\delta = 0.01$.



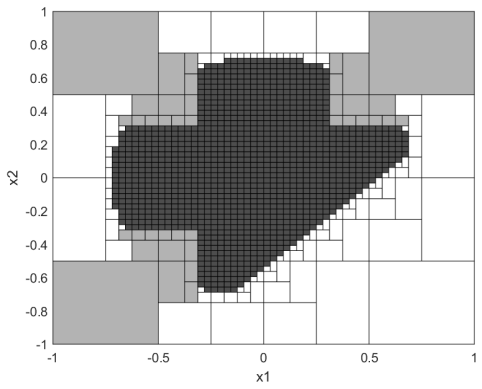
d: $Z^* = Z^1$, upper bounds by improved outer approximation, $\delta = 0.01$

Figure 7.6. Partition of the feasible set of Test instance T.30 after MOPDUBB.

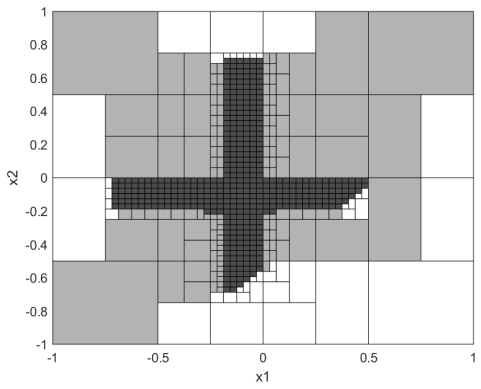
The sets Z^0 to Z^3 are chosen as

$$
\begin{aligned}
Z^0 &= \{0\},\\
Z^1 &= \{(z_1, z_2) \in \mathbb{R}^2 \mid z_1, z_2 \in \{-0.1, 0.3\}\} \cup \{0\},\\
Z^2 &= \{(z_1, z_2) \in \mathbb{R}^2 \mid z_1, z_2 \in \{0, 0.1, 0.2\}\} \text{ and}\\
Z^3 &= \{(z_1, z_2) \in \mathbb{R}^2 \mid z_1, z_2 \in \{-0.1, 0, 0.1, 0.2, 0.3\}\}.
\end{aligned}
$$

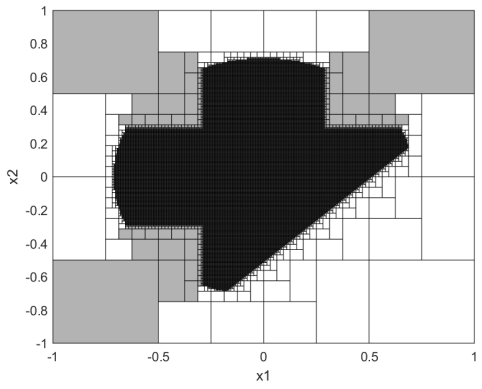
Note that Z^1 consists of the vertices of Z (and 0) while Z^2 only contains some interior points of Z. The results of MOPDUBB are shown in Figure 7.7 and Table 7.1, third part.



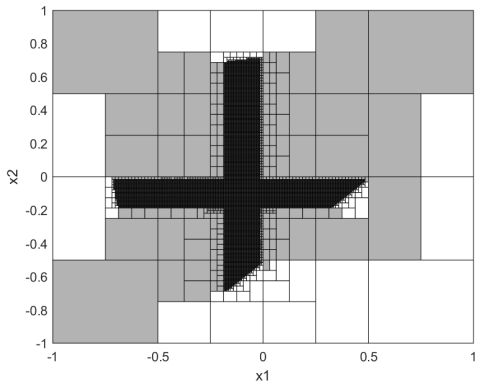
a: $Z^* = Z^0$, upper bounds by anti-ideal point or by improved outer approximation, $\delta = 0.05$.



b: $Z^* = Z^1$ or $Z^* = Z^3$, upper bounds by anti-ideal point or by improved outer approximation, $\delta = 0.05$.



c: $Z^* = Z^0$, upper bounds by anti-ideal point or by improved outer approximation,, $\delta = 0.01$.



d: $Z^* = Z^1$ or $Z^* = Z^3$, upper bounds by anti-ideal point or by improved outer approximation, $\delta = 0.01$.

Figure 7.7. Partition of the feasible set of Test instance T.31 after MOPDUBB.

Table 7.1. Results for Test instances T.29 to T.31, for different values of δ

			U by anti-ideal points				U by improved outer approximation			
δ	Z^*	$\lvert Z^*\rvert$	# subdiv.	$t\,[s]$	$\lvert\mathcal{L}_{S,\text{final}}\rvert$	# OP $(U+L)$	# subdiv.	$t\,[s]$	$\lvert\mathcal{L}_{S,\text{final}}\rvert$	# OP $(U+L)$
0.05	Z^0	1	12213	**2.4e+03**	11354	47146 + 48852	12213	3.1e+03	11354	70719 + 48852
	Z^1	5	9594	3.0e+03	8776	36746 + 191860	7615	2.8e+03	6686	42945 +152300
	Z^2	10	9593	4.5e+03	8776	36744 + 383680	6714	3.5e+03	5570	36936 + 268560
	Z^3	25	9584	8.8e+03	8776	36726 + 958300	**6175**	6.0e+03	**4704**	32739 + 617500
0.05	Z^0	1	454	**100**	359	1630 + 1816	454	154	359	2445 + 1816
	Z^1	13	317	194	224	1086 + 16484	**264**	192	**154**	1263 + 13728
0.01	Z^0	1	10242	**2.3e+03**	9583	39768 + 40968	10242	3.8e+03	9583	59652 + 40968
	Z^1	13	5833	3.6e+03	5142	22092 + 303316	**3215**	2.4e+03	**2319**	16872 + 167180
0.05	Z^0	1	1309	206	1171	4804 + 5236	1309	265	1171	7206 + 5236
	Z^1	5	**623**	**191**	**453**	2090 + 12124	**623**	218	**453**	3135 + 12124
	Z^2	9	1011	472	857	3614 + 35212	1011	519	857	5421 + 35212
	Z^3	25	**623**	678	**453**	2090 + 60284	**623**	706	**453**	3135 + 60284
0.01	Z^0	1	17834	3.0e+03	17096	69208 + 71336	17834	3.9e+03	17096	103812 + 71336
	Z^1	5	**6873**	**2.0e+03**	**6354**	26204 + 136372	**6873**	2.4e+03	**6354**	39306 + 136372
	Z^2	9	13108	5.7e+03	12404	50536 + 467456	13108	6.5e+03	12404	75804 + 467456
	Z^3	25	**6873**	6.6e+03	**6354**	26204 + 680772	**6873**	7.1e+03	**6354**	39306 + 680772

The results for this test instance, see Table 7.1, third part, show that the upper bound sets which are obtained from the improved outer approximations do not lead to any improvements. The number of subdivisions and boxes in the solution list $\mathcal{L}_{S,\text{final}}$ are the same. We suppose that this is caused by the convex objective functions, which are symmetric and do not depend on the same variables. The computational time is even higher if the upper bound sets are obtained by outer approximations, because more optimization problems have to be solved.

It can be seen that choosing a set Z^* with multiple points improves the results. If the vertices of Z are included in Z^*, i. e., in case of Z^1 and Z^3, we obtain the smallest amount of subdivisions and boxes in the solution list. The reason for this is the symmetry of the convex objective functions again. Therefore, for this example, the best choice for Z^* is Z^1, and it is sufficient to use the anti-ideal point of the concave overestimators only. Another thing is that the covering of the decision robust strictly efficient solution $(-0.1, -0.1)^T$ is cross shaped and lies more symmetrically around the real decision robust solution set in case of $Z^* \neq \{0\}$.

The influence of δ to the covering of the decision robust strictly efficient points is not that noticeable. This clarifies that with decreasing δ we cannot expect to obtain an arbitrary tight covering of the set X_E, as remarked below Theorem 7.13.

In all additional numerical experiments similar results have been observed. To summarize, choosing a set Z^* with more than one element is a better choice than $Z^* = \{0\}$. On the other hand, the cardinality of Z^* does not have to be very large. For simple (i. e., convex, concave, symmetric, independent,...) objective functions the set with some characteristic points at the boundary of Z, e. g. the vertices of $\hat{Z}$ and 0, leads to overall best results. In general, using outer approximations to obtain upper bound sets improves the results.

7.4 Conclusions

In this chapter, we proposed a B&B algorithm for multiobjective optimization problems with decision uncertainty, which was also published in [ENR19]. In case of nonlinear objective functions we need to work with convex underestimators or concave

overestimators, or even with both, to make the subproblems numerically tractable. The computational experiments showed that the algorithm is indeed able to discard areas which do not contain any decision robust strictly efficient solution. Nevertheless, the remaining parts of the feasible set can still be large and then other (local) algorithms could be applied afterwards for more exact solutions. Moreover, the results can be improved, i. e., the covering can be tightened by improving the outer approximation used in Lemma 7.9 with techniques from Section 3.3.

