Department of Computer Architecture and Electronics
University of Almería

Interval Methods for Competitive Location Problems

Boglárka Tóth

Main supervisor: Dr. José Fernández Hernández
Co-supervisors: Dr. Blas Pelegrín Pelegrín
Dr. Leocadio González Casado

2007

Acknowledgments

First of all, my heartfelt appreciation goes to my official and non-official supervisors: Dr. Tibor Csendes, Dr. José Fernández Hernández, Dr. Leocadio González Casado and Dr. Blas Pelegrín Pelegrín.

Tibor Csendes was the person who encouraged me to apply for a PhD studentship and introduced me to the beauty of reliable methods. I am grateful for his early guidance and inspiration during the years I have spent at the University of Szeged. On his proposal I have spent one semester at the University of Almería with an Erasmus grant, and he also patronized me to apply for the grant awarded by the Ministry of Education and Science of the Spanish government. He helped me a great deal during the last years despite the large distance, thus I acknowledge his supervision a lot.

José Fernández Hernández, my main director, introduced me to the flourishing field of locational analysis. With his guidance and help we have written a large part of the papers summarized in this thesis. I admire his enthusiastic way of working and it made me become as hard a worker as he is. I am really thankful for his inspiration and for showing me the way to being efficient in research. Finally, I have to admit, that I really appreciate our intellectual discussions.

With Leocadio González Casado, we started our joint work during my stay as an Erasmus scholar at the University of Almería. After my return, he recommended me the grant at the University of Murcia, with which I have finally spent almost four years in Spain. I am very thankful to him for the inspirational atmosphere, his guidance on the common works and his endless help in all my doubts on other research I did during my stay.

Last but not least, I am indebted to Blas Pelegrín Pelegrín, who made it possible for me to obtain this grant, that I could go to many conferences, and supported my working background with all the facilities I needed. I am thankful for his valuable suggestions concerning the thesis and for the common works, from which I learnt so much.

I appreciate the help of the Ministry of Education and Science of Spain both under grant BES-2003-2157, which is part of the research project *Análisis y optimización de estrategias para la localización de actividades económicas en situaciones de competencia* (Ref. BEC2002-01026), in part financed by the European Regional Development Fund (ERDF) and under the Hungarian-Spanish bilateral program *Acción Integrada Hispano-Húngaro* (Ref. HH2004-0014).

I am grateful for both the Institute of Informatics at the University of Szeged, and the Department of Statistics and Operations Research at the University of Murcia for providing me with excellent working conditions.

I would like to thank Professor Inmaculada García Fernández and the Department of Computer Architecture and Electronics at the University of Almería for providing me with a pleasant atmosphere during my stay, where most of the thesis was written. Also, many thanks for her helping me in everyday life.

Thanks to my Hungarian colleagues, Zsolt Gazdag, Balázs Szörényi, and Tamás Vinkó, who are also my friends, for the nice conversations about the big (and not so big) questions of life. I am also thankful to my Spanish colleagues and friends Vicente González Ruíz, José Antonio Martínez García and José Antonio Álvarez Bermejo for the nice breakfasts, swimming, windsurfing, etc.

I am very thankful to my mother for her love and support during my studies. Without her encouragement I could not have done this work, which I therefore dedicate to her.

Finally, my special thanks goes to my fiancé, Zsolti, for his love, tolerance, and understanding during the work on my thesis and in all aspects of my life.

Preface

Spatial competition models deal with the choice of locations from which firms sell goods or provide services. A single firm may operate several facilities in a geographical market and their locations affect not only its market share but also the market share of the other competing firms. Any profit oriented company has to face some location decision problems, whether the firm enters a new market or it is an expanding firm already operating in the market. For that reason, research on that topic is flourishing, both with new models to describe such problems as well as with new methods which are able to solve them.

Localization of competitive facilities is a real-life problem. Thus, it is necessary to describe it as close to reality as possible. The designed models usually either maximize profit/income or minimize costs, and the objective functions are normally rather complex, neither convex nor concave. Therefore, competitive facility location problems are usually global optimization problems, and global optimization techniques are required to cope with them.

Global optimization is a huge field of research due to the numerous applications in engineering, chemistry, biology, physics, economics, etc. In order to solve global optimization problems, different types of algorithms have been developed. There are many methods, either stochastic or deterministic, which can find local optima, or an approximation of the global optimum. However, for most of the algorithms how close the current solution is to the global optimum after a finite number of steps is not known, nor is the error produced by the finiteness of computer arithmetic. When one is interested in finding the global optimum within a given accuracy with guarantee, reliable global optimization methods have to be used. *Interval Branch and Bound algorithms* are among the most practical methods in that class, because thanks to interval analysis, no errors are committed due to rounding, thus the global optimum can be enclosed in a reliable way within a given accuracy.