We have assumed that the set Z is convex. An adaption also to nonconvex sets is possible. One can replace the set Z within the optimization problems in Section 7.2.2 by the convex hull of Z, or by a box which contains Z. If Z is a finite (and small) set, the problems can also be solved directly by enumeration.

In addition to the proposed procedure, it might be possible to improve the bounding procedure by using local upper bounds (see Section 3.4) for the comparison step of a lower bound set L and the non-singleton upper bounds in $\mathcal{L}_U$. Until now, this comparison is only implemented for biobjective problems and it uses geometrical properties. For higher dimensions, "local-upper bound like" points w. r. t. the lower bound set L should be used to check whether $U \preccurlyeq_u L$ holds for one $U \in \mathcal{L}_U$. For that, a similar problem to $(P_{\bar{p},\,M})$ can be solved for some of the "local-upper bound like" points. Then we obtain new hyperplanes for the upper bound U if a "local-upper bound like" point can be separated from $f_Z(\tilde{x})$. Thus, the upper bound set improves adaptively if necessary.

The techniques proposed for MOPDUBB have been developed for a set optimization problem with a very specific structure and with the upper-type less order relation. It is also of interest, and we believe it is possible, to adapt the methods for other set order relations and more general set-valued optimization problems. For example, the new methods can easily be adapted for the lower-type less order relation $\preccurlyeq_l$ ($A \preccurlyeq_l B \Leftrightarrow B \subseteq A + \mathbb{R}^m_+$). This is also related to an optimistic approach to uncertain multiobjective optimization, see Section 3.2 in [Ide+14].

To handle other and more general set-valued optimization problems, we have to ensure a certain structure of the set-valued objective function, i. e., convex underestimators or concave overestimators should be computable. This is the case for a set-valued map given by $f_Z(x) := \{f(x,z) \mid z \in Z\}$, where $f\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m$. A special case

of this is, for instance, $f_Z(x) := \{f(x + g(z)) \mid z \in Z\}$, where $g \colon \mathbb{R}^n \to \mathbb{R}^n$ is a twice continuously differentiable function.

A Appendix

Test Instances of Section 5.3

In this appendix, all test instances of Section 5.3 are listed which were not already given in a previous section. All optimization problems aim to minimize the objective $f = (f_1, \dots, f_m)^T$ on the feasible set described by a box X and convex constraints $g = (g_1, \dots, g_p)^T$.

Test instance T.4 [DTLZ1] The dimensions of the preimage space $n \in \mathbb{N}$ and the image space $m \in \mathbb{N}$ can be arbitrarily chosen. It holds $m \leq n$, $k := n - m + 1$.

$$\begin{aligned} & f_1(x) && = \tfrac{1}{2}(1 + h(x)) \textstyle\prod_{i=1}^{m-1} x_i \\ & f_j(x) && = \tfrac{1}{2}(1 + h(x)) \textstyle\prod_{i=1}^{m-j} x_i \cdot (1 - x_{m-j+1}) \text{ for } j = 2, \dots, m \\ \text{with} \quad & h(x) && = 100\,(k + \textstyle\sum_{i=m}^{n} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))) \\ & X && = [0,1]^n \end{aligned}$$

Test instance T.5 [DTLZ2] The dimensions of the preimage space $n \in \mathbb{N}$ and the image space $m \in \mathbb{N}$ can be arbitrarily chosen. It holds $m \leq n$.

$$\begin{aligned} & f_1(x) && = (1 + h(x)) \textstyle\prod_{i=1}^{m-1} \cos(0.5\pi x_i) \\ & f_j(x) && = (1 + h(x)) \textstyle\prod_{i=1}^{m-j} \cos(0.5\pi x_i) \cdot \sin(0.5\pi x_{m-j+1}) \\ & && \qquad \text{for } j = 2, \dots, m \\ \text{with} \quad & h(x) && = \textstyle\sum_{i=m}^{n} (x_i - 0.5)^2 \\ & X && = [0,1]^n \end{aligned}$$

Test instance T.6 [DTLZ3] The same like Test instance T.5 except $h(x)$ is chosen from Test instance T.4. The dimensions of the preimage space $n \in \mathbb{N}$ and the image space $m \in \mathbb{N}$ can be arbitrarily chosen. It holds $m \leq n$, $k := n - m + 1$.

$$
\begin{aligned}
f_1(x) &= (1 + h(x)) \prod_{i=1}^{m-1} \cos(0.5\pi x_i) \\
f_j(x) &= (1 + h(x)) \prod_{i=1}^{m-j} \cos(0.5\pi x_i) \cdot \sin(0.5\pi x_{m-j+1}) \\
&\qquad\qquad \text{for } j = 2, \ldots, m \\
\text{with} \quad h(x) &= 100 \left(k + \sum_{i=m}^{n} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))\right) \\
X &= [0, 1]^n
\end{aligned}
$$

Test instance T.7 [DTLZ4] The same like Test instance T.5 except all x_i are replaced by x_i^α. The dimensions of the preimage space $n \in \mathbb{N}$ and the image space $m \in \mathbb{N}$ can be arbitrarily chosen. It holds $m \leq n$.

$$
\begin{aligned}
f_1(x) &= (1 + h(x)) \prod_{i=1}^{m-1} \cos(0.5\pi x_i^\alpha) \\
f_j(x) &= (1 + h(x)) \prod_{i=1}^{m-j} \cos(0.5\pi x_i^\alpha) \cdot \sin(0.5\pi x_{m-j+1}^\alpha) \\
&\qquad\qquad \text{for } j = 2, \ldots, m \\
\text{with} \quad h(x) &= \sum_{i=m}^{n} (x_i^\alpha - 0.5)^2 \\
\alpha &= 100 \\
X &= [0, 1]^n
\end{aligned}
$$

Test instance T.8 [DTLZ5] The same like Test instance T.5 except all $0.5\pi x_i, i = 1, \ldots, m-1$ are replaced by θ_i. The dimensions of the preimage space $n \in \mathbb{N}$ and the image space $m \in \mathbb{N}$ can be arbitrarily chosen. It holds $m \leq n$.

$$
\begin{aligned}
f_1(x) &= (1 + h(x)) \cos(0.5\pi x_1) \textstyle\prod_{i=2}^{m-1} \cos(\theta_i) \\
f_j(x) &= (1 + h(x)) \cos(0.5\pi x_1) \textstyle\prod_{i=1}^{m-j} \cos(\theta_i) \cdot \sin(\theta_{m-j+1}) \\
&\qquad\qquad \text{for } j = 2, \ldots, m-1 \\
f_m(x) &= (1 + h(x)) \sin(0.5\pi x_1) \\
\text{with} \quad h(x) &= \textstyle\sum_{i=m}^{n} (x_i - 0.5)^2 \\
\theta_i &= \tfrac{1}{4}\pi \tfrac{1+2h(x)x_i}{1+h(x)} \qquad\qquad \text{for } i = 2, \ldots, m-1 \\
X &= [0, 1]^n
\end{aligned}
$$

Test instance T.9 [DTLZ6] The same like Test instance T.8 except $h(x)$ is $\sum_{i=m}^{n} x_i^{0.1}$. The dimensions of the preimage space $n \in \mathbb{N}$ and the image space $m \in \mathbb{N}$ can be arbitrarily chosen. It holds $m \leq n$.

$$
\begin{aligned}
f_1(x) &= (1 + h(x)) \cos(0.5\pi x_1) \textstyle\prod_{i=2}^{m-1} \cos(\theta_i) \\
f_j(x) &= (1 + h(x)) \cos(0.5\pi x_1) \textstyle\prod_{i=1}^{m-j} \cos(\theta_i) \cdot \sin(\theta_{m-j+1}) \\
&\qquad\qquad \text{for } j = 2, \ldots, m-1 \\
f_m(x) &= (1 + h(x)) \sin(0.5\pi x_1) \\
\text{with} \quad h(x) &= \textstyle\sum_{i=m}^{n} x_i^{0.1} \\
\theta_i &= \tfrac{1}{4}\pi \tfrac{1+2h(x)x_i}{1+h(x)} \qquad\qquad \text{for } i = 2, \ldots, m-1 \\
X &= [0, 1]^n
\end{aligned}
$$

Test instance T.10 [DTLZ7] The dimensions of the preimage space $n \in \mathbb{N}$ and the image space $m \in \mathbb{N}$ can be arbitrarily chosen. It holds $m \leq n$, $k := n - m + 1$.

$$\begin{aligned} f_j(x) &= x_j \qquad \text{for } j = 1, \ldots, m-1 \\ f_m(x) &= (1 + h_1(x))h_2(x, h_1) \\ \text{with} \qquad h_1(x) &= 1 + \tfrac{9}{k} \textstyle\sum_{i=m}^{n} x_i \\ h_2(x, h_1) &= m - \textstyle\sum_{i=1}^{m-1} \frac{x_i}{1+h_1(x)}(1 + \sin(3\pi x_i)) \\ X &= [0,1]^n \end{aligned}$$

Test instance T.11 [ZDT1] The dimension of the preimage space $n \in \mathbb{N}$ can be arbitrarily chosen.

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= h_1(x)h_2(x, h_1) \\ \text{with} \qquad h_1(x) &= 1 + \tfrac{9}{n-1} \textstyle\sum_{i=2}^{n} x_i \\ h_2(x, h_1) &= 1 - \sqrt{\frac{x_1}{h_1(x)}} \\ X &= [0,1]^n \end{aligned}$$

Test instance T.12 [ZDT2] The same like Test instance T.11 except $h_2(x, h_1)$ is $1 - \left(\frac{x_1}{h_1(x)}\right)^2$. The dimension of the preimage space $n \in \mathbb{N}$ can be arbitrarily chosen.

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= h_1(x)h_2(x, h_1) \\ \text{with} \qquad h_1(x) &= 1 + \tfrac{9}{n-1} \textstyle\sum_{i=2}^{n} x_i \\ h_2(x, h_1) &= 1 - \left(\frac{x_1}{h_1(x)}\right)^2 \\ X &= [0,1]^n \end{aligned}$$

Test instance T.13 [ZDT4] The dimension of the preimage space $n \in \mathbb{N}$ can be arbitrarily chosen.

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= h_1(x)h_2(x, h_1) \\ \text{with} \quad h_1(x) &= 1 + 10(n-1)\textstyle\sum_{i=2}^{n} + (x_i^2 - 10\cos(4\pi x_i)) \\ h_2(x, h_1) &= 1 - \sqrt{\tfrac{x_1}{h_1(x)}} \\ X &= [0,1] \times [-5,5]^{n-1} \end{aligned}$$

Test instance T.14 [Deb41Con] The constraint given by g is added by the author.

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= \tfrac{1}{x_1}\left(2 - \exp\left(-\left(\tfrac{x_2-0.2}{0.004}\right)^2\right) - 0.8\exp\left(-\left(\tfrac{x_2-0.6}{0.4}\right)^2\right)\right) \\ \text{with} \quad g(x) &= (x_1 - 0.5)^2 + (x_2 - 0.3)^2 - 0.09 \\ X &= [0.1, 1] \times [0, 1] \end{aligned}$$

Test instance T.15 [Deb513]

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= h_1(x)h_2(x, h_1) \\ \text{with} \quad h_1(x) &= 1 + 10x_2 \\ h_2(x, h_1) &= 1 - \left(\tfrac{x_1}{h_1(x)}\right)^{\alpha} - \tfrac{x_1}{h_1(x)}\sin(2\pi q x_1) \\ q &= 4 \\ \alpha &= 2 \\ X &= [0.1, 1] \times [0, 1] \end{aligned}$$

Test instance T.16 [Deb521a] Originally, this instance has a feasible set describes by $[0,1]^2$. Then the second objective function is not differentiable in $x_2 = 0$ and MOPBB could not be applied. For that reason, the feasible region was changed by the author. This affects also the efficient and nondominated set.

$$\begin{aligned} & f_1(x) && = x_1 \\ & f_2(x) && = h_1(x) h_2(x, h_1) \\ \text{with} \quad & h_1(x) && = 1 + x_2^{\gamma} \\ & h_2(x, h_1) && = 1 - \left(\frac{x_1}{h_1(x)}\right)^2 \\ & \gamma && = 0.25 \\ & X && = [0,1] \times [0.001, 1] \end{aligned}$$

Test instance T.17 [Deb521b] In contrast to [Deb521a] the second objective of this instance is differentiable in $x_2 = 0$.

$$\begin{aligned} & f_1(x) && = x_1 \\ & f_2(x) && = h_1(x) h_2(x, h_1) \\ \text{with} \quad & h_1(x) && = 1 + x_2^{\gamma} \\ & h_2(x, h_1) && = 1 - \left(\frac{x_1}{h_1(x)}\right)^2 \\ & \gamma && = 1 \\ & X && = [0,1]^2 \end{aligned}$$

Test instance T.18 [Viennet] This instance is the unconstrained version of Test instance T.3

$$\begin{aligned} & f_1(x) && = 0.5(x_1^2 + x_2^2)^2 + \sin(x_1^1 + x_2^2) \\ & f_2(x) && = \tfrac{(3x_1 - 2x_2 + 4)^2}{8} + \tfrac{(x_1 - x_2 + 1)^2}{27} + 15 \\ & f_3(x) && = \tfrac{1}{x_1^2 - x_2^2 - 1} - 1.1 \exp(-x_1^2 - x_2^2) \\ \text{with} \quad & X && = [-3, 3]^2 \end{aligned}$$

Test instance T.19 [SRN]

$$\begin{aligned} & f_1(x) &&= (x_1-2)^2+(x_2-1)^2+2 \\ & f_2(x) &&= 9x_1-(x_2-1)^2 \\ \text{with} \quad & g_1(x) &&= x_1^2+x_2^2-255 \\ & g_2(x) &&= x_1-3x_2+10 \\ & X &&= [-20,20]^2 \end{aligned}$$

Test instance T.20 [Pol]

$$\begin{aligned} & f_1(x) &&= 1+(A1-B1(x))^2+(A2-B2(x))^2 \\ & f_2(x) &&= (x_1+3)^2+(x_2)+1)^2 \\ \text{with} \quad & A1 &&= 0.5\sin(1)-2\cos(1)+\sin(2)-1.5\cos(2) \\ & A2 &&= 1.5\sin(1)-\cos(1)+2\sin(2)-0.5\cos(2) \\ & B1(x) &&= 0.5\sin(x_1)-2\cos(x_1)+\sin(x_2)-1.5\cos(x_2) \\ & B2(x) &&= 1.5\sin(x_1)-\cos(x_1)+2\sin(x_2)-0.5\cos(x_2) \\ & X &&= [-20,20]^2 \end{aligned}$$

Test instance T.21 [Far1]

$$\begin{aligned} f_1(x) &= -2\exp(15(-(x_1-0.1)^2-x_2^2))-\exp(20(-(x_1-0.6)^2-(x_2-0.6)^2)) \\ &\quad +\exp(20(-(x_1+0.6)^2-(x_2-0.6)^2)) \\ &\quad +\exp(20(-(x_1-0.6)^2-(x_2+0.6)^2)) \\ &\quad +\exp(20(-(x_1+0.6)^2-(x_2+0.6)^2)); \\ f_2(x) &= 2\exp(20(-x_1^2-x_2^2))+\exp(20(-(x_1-0.4)^2-(x_2-0.6)^2)) \\ &\quad -\exp(20(-(x_1+0.5)^2-(x_2-0.7)^2)) \\ &\quad -\exp(20(-(x_1-0.5)^2-(x_2+0.7)^2)) \\ &\quad +\exp(20(-(x_1+0.4)^2-(x_2+0.8)^2)) \\ \text{with } X &= [-1,1]^2 \end{aligned}$$

Test instance T.22 [KW2]

$$
\begin{aligned}
f_1(x) &= -(3(1-x_1)^2 \exp(-x_1^2-(x_2+1)^2) \\
&\quad -10(x_1/5 - x_1^3 - x_2^5)\exp(-x_1^2-x_2^2) \\
&\quad -3\exp(-(x_1+2)^2 - x_2^2) + 0.5(2x_1+x_2)) \\
f_2(x) &= -(3(1+x_2)^2 \exp(-x_2^2-(1-x_1)^2) \\
&\quad -10(-x_2/5 + x_2^3 + x_1^5)\exp(-x_2^2-x_1^2) \\
&\quad -3\exp(-(2-x_2)^2 - x_1^2)) \\
\text{with} \qquad X &= [-3,3]^2
\end{aligned}
$$

List of Nomenclature

anti-ideal point Let $S \subseteq \mathbb{R}^n$ and $f\colon \mathbb{R}^n \to \mathbb{R}^m$. The anti-ideal point $a = (a_1, \ldots, a_m)^T$ of f on S is defined by: $a_j = \max_{x \in S} f_j(x)$ for all $j = 1, \ldots, m$.

box A set $X \subseteq \mathbb{R}^n$ is called n-dimensional box if there are two vectors $\underline{x}, \overline{x} \in \mathbb{R}^n$ with $\underline{x} \leq \overline{x}$ such that $X = [\underline{x}, \overline{x}] := \{x \in \mathbb{R}^n \mid \underline{x} \leq x \leq \overline{x}\}$.

cone A set $K \subseteq \mathbb{R}^m$ is a cone if for all $k \in K$ and $\lambda > 0$ holds that $\lambda k \in K$. A convex cone K is *solid* if it has a non-empty interior and *pointed* if $K \cap (-K) = \{0\}$ holds. A *polyhedral* cone K can be given by a matrix $Z \in \mathbb{R}^{m \times m'}$ such that $K = \{y \in \mathbb{R}^m \mid Z^T y \geq 0\}$.