The aim of this thesis is twofold. Our first goal is to contribute to reliable global optimization. On the one hand, our objective is to provide new accelerating devices for the well-known interval Branch and Bound method in order to make it faster and applicable to a wider range of problems. On the other hand, we aim to develop new methods to cope with biobjective problems in a reliable way, i.e. to find either a specific part of the efficient set or an enclosure of the whole efficient set.

Our second goal is to study new realistic models for planar competitive facility location problems. In particular, we are going to construct models where the attraction of a costumer towards a facility depends not only on the distance to it, but also on the quality that the costumer perceives from the facility. As in real life problems, our aim is to maximize the profit obtained by the company, to be understood as the income obtained from the market share captured by the facilities minus the costs incurred by running them. To show the applicability of the new models, our last goal is to solve problems with real data by using the reliable methods previously developed.

The thesis is organized in two parts corresponding to the two main aims described above. Part I collects our findings on *Interval Methods* in three chapters, while Part II consists of *Competitive Location Problems* in five chapters. Due to the large amount of notation in this thesis a Symbol Index is provided at the end of each part.

In Chapter 1 we introduce the field of global optimization, the notations and properties of interval analysis and its use within the Branch and Bound schema. We discuss the usual rules and accelerating devices of the interval B&B method, and we introduce the notion of δ-optimality as well.

The new accelerating devices developed for the interval B&B method are collected in Chapter 2. First, the empirical convergence speed is introduced and computational studies are presented for a set of test functions. Then, two kinds of multi-dimensional pruning techniques are described and evaluated on a set of test problems. Finally, some modifications of existing discarding tests are described.

In Chapter 3 we present three methods which deal with biobjective problems. Our first method, the Lexicographical-like method, is able to find weakly efficient points from any part of the weakly efficient set. It always takes both objectives into account, and allows a worsening of one of the objectives to be traded off with an improvement of the other objective. Thus, it is specially suitable to represent the preferences of the decision maker. Our second method, the Constrained-like method, is developed to find an enclosure of the whole efficient set. It explores the nondominated set from left-top to right-bottom in small steps, solving a single objective constrained problem in every step. Our last algorithm, the biobjective Branch and Bound method, deals with the two objectives directly, and finds a tight enclosure of the whole efficient set as well. Good convergence properties of the algorithm are also shown.

Part II starts with a brief introduction to competitive location problems in Chapter 4. First, the main ingredients of location analysis are described, and later the different factors of competitive location are discussed. Different multiobjective approaches are also introduced.

In Chapter 5 we introduce a new model for the single facility planar location and design problem. After the description of the concrete interval B&B method used for solving it, some computational results are evaluated. We also study which inclusion function is the best one for this class of problems, using the empirical convergence speed introduced in Chapter 2. Finally an extensive sensitivity analysis is conducted in order to check the importance of the different parameters of the model.

The single facility location problem is extended to the location of two facilities in Chapter 6. The main goal of this chapter is to examine the difference between the solutions obtained by the sequential and simultaneous approaches. The sequential approach first determines the location and design of one facility, and then, taking it as an existing facility, determines the location and design of the second facility. On the contrary, the simultaneous approach determines the location and quality of the two new facilities at the same time. The results of the two approaches are compared on some test problems.

When a chain decides to build a new facility, it does not usually only care about the new profit it can gain. If the new facility captures part of the market share from other chain-owned existing facilities (an effect known as *cannibalization*), the company may have trouble running the affected facilities, and it may be forced to close some of them. Thus, it is important to minimize cannibalization while maximizing profit. This leads to a biobjective model, which is studied in Chapter 7. The model is solved with the lexicographical-like method introduced in Chapter 3, and an economic analysis is also carried out.

In Chapter 7 the case of a franchise which wants to expand its presence in the market by opening a new outlet is studied. The new facility should bring the largest possible profit to both the franchisor (the owner of the franchise) and the franchisee (the actual owner of the new outlet). These two objectives are conflicting, therefore a biobjective model is built to describe the problem. Computational experiments are carried out using all the biobjective methods introduced in Chapter 3. The effectiveness of the developed accelerating devices are tested, and the methods are compared using some examples of the franchise problem.

The thesis concludes with a summary where the main results are outlined and future lines of research are presented.

Contents

Part I

Interval Methods

Chapter 1

Introduction to Interval Methods

The general optimization problem, in its minimization form, is defined as follows: Given a nonempty closed set S and a function f, find the minimal value f^* and all the points $x^* \in S$ such that $f^* = f(x^*) \leq f(x)$ for all $x \in S$, or show that no such a point exists. The problem can be written as

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in S. \end{array} \tag{1.1}$$

We want to find the global minimum value f^* and the set of global minimizer points $X^* = \{x^* \in S \mid f(x^*) = f^*\}$. The conversion of a maximization problem to a minimization one is straightforward ($\max\{f(x) \mid x \in S\} = -\min\{-f(x) \mid x \in S\}$). Hence, in this part of the thesis only minimization problems will be treated.

In this chapter we point out the type of problems belonging to global optimization and the methods which can be used to solve those problems. In Section 1.2 the Branch and Bound methods are introduced. In Section 1.3 we describe the notion and properties of Interval Analysis together with the different inclusion functions. The interval Branch and Bound method is described in Section 1.4, and finally the δ-optimal region is introduced in Section 1.5.

1.1 Global optimization

When any local minimum of problem (1.1) is also a global minimum (e.g., in linear or convex programming), local optimization methods can find the optimum easily. However, in many problems this is not the case. The field of global optimization is devoted to those problems of type (1.1) which can have several local optima apart from the global optimum. The algorithms constructed to solve these problems can be classified in different ways. A usual classification distinguishes between deterministic or stochastic methods, depending on how the algorithm works. Stochastic methods (for instance, tabu search, genetic algorithms or simulated annealing) apply some random factors to avoid getting trapped in local optima. However, they do not necessarily find the global optimum, and even if they found it, one would not know of it. In deterministic methods (such as outer approximation, Lipschitzian optimization or Branch and Bound) no random factors are included. Some of them converge to the global optimum under some conditions, but when the algorithm is stopped after a finite number of iterations the accuracy of the solution may not be known with exactness.

A better classification of the algorithms, based on the degree of rigor with which they find the global optimum, is the following [112]: *incomplete methods*, which may get stuck in a local optimum; *asymptotically complete methods*, which reach a global optimizer with probability one if allowed to run infinitely long, but have no means of knowing when a global minimizer has been found; *complete methods*, which reach a global optimizer with certainty (assuming exact computations and infinitely long run), but knowing after a finite time that an approximate global optimizer has been found; and finally *rigorous methods*, which reach the global optimizers with certainty within a given tolerance even in the presence

of rounding errors. We will focus on methods of the last class, in particular in Branch and Bound methods, which are most commonly used among complete and rigorous methods for nonlinear nonconvex problems.

1.2 Branch and Bound methods

The Branch and Bound schema was first proposed by A. H. Land and A. G. Doig in 1960 for linear programming [95]. This type of methods is used for finding optimal solutions of various optimization problems, especially in discrete and combinatorial optimization. It belongs to the class of implicit enumeration methods, and it is used for handling a number of NP-hard problems besides nonlinear optimization, such as the knapsack problem, the traveling salesman problem (from integer programming), and the quadratic assignment problem, to name a few.

The basic idea in B&B methods consists of a recursive decomposition of the original problem into smaller disjoint subproblems until the solution is found. The method avoids visiting those subproblems which are known for not containing a solution. The general schema of the method is as follows: In every step we select (by the *selection rule*) a region (initially the whole search space). Then we divide it (by the *division rule*) into smaller regions and find lower and upper bounds of the function over these regions (by the *bounding rule*). When a good upper bound of the minimum is found we are able to reject those regions which have a higher lower bound than the best upper bound found. If a region cannot be rejected, it is stored in the working list from where the next box is chosen in every iteration. The algorithm terminates if some specified threshold is met between the best upper and lower bounds found on the minimum, or other measures, as specified in the *termination rule*. The union of the regions which are not discarded contains the global minimizers.