convex underestimator Let a function $h\colon \mathbb{R}^n \to \mathbb{R}$ on a box $X = [\underline{x}, \overline{x}] \in \mathbb{IR}^n$ be given. A convex underestimator for h on X is a convex function $\widehat{h}\colon X \to \mathbb{R}$ with $\widehat{h}(x) \leq h(x)$ for all $x \in X$.

elementary function unary function like $\sqrt{\cdot}$, sin, cos, arctan, ln, exp, $|\cdot|$

external stability [SNT85] For a given multiobjective optimization problem with objective function $f\colon \mathbb{R}^n \to \mathbb{R}^m$, feasible set M and an ordering cone $K \subseteq \mathbb{R}^m$, a set $Y \subseteq f(M)$ is said to be externally stable if for each $y \in f(M) \setminus Y$ there exists some $\hat{y} \in Y$ such that $y \in \hat{y} + K$.
Let K be a pointed closed convex cone, $f(M)$ be a nonempty K-compact set (for all $y \in f(M)$, the set $(\{y\} - \mathrm{cl}(K)) \cap f(M)$ is compact) and X_E be the efficient set. Then $f(X_E)$ is externally stable, i.e., $f(M) \subseteq f(X_E) + K$.

factorable function A function $h\colon \mathbb{R}^n \to \mathbb{R}$ is factorable if it can be formulated as a concatenation of basic arithmetic operations $(+, -, \cdot, /)$ and elementary functions

feasible For a given optimization problem $\min\{f(x) \mid x \in S\}$ a feasible point is a point which belongs to S. The set S is the feasible set.

Fritz-John condition [Mie12] A point $(x^*, \lambda^*, \mu^*) \in \mathbb{R}^{n+m+p}$ is a Fritz-John point for the multiobjective optimization problem (MOP), if the following (Fritz-John) conditions are satisfied:

(i) $\sum_{i=1}^{n} \lambda^* \nabla f_i(x^*) + \sum_{k=1}^{p} \mu_k^* g_k(x^*) = 0$,
(ii) $\sum_{k=1}^{p} \mu_k^* g_k(x^*) = 0$,
(iii) $g(x^*) \leq 0$,
(iv) $\lambda^*, \mu^* \geq 0$ and $(\lambda^*, \mu^*) \neq 0$.

hyperplane An set $H \subseteq \mathbb{R}^m$ is a hyperplane if there are $\lambda, \bar{y} \in \mathbb{R}^m$ such that $H = \{y \in \mathbb{R}^m \mid \lambda^T y = \lambda^T \bar{y}\}$. The vector λ is the *normal vector* of H and $\bar{y}$ is a *support vector* of H.

ideal point Let $S \subseteq \mathbb{R}^n$ and $f \colon \mathbb{R}^n \to \mathbb{R}^m$. The ideal point $a = (a_1, \ldots, a_m)^T$ of f on S is defined by: $a_j = \min_{x \in S} f_j(x)$ for all $j = 1, \ldots, m$.

Karush-Kuhn-Tucker condition [Ehr05] A point $(x^*, \mu^*) \in \mathbb{R}^{n+p}$ is a Karush-Kuhn-Tucker point for the single objective optimization problem $\min h(x)$ s. t. $g(x) \leq 0$, if the following (Karush-Kuhn-Tucker) conditions are satisfied:

(i) $\nabla h(x^*) + \sum_{k=1}^{p} \mu_k^* g_k(x^*) = 0$,
(ii) $\sum_{k=1}^{p} \mu_k^* g_k(x^*) = 0$,
(iii) $g(x^*) \leq 0$,
(iv) $\mu^* \geq 0$.

[Mie12] A point $(x^*, \lambda^*, \mu^*) \in \mathbb{R}^{n+m+p}$ is a Karush-Kuhn-Tucker point for the multiobjective optimization problem (MOP), if the following (Karush-Kuhn-Tucker) conditions are satisfied:

(i) $\sum_{i=1}^{n} \lambda^* \nabla f_i(x^*) + \sum_{k=1}^{p} \mu_k^* g_k(x^*) = 0$,
(ii) $\sum_{k=1}^{p} \mu_k^* g_k(x^*) = 0$,
(iii) $g(x^*) \leq 0$,
(iv) $\lambda^*, \mu^* \geq 0$ and $(\lambda^*, \mu^*) \neq 0$,
(v) $\lambda^* \neq 0$.

mean value theorem Let $h\colon \mathbb{R}^n \to \mathbb{R}$ be differentiable on the open set $S \subseteq \mathbb{R}^n$. Moreover, let $x^1, x^2 \in S$ with $\lambda x^1 + (1-\lambda)x^2 \in S$ for all $\lambda \in [0,1]$. Then exists an $\lambda \in (0,1)$ with $\xi := \lambda x^1 + (1-\lambda)x^2 \in S$ such that

$$f(x^2) - f(x^1) = \nabla f(\xi)^T (x^2 - x^1).$$

ordering cone An ordering cone is a convex cone $K \neq \{0\}$ which defines a partial order on $\mathbb{R}^m$ by: $y^1 \leq_K y^2 \Leftrightarrow y^2 - y^1 \in K$. The *natural ordering cone* in $\mathbb{R}^m$ is $\mathbb{R}^m_+$.

polyhedral convex set [Roc70] A polyhedral convex set in $\mathbb{R}^m$ is a set which can be expressed as the intersection of some finite collection of closed half-spaces.

real linear space [Jah11] A real linear space S is equipped with an addition operation $+\colon S \times S \to S$ and a scalar multiplication $\cdot\colon \mathbb{R} \times S \to S$ satisfying the following axioms:

(i) $(x+y)+z = x+(y+z)$ for all $x, y, z \in S$
(ii) $x + y = y + x$ for all $x, y \in S$
(iii) there is an element $0_S \in S$ with $x + 0_S = x$ for all $x \in S$
(iv) for every $x \in S$ there is a $y \in S$ with $x + y = 0_S$
(v) $\lambda(x+y) = \lambda x + \lambda y$ for all $x, y \in S$ and $\lambda \in \mathbb{R}$
(vi) $(\lambda + \mu)x = \lambda x + \mu x$ for all $x \in S$ and $\lambda, \mu \in \mathbb{R}$
(vii) $\lambda(\mu x) = (\lambda\mu)x$ for all $x \in S$ and $\lambda, \mu \in \mathbb{R}$
(viii) $1x = x$ for all $x \in S$

regularity conditions/constraint qualifications (CQ) conditions on a set which is usually a feasible set, given by equality and inequality constraints, of an optimization problem. Usual constraint qualifications, see [Kla09] for more details:

- Abadie-CQ
- Mangasarian-Fromovitz-CQ
- Linear Independence CQ
- Slater's condition for convex optimizations problems

subbox A box $\tilde{X}$ is a subbox of a box X if $\tilde{X} \subseteq X$ holds. In particular boxes which are obtained by iterative subdivisions of X are subboxes of X.

supporting hyperplane A hyperplane $H^{\lambda,\bar{y}} \subseteq \mathbb{R}^m$ with normal vector λ and support vector $\bar{y}$ is a supporting hyperplane of a set $S \in \mathbb{R}^m$ if it holds that $\lambda^T s \geq \lambda^T \bar{y}$ for all $s \in S$ and $\bar{y} \in \operatorname{bd} S$.

List of Abbreviations

B&B	branch-and-bound
CPU	central processing unit
GPU	graphics processing unit
LFV	Lorentz force velocimetry
MOMICP	multiobjective mixed integer convex optimization problem
MOMIX	branch-and-bound based algorithm for multiobjective optimization mixed integer problems
MOPBB	branch-and-bound based algorithm for multiobjective optimization problems
MOPDUBB	branch-and-bound based algorithm for multiobjective optimization problems with decision uncertainty
MultiGLODS	Multiobjective Global and Local Optimization using Direct Search
NSGA	Nondominated Sorting Genetic Algorithm
SQP	sequential quadratic programming
s. t.	subject to
w. r. t.	with respect to