The basic rules of this general method can be chosen differently for the given application. Depending on the problem, the search region can be divided into general polygons or into other special sets, like triangles (see Big Triangle Small Triangle method [34]) or squares (see Big Square Small Square method [77]). In fact, this division depends strongly on how one can compute bounds for the given regions. For instance, for convex or Lipschitz functions the bounding rule can use its specific properties, therefore the shape of the regions is specified accordingly. For our study we only require that the search region be closed and form part of the Euclidean space, and that the function to be minimized be continuous or differentiable (sometimes twice differentiable), i.e $S \subseteq \mathbb{R}^n, f : \mathbb{R}^n \to \mathbb{R}, f \in C^l, l \geq 0, 1,$ or 2 in (1.1). Notice that we cannot use any specific property of the functions (apart from continuity or differentiability) to construct bounding rules. To obtain automatic bounds for such functions one can use Interval Analysis.

1.3 Interval Analysis

Interval arithmetic was developed as an approach to control rounding errors in mathematical computation and thus, to obtain reliable results. Where real arithmetic defines operations on individual numbers, interval arithmetic defines operations on intervals. A form of interval arithmetic probably first appeared in 1924 and 1931 in [8, 155], and later in [139]. Modern development of interval arithmetic began with the first articles by R. E. Moore [105, 108], followed by his dissertation [106] and his book [107]. During the same period, Hansen studied interval manipulation in linear algebra, and a group of German researchers including Alefeld, Krawczyk and Nickel developed many aspects of its computer implementation. Their work inspired many other researchers to experiment in this new field, and as a result, thousands of articles and dozens of books have been published (see [3, 73, 87, 88, 110, 124, 125] for introduction and for advanced methods).

Now, let us introduce the notation of intervals and related concepts, which will be used throughout this thesis.

Real numbers are denoted by $x, y, \ldots$ and compact intervals by $\boldsymbol{x} = [\underline{x}, \overline{x}], \boldsymbol{y} = [\underline{y}, \overline{y}], \ldots$, where $\underline{x} = \min\{x \in \boldsymbol{x}\}$ and $\overline{x} = \max\{x \in \boldsymbol{x}\}$ are the lower and upper bounds of $\boldsymbol{x}$, respectively. The set of compact intervals is denoted by $\mathbb{I} := \{[a, b] \mid a, b \in \mathbb{R},\ a \leq b\}$.

The notation $x = (x_1, \ldots, x_n)^T$ and $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$, where $x_i \in \mathbb{R}$, $\boldsymbol{x}_i \in \mathbb{I}$ $(i = 1, \ldots, n)$ is used for real and interval vectors, respectively. The set of n-dimensional intervals (also called boxes) is denoted by $\mathbb{I}^n$. The set of all subsets of an interval $\boldsymbol{x}$ is denoted by $\mathbb{I}^n(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathbb{I}^n \mid \boldsymbol{y} \subseteq \boldsymbol{x}\}$.

The width of the interval $\boldsymbol{x}$ is defined by $\mathrm{w}(\boldsymbol{x}) = \overline{x} - \underline{x}$, if $\boldsymbol{x} \in \mathbb{I}$, and $\mathrm{w}(\boldsymbol{x}) = \max_{i=1,\ldots,n} \mathrm{w}(\boldsymbol{x}_i)$, when $\boldsymbol{x} \in \mathbb{I}^n$. The midpoint is defined by $\mathrm{m}(\boldsymbol{x}) = (\underline{x} + \overline{x})/2$, where $\underline{x} = (\underline{x}_1, \ldots, \underline{x}_n)$, $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_n)$ when $\boldsymbol{x} \in \mathbb{I}^n$. The relative width is defined by $\mathrm{w}_{rel}(\boldsymbol{x}) = \mathrm{w}(\boldsymbol{x}) / \max\{1, \min_{x \in \boldsymbol{x}} |x|\}$ if $\boldsymbol{x} \in \mathbb{I}$. For $\boldsymbol{x} \in \mathbb{I}^n$ $\mathrm{w}_{rel}(\boldsymbol{x}) = \max_{i=1,\ldots,n} \mathrm{w}_{rel}(\boldsymbol{x}_i)$.

The main idea behind interval analysis is the natural extension of the real arithmetical operations to interval operations. For any interval $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{I}$ and arithmetical operator $\circ \in \{+, -, \cdot, /\}$

$$\boldsymbol{x} \circ \boldsymbol{y} = \{x \circ y \mid x \in \boldsymbol{x}, y \in \boldsymbol{y}\},$$

i.e. the result of the interval operation contains all the possible values which can be obtained as a result of the real operation on any pair of values belonging to the argument intervals. Because of the continuity of the operations these sets are always intervals. Of course for the division operation it is a must that zero does not belong to the denominator. The arithmetic operations are monotonous, thus the definitions of the corresponding interval versions are straightforward:

$$\begin{aligned}
\boldsymbol{x} + \boldsymbol{y} &= [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \\
\boldsymbol{x} - \boldsymbol{y} &= [\underline{x} - \overline{y}, \underline{y} - \overline{x}] \\
\boldsymbol{x} \cdot \boldsymbol{y} &= [\min\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}, \max\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}] \\
\boldsymbol{x} / \boldsymbol{y} &= [\underline{x}, \overline{x}] * \left[\frac{1}{\overline{y}}, \frac{1}{\underline{y}}\right] \quad \text{if } 0 \notin \boldsymbol{y}
\end{aligned}$$

The algebraic properties of the interval arithmetic operations are not the same as those of real arithmetical operations (for instance, the subtraction and division in $\mathbb{I}$ are not the inverse operations of addition and multiplication, respectively), but the main properties from the operational point of view still hold. For instance, addition and multiplication are associative and commutative, the additive identity is [0,0], while the multiplicative identity is [1,1]. Distributivity does not hold, but the so-called subdistributive law is valid:

$$\boldsymbol{x} \cdot (\boldsymbol{y} + \boldsymbol{z}) \subseteq \boldsymbol{x} \cdot \boldsymbol{y} + \boldsymbol{x} \cdot \boldsymbol{z} \text{ for } \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{I}.$$

Let $\phi : D \subseteq \mathbb{R} \to \mathbb{R}$ be an elementary real function predeclared in some programming language (like sin, log, exp, etc.), that is continuous over every interval $\boldsymbol{x} \subseteq D$. The corresponding interval version of the function ϕ can be defined as

$$\boldsymbol{\phi}(\boldsymbol{x}) = \{\phi(x) \mid x \in \boldsymbol{x}\}.$$

The interval version of most of the elementary functions can be easily constructed by using its monotonicity property, for instance

$$\begin{aligned}
e^{\boldsymbol{x}} &= [e^{\underline{x}}, e^{\overline{x}}], \\
\sqrt{\boldsymbol{x}} &= [\sqrt{\underline{x}}, \sqrt{\overline{x}}], \ \boldsymbol{x} > 0.
\end{aligned}$$

For a non-monotonic function, such as sine or cosine, its periodic nature can be used to construct its interval extension. Therefore, if a function can be given as a compound of variables, arithmetical operations and elementary functions, one can construct its interval extension by changing the variables to intervals and every function or operation to its interval extension.

The above inclusions are valid when errors due to the finiteness of computer representation are not assumed. To avoid rounding errors of computer arithmetic for interval operations outward rounding have to be used. It means that computing the lower bound downward rounding, while for the upper bounds upward rounding is used. In this way the resulting interval contains with guarantee all the possible results of the operation.

1.3.1 Inclusion functions

Let $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ (D be a continuous function, and $\mathbb{I}^n(D) = \{\boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{I}^n, \boldsymbol{x} \subseteq D\}$. An inclusion function of f is defined as follows (see Figure 1.1).

Definition 1.1. *The function $\boldsymbol{f} : \mathbb{I}^n(D) \to \mathbb{I}$ is an* inclusion function *of f, if for every $\boldsymbol{x} \in \mathbb{I}^n(D)$ and $x \in \boldsymbol{x}$, $f(x) \in \boldsymbol{f}(\boldsymbol{x})$, i.e. the range $f(\boldsymbol{x}) = \{f(x) \mid x \in \boldsymbol{x}\}$ is included in $\boldsymbol{f}(\boldsymbol{x})$.*