List of Symbols

$\mathbb{N}$	set of natural numbers without 0
$\mathbb{Z}$	set of integer numbers
$\mathbb{R}$	set of real numbers
$\mathbb{R}^m$	set of vectors of m real numbers, $y = (x_1, \dots, y_m)^T \in \mathbb{R}^m \Leftrightarrow y_j \in \mathbb{R}$ for all $j = 1, \dots, m$
$\mathbb{R}^m_+$	$\mathbb{R}^m_+ := \{y \in \mathbb{R}^m \mid y_j \geq 0 \text{ for all } j = 1, \dots, m\}$
$y^1 \neq y^2$	$\Leftrightarrow y^1_j \neq y^2_j$ for at least one $j = 1, \dots, m$
$y^1 \leq y^2$	$\Leftrightarrow y^2 - y^1 \in \mathbb{R}^m_+ \Leftrightarrow y^1_j \geq y^2_j$ for all $j = 1, \dots, m$
$y^1 < y^2$	$\Leftrightarrow y^2 - y^1 \in \text{int}(\mathbb{R}^m_+) \Leftrightarrow y^1_j < y^2_j$ for all $j = 1, \dots, m$
$y^1 \lneq y^2$	$\Leftrightarrow y^1 \leq y^2$ and $y^1 \neq y^2$
$y^1 \not\leq y^2$	$\Leftrightarrow y^1_j > y^2_j$ for at least one $j = 1, \dots, m$
$\lfloor \cdot \rfloor$	round down to the nearest integer number, $\lfloor x \rfloor = \max\{c \in \mathbb{Z} \mid c \leq x\}$
$\lceil \cdot \rceil$	round up to the nearest integer number, $\lceil x \rceil = \min\{c \in \mathbb{Z} \mid x \leq c\}$
$[\cdot]$	round to the nearest integer number, $[x] = \begin{cases} \lceil x \rceil & \text{if } x + 0.5 \geq \lceil x \rceil \\ \lfloor x \rfloor & \text{otherwise} \end{cases}$
$\mathcal{C}^k(\mathbb{R}^n, \mathbb{R}^m)$	space of all k-times continuously differentiable functions, i.e., $f \in \mathcal{C}^k(\mathbb{R}^n, \mathbb{R}^m) \Leftrightarrow f_j$ is k-times continuously differentiable for all $j = 1, \dots, m$
$\mathbb{IR}^n$	set of all n-dimensional boxes
$\|x\|$	the Euclidean norm of a vector $x \in \mathbb{R}^n$: $\|x\| := \sqrt{\sum_{i=1}^n x_i^2}$
$\|x\|_1$	l_1 norm or sum norm of a vector $x \in \mathbb{R}^n$: $\|x\|_1 := \sum_{i=1}^n \|x_i\|$
$\text{int}(S)$	the interior of the set S
$\text{bd}(S)$	the boundary of the set S

$\mathrm{cl}(S)$	the closure of the set S: $\mathrm{cl}(S) := S \cup \mathrm{bd}(S)$
e	the m-dimensional all-ones vector $e = (1, \ldots, 1)^T$
u^j	the j-th unit vector of $\mathbb{R}^m$: $u_i^j = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$
0	the zero element of a real linear space
$f\colon X \to Y$	function f with domain X and image space Y
$x \mapsto f(x)$	f maps the value $f(x)$ to every x of the domain of f
$f_1, \ldots, f_m$	objective functions of (P)
$g_1, \ldots, g_p$	constraint functions of (MOP)
M	feasible set of (P)
X_E	efficient set of (P)
$\frac{\partial}{\partial x_i} h(x)$	partial derivative of the function $h\colon \mathbb{R}^n \to \mathbb{R}$ in direction of x_i of h at the point x
$\nabla h(x)$	gradient of the function $h\colon \mathbb{R}^n \to \mathbb{R}$ at the point x: $\nabla h(x) = (\frac{\partial}{\partial x_1} h(x), \ldots, \frac{\partial}{\partial x_n} h(x))^T$.
$H_h(x)$	Hessian of the function $h\colon \mathbb{R}^n \to \mathbb{R}$ at the point x
$\lambda_{\min}(x)$	smallest eigenvalue of Hessian of a function $h\colon \mathbb{R}^n \to \mathbb{R}$ at the point x. As the function h is not a part of this term, the underlying function is always mentioned in the text.
$\omega(X)$	box width of a box $X = [\underline{x}, \overline{x}]$: $\omega(X) := \|\overline{x} - \underline{x}\|$

List of Tables

List of Figures

List of Algorithms

Bibliography

Primary Sources

[Adj+98] C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. "A Global Optimization Method, αBB, for General Twice-Differentiable Constrained NLPs: I - Theoretical Advances." In: *Computers and Chemical Engineering* 22.9 (1998), pp. 1137–1158 (cit. on pp. 5, 22 sq., 39, 72).

[ACA19] I. Araya, J. Campusano, and D. Aliquintui. "Nonlinear biobjective optimization: improvements to interval branch & bound algorithms." In: *Journal of Global Optimization* 75.1 (2019), pp. 91–110 (cit. on pp. 5, 39, 66, 69).

[Aud+18] C. Audet, J. Bigeon, D. Cartier, S. Le Digabel, and L. Salomon. *Performance indicators in multiobjective optimization.* Tech. rep. G-2018-90. Les Cahiers du GERAD, 2018. URL: http://www.optimization-online.org/DB_FILE/2018/10/6887.pdf (cit. on p. 87).

[Aud+00] Ch. Audet, P. Hansen, B. Jaumard, and G. Savard. "A branch and cut algorithm for nonconvex quadratically constrained quadratic programming." In: *Mathematical Programming* 87.1 (2000), pp. 131–152 (cit. on pp. 5, 39).

[BA06] C. Barrico and C.H. Antunes. "Robustness Analysis in Multi-Objective Optimization Using a Degree of Robustness Concept." In: *IEEE Congress on Evolutionary Computation. CEC 2006.* IEEE Computer Society, 2006, pp. 1887–1892 (cit. on p. 8).

[Bel+13] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan. "Mixed-integer nonlinear optimization." In: *Acta Numerica* 22 (2013), pp. 1–131 (cit. on p. 106).

[BGN09] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, 2009 (cit. on p. 8).

[BN02] A. Ben-Tal and A. Nemirovski. "Robust optimization – methodology and applications." In: *Mathematical Programming* 92.3 (2002), pp. 453–480 (cit. on p. 8).

[Ben98a] H. P. Benson. "An outer approximation algorithm for generating all efficient extreme points in the outcome set of a multiple objective linear programming problem." In: *Journal of Global Optimization* 13.1 (1998), pp. 1–24 (cit. on pp. 4, 26 sq., 45).

[Ben98b] H. P. Benson. "Further Analysis of an Outcome Set-Based Algorithm for Multiple-Objective Linear Programming." In: *Journal of Optimization Theory and Applications* 97.1 (1998), pp. 1–10 (cit. on p. 26).

[BK97] T. T. Binh and U. Korn. "MOBES: A multiobjective evolution strategy for constrained optimization problems." In: *The Third International Conference on Genetic Algorithms (Mendel 97)*. Vol. 25. 1997, p. 27 (cit. on p. 155).

[BTE18] T. Boeck, D. Terzijska, and G. Eichfelder. "Maximum electromagnetic drag configurations for a translating conducting cylinder with distant magnetic dipoles." In: *Journal of Engineering Mathematics* 108.1 (2018), pp. 123–141 (cit. on pp. 82, 84).

[BCS15] N. Boland, H. Charkhgard, and M. Savelsbergh. "A criterion space search algorithm for biobjective integer programming: The balanced box method." In: *INFORMS Journal on Computing* 27.4 (2015), pp. 735–754 (cit. on p. 6).

[BCS16] N. Boland, H. Charkhgard, and M. Savelsbergh. "The L-shape search method for triobjective integer programming." In: *Mathematical Programming Computation* 8.2 (2016), pp. 217–251 (cit. on p. 6).

[BCS17a] N. Boland, H. Charkhgard, and M. Savelsbergh. "A new method for optimizing a linear function over the efficient set of a multiobjective integer program." In: *European Journal of Operational Research* 260.3 (2017), pp. 904–919 (cit. on p. 6).

[BCS17b] N. Boland, H. Charkhgard, and M. Savelsbergh. "The Quadrant Shrinking Method: A simple and efficient algorithm for solving tri-objective integer programs." In: *European Journal of Operational Research* 260.3 (2017), pp. 873–885 (cit. on p. 6).

[Bol+12] N. L. Boland, A. C. Eberhard, F. Engineer, and A. Tsoukalas. "A new approach to the feasibility pump in mixed integer programming." In: *SIAM Journal on Optimization* 22.3 (2012), pp. 831–861 (cit. on p. 110).

[Bon+09] P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot. "A feasibility pump for mixed integer nonlinear programs." In: *Mathematical Programming* 119.2 (2009), pp. 331–352 (cit. on p. 110).

[Bra11] L. Bradstreet. "The hypervolume indicator for multi-objective optimisation: calculation and use." PhD thesis. University of Western Australia Perth, 2011 (cit. on p. 87).

[BFM98] D. Bremner, K. Fukuda, and A. Marzetta. "Primal—dual methods for vertex and facet enumeration." In: *Discrete & Computational Geometry* 20.3 (1998), pp. 333–357 (cit. on p. 28).

[BKR17] R. S. Burachik, C. Y. Kaya, and M. M. Rizvi. "A new scalarization technique and new algorithms to generate Pareto fronts." In: *SIAM Journal on Optimization* 27.2 (2017), pp. 1010–1034 (cit. on p. 7).

[BKR19] R. S. Burachik, C. Y. Kaya, and M. M. Rizvi. "Algorithms for Generating Pareto Fronts of Multi-objective Integer and Mixed-Integer Programming Problems." arXiv:1903.07041v1. 2019 (cit. on p. 7).

[CD17] V. Cacchiani and C. D'Ambrosio. "A branch-and-bound based heuristic algorithm for convex multi-objective MINLPs." In: *European Journal of Operational Research* 260.3 (2017), pp. 920–933 (cit. on p. 7).