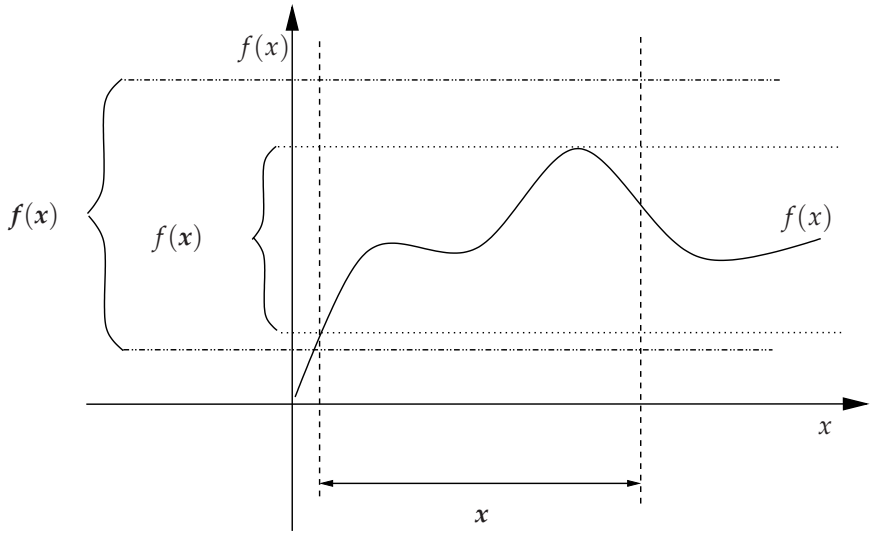


Figure 1.1: $\boldsymbol{f}(\boldsymbol{x})$ is an inclusion function of $f(x)$.

For an inclusion function the most important properties are inclusion isotonicity, zero convergence, and α-convergence. The definitions of these properties are as follows.

Definition 1.2. *An inclusion function $\boldsymbol{f}$ is called* inclusion isotone, *if $\boldsymbol{f}(\boldsymbol{x}) \subseteq \boldsymbol{f}(\boldsymbol{y})$ holds for every $\boldsymbol{x} \subseteq \boldsymbol{y} \in \mathbb{I}^n(D)$.*

Definition 1.3. *If the convergence condition*

$$\lim_{\mathrm{w}(\boldsymbol{x}) \to 0} \mathrm{w}(\boldsymbol{f}(\boldsymbol{x})) - \mathrm{w}(f(\boldsymbol{x})) = 0 \tag{1.2}$$

holds, then the inclusion function $\boldsymbol{f}$ is called zero-convergent.

A similar but somewhat stronger property is the contraction property. As will be demonstrated, for continuous real-valued functions the two properties are equivalent.

Definition 1.4. *An inclusion function $\boldsymbol{f}$ of a function f is said to have the* contraction property *if*

$$\mathrm{w}(\boldsymbol{f}(\boldsymbol{x})) \to 0 \textit{ when } \mathrm{w}(\boldsymbol{x}) \to 0.$$

Proposition 1.5. *For a real-valued continuous function f the contraction property and the zero-convergent property are equivalent.*

Proof. The statement is evident if $\mathrm{w}(f(\boldsymbol{x})) \to 0$ when $\mathrm{w}(\boldsymbol{x}) \to 0$. Suppose there is a nested interval sequence $\{\boldsymbol{x}_i\}_{i=1}^{\infty}$, $\boldsymbol{x}_{i+1} \subseteq \boldsymbol{x}_i, \boldsymbol{x}_i \neq \emptyset, \forall i$, such that $\mathrm{w}(\boldsymbol{x}_i) \underset{i\to\infty}{\longrightarrow} 0$. Then, there exists a point x, such that $x \in \boldsymbol{x}_i, \forall i$, i.e. x is the accumulation point of $\{\boldsymbol{x}_i\}_{i=1}^{\infty}$. Since f is real-valued, $\mathrm{w}(f(x)) = 0$. Because of the continuity of f and as $\boldsymbol{x}_i \underset{i\to\infty}{\longrightarrow} x$, $f(\boldsymbol{x}_i) \underset{i\to\infty}{\longrightarrow} f(x)$. Thus, $\mathrm{w}(f(\boldsymbol{x}_i)) \underset{i\to\infty}{\longrightarrow} \mathrm{w}(f(x)) = 0$, which is what we wanted to see. □

Definition 1.6. The convergence order *of an inclusion function* $\boldsymbol{f}$ *of* f *is* $\alpha(\in \mathbb{R}^+)$*, if the inequality*

$$\mathrm{w}(\boldsymbol{f}(\boldsymbol{x})) - \mathrm{w}(f(\boldsymbol{x})) \leq c\mathrm{w}(\boldsymbol{x})^{\alpha} \tag{1.3}$$

holds for every $\boldsymbol{x} \in \mathbb{I}^n(D)$ *for a real* c*.*

From the last definition it is easy to see that every inclusion function is zero-convergent if it is α-convergent for some α.

An inclusion function generally overestimates the range of a function. The extent of the overestimation depends on the type of the inclusion function, on the considered function and on the width of the interval. If the computational costs are the same, of course the smaller the overestimation is, the better the inclusion function is.

In literature one can find various methods for constructing inclusion functions. Some of them can be used for every continuous function, even for non-differentiable ones, while others are available only for polynomial functions or for one-dimensional functions. In other words, the used information can be very different ranging from the simple methods that use only function information to very sophisticated ones that use higher derivative information. The most used inclusion functions based on interval analysis follow.

1.3.1.1 Natural Extension

This inclusion function was first studied in detail by Ramon E. Moore [107]. It is also called naive interval extension, since it uses the natural extension of arithmetical operations and elementary functions as explained before. The natural inclusion function of an implemented function given as a computer program, is obtained by substituting the real variables for intervals and the real operations or standard functions for interval operations or the respective inclusion functions.

We denote the natural inclusion function for a function f as $\boldsymbol{f}_n(\boldsymbol{x})$ or simply $\boldsymbol{f}(\boldsymbol{x})$. It was shown (see for example, [125]) that the convergence order of the natural inclusion function is at least 1.

1.3.1.2 Centered Form

This inclusion function uses the gradient of the function, hence, we can only apply this method if the function is differentiable and we can compute an enclosure of the gradient. The gradient is usually easy to obtain by automatic differentiation [66]. One can easily get the centered form from the following deduction.

Lagrange's mean-value theorem says that there exists a $\xi \in [x, c]$ such that

$$f(x) = f(c) + f'(\xi)(x - c),$$

where f' denotes the gradient of f (understood as a row vector). Let $\boldsymbol{x}$ be a box, such that $x, c \in \boldsymbol{x}$. If an inclusion function of the gradient, $\boldsymbol{g}(\boldsymbol{x})$, is known, then

$$f(x) \subseteq \boldsymbol{f}(c) + \boldsymbol{g}(\boldsymbol{x})(x - c), \quad \forall x \in \boldsymbol{x} \tag{1.4}$$

and the inclusion

$$f(\boldsymbol{x}) \subseteq \boldsymbol{f}(c) + \boldsymbol{g}(\boldsymbol{x})(\boldsymbol{x} - c)$$

holds. Thus, the centered form is defined as

$$\boldsymbol{f}_c(\boldsymbol{x}) = \boldsymbol{f}(c) + \boldsymbol{g}(\boldsymbol{x})(\boldsymbol{x} - c),$$

where $c = \mathrm{m}(\boldsymbol{x})$. Although we often use the midpoint for the parameter c, it is also reasonable to choose it elsewhere inside $\boldsymbol{x}$. In case the convergence order of the inclusion function $\boldsymbol{g}(\boldsymbol{x})$ of the gradient is at least one, then the convergence order of the centered form is at least quadratic [70, 93].

1.3.1.3 Baumann's Optimal Centered Form

Baumann introduced the optimal c for the centered form in [4]. He proved that $\underline{f_c}(\boldsymbol{x}) \leq \underline{f_{b^-}}(\boldsymbol{x}), \forall c \in \boldsymbol{x}$ for an optimal $b^- \in \boldsymbol{x}$. The optimal lower bound is attained at $c = b^-$, which can be obtained in the one dimensional case as

$$b^- = \frac{u\underline{x} - l\overline{x}}{u - l},$$

where $\boldsymbol{x} = [\underline{x}, \overline{x}]$ and $\boldsymbol{g}(\boldsymbol{x}) = [l, u] \ni 0$. In case $\boldsymbol{g}(\boldsymbol{x}) > 0$, the optimal center is $b^- = \overline{x}$, and when $\boldsymbol{g}(\boldsymbol{x}) < 0$ it is $b^- = \underline{x}$. If the optimal lower bound is attained at b^-, the optimal upper bound attained at b^+ can be obtained by just reflecting b on the midpoint of the interval $\boldsymbol{x}$. The best inclusion can be obtained using both b^+ and b^-, although it needs more function evaluations. We can see the difference between the centered form and Baumann's optimal centered form in Figure 1.2.