[Cam+18] E. F. Campana, M. Diez, G. Liuzzi, S. Lucidi, R. Pellegrini, V. Piccialli, F. Rinaldi, and A. Serani. "A Multi-objective DIRECT algorithm for ship hull optimization." In: *Computational Optimization and Applications* 71.1 (2018), pp. 53–72 (cit. on pp. 5, 39).

[CGC00] L. G. Casado, I. García, and T. Csendes. "A New Multisection Technique in Interval Methods for Global Optimization." In: *Computing* 65.3 (2000-12), pp. 263–269 (cit. on p. 69).

[CCC00a] M. Csaba Markót, T. Csendes, and A. E. Csallner. "Multisection in Interval Branch-and-Bound Methods for Global Optimization II. Numerical Tests." In: *Journal of Global Optimization* 16.3 (2000-03), pp. 219–228 (cit. on p. 69).

[CCC00b] A. E. Csallner, T. Csendes, and M. Csaba Markót. "Multisection in Interval Branch-and-Bound Methods for Global Optimization – I. Theoretical Results." In: *Journal of Global Optimization* 16.4 (2000-04), pp. 371–392 (cit. on p. 69).

[CM18] A. L. Custódio and J. F. A. Madeira. "MultiGLODS: Global and Local Multiobjective Optimization Using Direct Search." In: *Journal of Global Optimization* 72.2 (2018), pp. 323–345 (cit. on pp. 4, 73).

[DLR13] M. De Santis, S. Lucidi, and F. Rinaldi. "A new class of functions for measuring solution integrality in the Feasibility Pump approach." In: *SIAM Journal on Optimization* 23.3 (2013), pp. 1575–1606 (cit. on p. 110).

[Deb99] K. Deb. "Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems." In: *Evolutionary Computation* 7.3 (1999), pp. 205–230 (cit. on pp. 4, 79, 90).

[Deb01] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Vol. 16. John Wiley & Sons, 2001 (cit. on pp. 4, 73, 86, 90, 92).

[DG01] K. Deb and T. Goel. "Controlled elitist non-dominated sorting genetic algorithms for better convergence." In: *International conference on evolutionary multi-criterion optimization*. Springer. 2001, pp. 67–81 (cit. on p. 86).

[DG06] K. Deb and H. Gupta. "Introducing Robustness in Multi-Objective Optimization." In: *Evolutionary Computation* 14.4 (2006), pp. 463–494 (cit. on p. 8).

[DJ12] K. Deb and H. Jain. "Handling many-objective problems using an improved NSGA-II procedure." In: *2012 IEEE Congress on Evolutionary Computation*. IEEE. 2012, pp. 1–8 (cit. on p. 72).

[Deb+02a] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. "A fast and elitist multiobjective genetic algorithm: NSGA-II." In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 181–197 (cit. on pp. 4, 73).

[Deb+02b] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. "Scalable multi-objective optimization test problems." In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*. Vol. 1. 2002, pp. 825–830 (cit. on pp. 89 sq.).

[Dür01] M. Dür. "Dual bounding procedures lead to convergent branch–and–bound algorithms." In: *Mathematical Programming* 91.1 (2001), pp. 117–125 (cit. on pp. 5, 39).

[Ehr05] M. Ehrgott. *Multicriteria Optimization*. New York: Springer, 2005 (cit. on pp. 3, 11–14, 27, X).

[EG07] M. Ehrgott and X. Gandibleux. "Bound sets for biobjective combinatorial optimization problems." In: *Computers & Operations Research* 34 (2007), pp. 2674–2694 (cit. on p. 6).

[ELS12] M. Ehrgott, A. Löhne, and L. Shao. "A dual variant of Benson's "outer approximation algorithm" for multiple objective linear programming." In: *Journal of Global Optimization* 52.4 (2012), pp. 757–778 (cit. on p. 26).

[ESS11] M. Ehrgott, L. Shao, and A. Schöbel. "An approximation algorithm for convex multi-objective programming problems." In: *Journal of Global Optimization* 50.3 (2011), pp. 397–416 (cit. on pp. 26 sq., 45, 48, 115, 145).

[Ehr+09] M. Ehrgott, C. Waters, R. Kasimbeyli, and O. Ustun. "Multiobjective programming and multiattribute utility functions in portfolio optimiza-

tion." In: *INFOR Information Systems and Operational Research* 47.1 (2009), pp. 31–42 (cit. on p. 105).

[Eic08] G. Eichfelder. *Adaptive Scalarization Methods in Multiobjective Optimization.* New York: Springer, 2008 (cit. on p. 3).

[EG19] G. Eichfelder and T. Gerlach. "On classes of set optimization problems which are reducible to vector optimization problems and its impact on numerical test instances." In: *Variational Analysis and Set Optimization.* Ed. by A. Khan, E. Köbis, and C. Tammer. CRC Press (Taylor & Francis Group), 2019. Chap. 10, pp. 265–290 (cit. on p. 136).

[EKS17] G. Eichfelder, C. Krüger, and A. Schöbel. "Decision uncertainty in multiobjective optimization." In: *Journal of Global Optimization* 69.2 (2017), pp. 485–510 (cit. on pp. 7 sqq., 136 sqq.).

[EP13] Y. G. Evtushenko and M. A. Posypkin. "Nonuniform covering method as applied to multicriteria optimization problems with guaranteed accuracy." In: *Computational Mathematics and Mathematical Physics* 53.2 (2013), pp. 144–157 (cit. on pp. 5, 66).

[EP14] Y. G. Evtushenko and M. A. Posypkin. "A deterministic algorithm for global multi-objective optimization." In: *Optimization Methods and Software* 29.5 (2014), pp. 1005–1019 (cit. on pp. 5, 17, 66).

[Far02] M. Farina. "A neural network based generalized response surface multiobjective evolutionary algorithm." In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600).* Vol. 1. 2002, pp. 956–961 (cit. on pp. 90 sq.).

[FT09] J. Fernández and B. Tóth. "Obtaining the efficient set of nonlinear biobjective optimization problems via interval branch-and-bound methods." In: *Computational Optimization and Applications* 42.3 (2009), pp. 393–419 (cit. on pp. 5, 17, 39, 41, 44, 49, 52 sq., 66 sqq., 75).

[FGL05] M. Fischetti, F. Glover, and A. Lodi. "The feasibility pump." In: *Mathematical Programming* 104.1 (2005), pp. 91–104 (cit. on p. 109).

[FV16] J. Fliege and A. I. F. Vaz. "A method for constrained multiobjective optimization based on SQP techniques." In: *SIAM Journal on Optimization* 26.4 (2016), pp. 2091–2119 (cit. on p. 3).

[FW14] J. Fliege and R. Werner. "Robust multiobjective optimization & applications in portfolio optimization." In: *European Journal of Operational Research* 234.2 (2014), pp. 422–433 (cit. on p. 8).

[FF95] C. Fonseca and P. Fleming. "Multiobjective genetic algorithms made easy: selection sharing and mating restriction." In: *Proceedings of the 1st International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*. IEEE Press. Piscataway, NJ, 1995, pp. 45–52 (cit. on pp. 4, 78, 90).

[Gei+17] B. Geißler, A. Morsi, L. Schewe, and M. Schmidt. "Penalty alternating direction methods for mixed-integer optimization: A new view on feasibility pumps." In: *SIAM Journal on Optimization* 27.3 (2017), pp. 1611–1636 (cit. on p. 110).

[GW90] C. Gerth and P. Weidner. "Nonconvex separation theorems and some applications in vector optimization." In: *Journal of Optimization Theory and Applications* 67.2 (1990), pp. 297–320 (cit. on p. 29).

[Gle+18] A. Gleixner, M. Bastubbe, L. Eifler, T. Gally, G. Gamrath, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. E. Lübbecke, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, F. Schlösser, C. Schubert, F. Serrano, Y. Shinano, J. M. Viernickel, M. Walter, F. Wegscheider, J. T. Witt, and J. Witzig. *The SCIP Optimization Suite 6.0*. ZIB-Report 18-26. Zuse Institute Berlin, 2018-07 (cit. on p. 122).

[Gob+14] M. A. Goberna, V. Jeyakumar, G. Li, and J. Vicente-Pérez. "Robust Solutions of MultiObjective Linear Semi-Infinite Programs under Constraint Data Uncertainty." In: *SIAM Journal on Optimization* 24.3 (2014), pp. 1402–1419 (cit. on p. 8).

[GDC14] A. Goldsztejn, F. Domes, and B. Chevalier. "First order rejection tests for multiple-objective optimization." In: *Journal of Global Optimization* 58.4 (2014), pp. 653–672 (cit. on pp. 67 sq.).

[GLW07] O. Günlük, J. Lee, and R. Weismantel. "MINLP strengthening for separable convex quadratic transportation-cost UFL." In: *IBM Research Report* (2007), pp. 1–16 (cit. on p. 105).

[Gur18] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2018. URL: http://www.gurobi.com (cit. on pp. 122, 131).

[GJN06] C. Gutiérrez, B. Jiménez, and V. Novo. "A Unified Approach and Optimality Conditions for Approximate Solutions of Vector Optimization Problems." In: *SIAM Journal on Optimization* 17.3 (2006), pp. 688–710 (cit. on p. 16).

[HP09] W. W. Hager and D. T. Phan. "An ellipsoidal branch and bound algorithm for global optimization." In: *SIAM Journal on Optimization* 20.2 (2009), pp. 740–758 (cit. on pp. 5, 39).

[HLR14] A. H. Hamel, A. Löhne, and B. Rudloff. "Benson type algorithms for linear vector optimization and applications." In: *Journal of Global Optimization* 59.4 (2014), pp. 811–836 (cit. on p. 26).

[Han+14] Y.-Y. Han, D. Gong, X.-Y. Sun, and Q.-K. Pan. "An improved NSGA-II algorithm for multi-objective lot-streaming flow shop scheduling problem." In: *International Journal of Production Research* 52.8 (2014), pp. 2211–2231 (cit. on p. 72).

[Han92] E. Hansen. *Global Optimzation Using Interval Analysis*. Dekker, 1992 (cit. on pp. 19, 21).

[HT10] E. M. T. Hendrix and B. G. Tóth. *Introduction to Nonlinear and Global Optimization*. New York: Springer, 2010 (cit. on p. 4).

[Hub+06] S. Huband, P. Hingston, L. Barone, and L. While. "A review of multiobjective test problems and a scalable test problem toolkit." In: *IEEE Transactions on Evolutionary Computation* 10.5 (2006), pp. 477–506 (cit. on p. 91).

[Ide+14] Jonas Ide, Elisabeth Köbis, Daishi Kuroiwa, Anita Schöbel, and Christiane Tammer. "The relationship between multi-objective robustness concepts and set-valued optimization." In: *Fixed Point Theory and Applications* 2014.1 (2014), p. 83 (cit. on p. 162).

[Jah06] J. Jahn. "Multiobjective search algorithm with subdivision technique." In: *Computational Optimization and Applications* 35.2 (2006), pp. 161–175 (cit. on p. 4).

[Jah11] J. Jahn. *Vector Optimization - Theory, Applications, and Extensions*. New York: Springer, 2011 (cit. on pp. 3, XI).

[Jah13] J. Jahn. "Vectorization in Set Optimization." In: *Journal of Optimization Theory and Applications* 167.3 (2013), pp. 783–795 (cit. on pp. 12, 138).

[Jah15] J. Jahn. "A derivative-free descent method in set optimization." In: *Computational Optimization and Applications* 60.2 (2015), pp. 393–411 (cit. on pp. 8 sq., 138).

[Jah18] J. Jahn. "A derivative-free rooted tree method in nonconvex set optimization." In: *Pure and Applied Functional Analysis 3* (2018), pp. 603–623 (cit. on p. 9).

[JH11] J. Jahn and T. X. D. Ha. "New order relations in set optimization." In: *Journal of Optimization Theory and Applications* 148 (2011), pp. 209–236 (cit. on pp. 8, 139).

[JPS93] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. "Lipschitzian optimization without the Lipschitz constant." In: *Journal of Optimization Theory and Applications* 79.1 (1993), pp. 157–181 (cit. on p. 5).

[KTZ15] A. A. Khan, C. Tammer, and C. Zălinescu. *Set-valued Optimization*. Springer, 2015 (cit. on p. 139).

[KW05] I. Y. Kim and O. L. de Weck. "Adaptive weighted-sum method for bi-objective optimization: Pareto front generation." In: *Structural and multidisciplinary optimization* 29.2 (2005), pp. 149–158 (cit. on p. 90).

[KSS15] P. Kirst, O. Stein, and P. Steuermann. "Deterministic upper bounds for spatial branch-and-bound methods in global minimization with nonconvex constraints." In: *TOP* 23.2 (2015), pp. 591–616 (cit. on pp. 4, 39, 75).

[Kla17] K. Klamroth. Personal communication. 2017 (cit. on p. 35).

[KLV15] K. Klamroth, R. Lacour, and D. Vanderpooten. "On the representation of the search region in multi-objective optimization." In: *European Journal of Operational Research* 245.3 (2015), pp. 767–778 (cit. on pp. 34–39, XIX).

[Kla09] Diethard Klatte. "First order constraint qualifications." In: *Encyclopedia of Optimization* (2009), pp. 1055–1060 (cit. on p. XI).

[KK16] E. Köbis and M.A. Köbis. "Treatment of set order relations by means of a nonlinear scalarization functional: a full characterization." In: *Optimization* 65.10 (2016), pp. 1805–1827 (cit. on p. 9).

[Krü18a] C. Krüger. "On Minmax Robustness for Multiobjective Optimization with Decision or Parameter Uncertainty." PhD thesis. Georg-August Universität Göttingen, 2018 (cit. on p. 8).

[Krü18b] C. Krüger. "Peat and pots: Analysis of robust solutions for a biobjective problem in agriculture." In: *Preprint-Reihe, Institut für Numerische und Angewandte Mathematik, Georg-August Universität Göttingen* (2018) (cit. on pp. 8 sq.).

[Krü+18] C. Krüger, F. Castellani, J. Geldermann, and A. Schöbel. "Peat and pots: An application of robust multiobjective optimization to a mixing problem in agriculture." In: *Computers and Electronics in Agriculture* 154 (2018), pp. 265–275 (cit. on pp. 8 sq.).

[Kur98] D. Kuroiwa. "The natural criteria in set-valued optimization." In: *RIMS Kokyuroku* 1031 (1998), pp. 85–90 (cit. on p. 8).

[KL12] D. Kuroiwa and G. M. Lee. "On Robust Multiobjective Optimization." In: *Vietnam Journal of Mathematics* 40.2&3 (2012), pp. 305–317 (cit. on p. 8).

[Kut79] S. S. Kutateladze. "Convex ε-programming." In: *Soviet Mathematics - Doklady* 20 (1979), pp. 391–393 (cit. on p. 16).

[LS13a] D. Lera and Y. D. Sergeyev. "Acceleration of univariate global optimization algorithms working with Lipschitz functions and Lipschitz first derivatives." In: *SIAM Journal on Optimization* 23.1 (2013), pp. 508–529 (cit. on p. 5).

[LP09] A. S. Lewis and C. H. J. Pang. "Lipschitz Behavior of the Robust Regularization." In: *SIAM Journal on Control and Optimization* 48.5 (2009), pp. 3080–3105 (cit. on p. 8).

[LS13b] M. Locatelli and F. Schoen. *Global Optimization: Theory, Algorithms, and Applications*. Philadelphia: SIAM, 2013. PA (cit. on p. 5).

[LRU14] A. Löhne, B. Rudloff, and F. Ulus. "Primal and dual approximation algorithms for convex vector optimization problems." In: *Journal of Global Optimization* 60.4 (2014), pp. 713–736 (cit. on pp. 4, 26, 115, 147).

[LS13c] A. Löhne and C. Schrage. "An algorithm to solve polyhedral convex set optimization problems." In: *Optimization* 62.1 (2013), pp. 131–141 (cit. on p. 8).

[LW17] A. Löhne and B. Weißing. "The vector linear program solver Bensolve – notes on theoretical background." In: *European Journal of Operational Research* 260.3 (2017), pp. 807–813 (cit. on p. 28).

[Lor84] P. Loridan. "ε-solutions in vector minimization problems." In: *Journal of Optimization Theory and Applications* 43.2 (1984), pp. 265–276 (cit. on p. 16).

[MF94] C. D. Maranas and C. A. Floudas. "Global minimum potential energy conformations of small molecules." In: *Journal of Global Optimization* 4.2 (1994), pp. 135–170 (cit. on pp. 5, 22, 24, 42, 142).

[Mar+16] B. Martin, A. Goldsztejn, L. Granvilliers, and C. Jermann. "Constraint propagation using dominance in interval Branch & Bound for nonlinear biobjective optimization." In: *European Journal of Operational Research* 260.3 (2016), pp. 934–948 (cit. on pp. 5, 39, 66, 68 sqq.).

[MAT18a] MATLAB. *9.4.0.813654 (R2018a)*. Natick, Massachusetts: The MathWorks Inc., 2018 (cit. on pp. 77, 91, 122).

[MAT18b] MATLAB. *Global Optimization Toolbox*. Natick, Massachusetts: The MathWorks Inc., 2018 (cit. on pp. 74, 86).

[MAT18c] MATLAB. *Optimization Toolbox*. Natick, Massachusetts: The MathWorks Inc., 2018 (cit. on p. 77).

[Mav09] G. Mavrotas. "Effective implementation of the ε-constraint method in multi-objective mathematical programming problems." In: *Applied Mathematics and Computation* 213.2 (2009), pp. 455–465 (cit. on p. 6).

[MD98] G. Mavrotas and D. Diakoulaki. "A branch and bound algorithm for mixed zero-one multiple objective linear programming." In: *European Journal of Operational Research* 107.3 (1998), pp. 530–541 (cit. on p. 6).

[MD05] G. Mavrotas and D. Diakoulaki. "Multi-criteria branch and bound: A vector maximization algorithm for mixed 0-1 multiple objective linear programming." In: *Applied Mathematics and Computation* 171.1 (2005), pp. 53–71 (cit. on p. 6).

[McC76] G. P. McCormick. "Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems." In: *Mathematical Programming* 10.1 (1976), pp. 147–175 (cit. on p. 20).

[Mie12] K. Miettinen. *Nonlinear multiobjective optimization*. Vol. 12. Springer Science & Business Media, 2012 (cit. on p. X).

[Moo66] R. E. Moore. *Interval analysis*. Vol. 4. Prentice-Hall Englewood Cliffs, NJ, 1966 (cit. on pp. 19, 21).

[MKC09] R. E. Moore, R. B. Kearfott, and M. J. Cloud. *Introduction to interval analysis*. Vol. 110. Siam, 2009 (cit. on pp. 19 sqq.).

[MGS09] D. Mueller-Gritschneder, H. Graeb, and U. Schlichtmann. "A successive approach to compute the bounded Pareto front of practical multiobjective optimization problems." In: *SIAM Journal on Optimization* 20.2 (2009), pp. 915–934 (cit. on p. 3).

[Neu90] A. Neumaier. *Interval methods for systems of equations.* Cambridge: Cambridge University Press, 1990 (cit. on pp. 19 sqq., 52).

[PŽŽ17] P. M. Pardalos, A. Žilinskas, and J. Žilinskas. *Non-Convex Multi-Objective Optimization.* New York: Springer, 2017 (cit. on p. 5).

[PY14] Y. Peng and L. Yu. "Multiple criteria decision making in emergency management." In: *Computers & Operations Research* 42 (2014), pp. 1–2 (cit. on p. 105).

[PMC96] C. Poloni, G. Mosetti, and S. Contessi. "Multi objective optimization by GAs: Application to system and component design." In: *ECCOMAS'96: Computational Methods in Applied Sciences' 96.* 1996, pp. 1–7 (cit. on p. 90).

[Roc70] R. T. Rockafellar. *Convex analysis.* Princeton university press, 1970 (cit. on pp. 27, XI).

[Roc16] S. Rocktäschel. "Ein Algorithmus zur Bestimmung einer Lösungsüberdeckung spezieller mengenwertiger Optimierungsprobleme." Bachelor's Thesis. Institute for Mathematics, Technische Universität Ilmenau, Germany, 2016-10-18 (cit. on p. 10).

[Roc18] S. Rocktäschel. "A BB algorithm for multiobjective mixed-integer convex optimization." Master's Thesis. Institute for Mathematics, Technische Universität Ilmenau, German, 2018-09-03 (cit. on pp. 10, 106).

[RS07] L. Rodríguez-Marín and M. Sama. "(Λ, C)-contingent derivatives of set-valued maps." In: *Journal of Mathematical Analysis and Applications* 335.2 (2007), pp. 974–989 (cit. on p. 139).

[Rum99] S. M. Rump. "INTLAB - INTerval LABoratory." In: *Developments in Reliable Computing.* Ed. by Tibor Csendes. http://www.ti3.tuhh.de/rump/. Dordrecht: Kluwer Academic Publishers, 1999, pp. 77–104 (cit. on pp. 21, 24, 77, 155).

[SNT85] Y. Sawaragi, H. Nakayama, and T. Tanino. *Theory of Multiobjective Optimization.* New York: Academic Press, 1985 (cit. on pp. 62, IX).

[SKW02] B. Schandl, K. Klamroth, and M. M. Wiecek. "Norm-based approximation in multicriteria programming." In: *Computers & Mathematics with Applications* 44.7 (2002), pp. 925–942 (cit. on p. 4).

[Sch12] D. Scholz. *Deterministic Global Optimization: Geometric Branch-and-bound Methods and their Applications*. New York: Springer, 2012 (cit. on pp. 5 sq., 16, 39, 65 sqq., 70 sq., 75).

[Sch95] J. R. Schott. "Fault tolerant design using single and multicriteria genetic algorithm optimization." MA thesis. Massachusetts Institute of Technology, 1995 (cit. on p. 88).

[SM16] M. Schulze Darup and M. Mönnigmann. "Improved Automatic Computation of Hessian Matrix Spectral Bounds." In: *SIAM Journal on Scientific Computing* 38.4 (2016), A2068–A2090 (cit. on p. 23).

[Sha17] N. Shafiei. "Benson's algorithm for nonconvex multiobjective problems via nonsmooth Wolfe duality." In: *Bulletin of the Iranian Mathematical Society* 43.5 (2017), pp. 975–994 (cit. on p. 26).

[SS08] F. Sourd and O. Spanjaard. "A Multiobjective Branch-and-Bound Framework: Application to the Biobjective Spanning Tree Problem." In: *INFORMS Journal on Computing* 20.3 (2008), pp. 472–484 (cit. on p. 6).

[Ste17] Oliver Stein. *Grundzüge der Globalen Optimierung*. Springer-Verlag, 2017 (cit. on p. 20).

[TH88] H. Tuy and R. Horst. "Convergence and restart in branch-and-bound algorithms for global optimization. Application to concave minimization and d.c. optimization problems." In: *Mathematical Programming* 41.1-3 (1988), pp. 161–183 (cit. on pp. 5, 39).

[Van99] D. A. Van Veldhuizen. "Multiobjective evolutionary algorithms: classifications, analyses, and new innovations." PhD thesis. School of Engineering of the Air Force Institute of Technology, 1999 (cit. on p. 89).

[VFM96] R. Viennet, C. Fonteix, and I. Marc. "Multicriteria optimization using a genetic algorithm for determining a Pareto set." In: *International Journal of Systems Science* 27.2 (1996), pp. 255–260 (cit. on pp. 4, 82, 90).

[Wan+08] L. Wang, H. Ishida, T. Hiroyasu, and M. Miki. "Examination of multiobjective optimization method for global search using DIRECT and GA." In: *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. Piscataway. 2008, pp. 2446–2451. NJ (cit. on pp. 5, 39).

[XMP10] P. Xidonas, G. Mavrotas, and J. Psarras. "Equity portfolio construction and selection using multiobjective mathematical programming." In: *Journal of Global Optimization* 47.2 (2010), pp. 185–209 (cit. on p. 105).

[ZMK19] Y. Zhou-Kangas, K. Miettinen, and K. K. Sindhya. "Solving multiobjective optimization problems with decision uncertainty: an interactive approach." In: *Journal of Business Economics* 89.1 (2019), pp. 25–51 (cit. on p. 8).

[ŽŽ16] A. Žilinskas and J. Žilinskas. "Adaptation of a one-step worst-case optimal univariate algorithm of bi-objective Lipschitz optimization to multidimensional problems." In: *Communications in Nonlinear Science and Numerical Simulation* 21.1-3 (2016), pp. 89–98 (cit. on pp. 5, 39, 66, 69, 75).

[ZDT00] E. Zitzler, K. Deb, and L. Thiele. "Comparison of multiobjective evolutionary algorithms: Empirical results." In: *Evolutionary computation* 8.2 (2000), pp. 173–195 (cit. on p. 90).

Publications and Preprints

In relation to the work on this thesis, the following publications were published in international journals and conference proceedings or are available as preprints. They are sorted in descending chronological order.

[ENR19] G. Eichfelder, J. Niebling, and S. Rocktäschel. "An Algorithmic Approach to Multiobjective Optimization with Decision Uncertainty." In: *Journal of Global Optimization* (2019-07-27). DOI: 10.1007/s10898-019-00815-9 (cit. on pp. 10, 135, 161).

[DeS+19] M. De Santis, G. Eichfelder, J. Niebling, and S. Rocktäschel. "Solving Multiobjective Mixed Integer Convex Optimization Problems." In: *Preprint-Series of the Institute for Mathematics, Technische Universität Ilmenau, Germany* (2019-05). URL: http://www.optimization-online.org/DB_HTML/2019/05/7229.html (cit. on pp. 10, 132).

[NE19] J. Niebling and G. Eichfelder. "A Branch-and-Bound-based Algorithm for Nonconvex Multiobjective Optimization." In: *SIAM Journal on Optimization* 29.1 (2019-03), pp. 794–821 (cit. on pp. 9, 35, 42).

[EKN19] G. Eichfelder, K. Klamroth, and J. Niebling. "Using a B&B Algorithm from Multiobjective Optimization to Solve Constrained Optimization Problems." In: *AIP Conference Proceedings*. Vol. 2070. 2019-02, p. 020028.

[NE16] J. Niebling and G. Eichfelder. "A Branch-and-Bound-Algorithm for Bi-Objective Problems." In: *Proceedings of the XIII Global Optimization Workshop GOW'16*. Braga, Portugal, 2016-09, pp. 57–60.