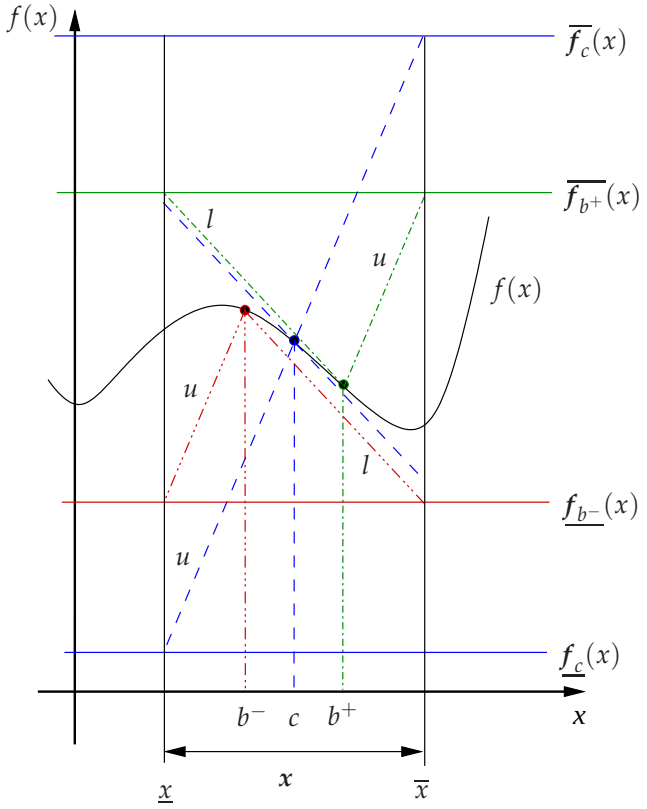


Figure 1.2: The centered form and Baumann's optimal centered form.

In case the condition $0 \in \boldsymbol{g}(\boldsymbol{x})$ does not hold, i.e. the inclusion of the derivative does not contain zero, then the respective function f is monotone in the interval $\boldsymbol{x}$. For monotone functions, a very good inclusion can be given by evaluating any kind of inclusion function at the end points of the interval.

In higher dimensional functions the above formula must be used componentwise. Although Baumann's optimal centered form always gives at least as good an enclosure as the general centered form, its convergence order is still only quadratic, as for the centered form.

1.3.1.4 Slope Form

Instead of the inclusion of the first derivative, the so called slope is usually better for building an enclosure of the underlying function [126]. The slope itself is a lower and upper bounding line going through a given basis point within the given interval (see Figure 1.3). The slope can be determined in a similar

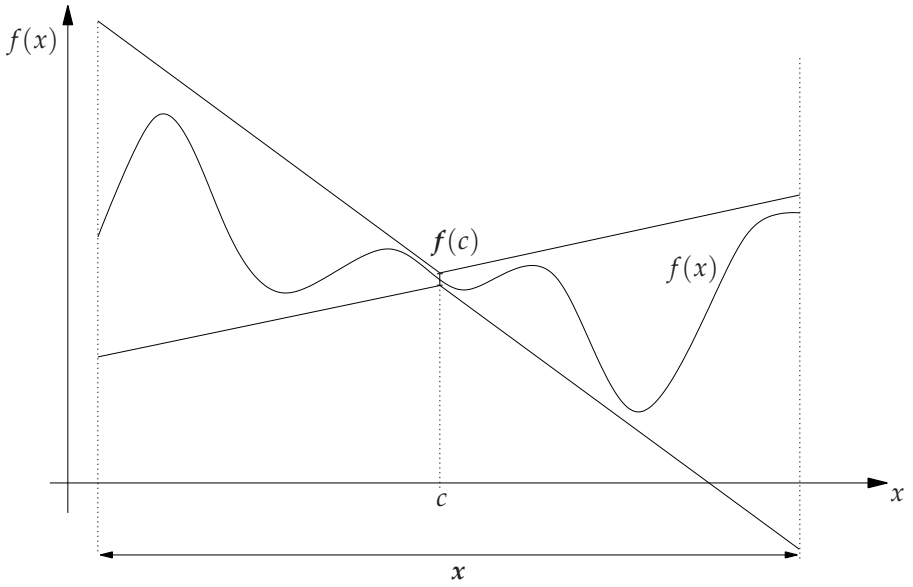


Figure 1.3: The slope form.

way as it is done in automatic differentiation, by operator overloading. The slopes can be evaluated in two modes (similarly as in automatic differentiation): in forward and reverse modes. In forward mode the rules of the slope are evaluated from bottom to top in the computational graph, while in reverse mode from top to bottom.

1.3.1.5 Affine Form

The affine form was introduced by Stolfi and Figueiredo [17, 22, 23], and later extended by Messine [102]. In the affine form $\boldsymbol{f}_a(\boldsymbol{x})$, a variable x is represented by a first-degree polynomial

$$\hat{x} = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \ldots + x_n\varepsilon_n,$$

where the coefficients $x_i (i = 1, \ldots, n)$ are finite floating-point numbers, and $\varepsilon_i (i = 1, \ldots, n)$ are symbolic real variables whose values are unknown but lie in the interval [-1,1]. We call x_0 the *central value* of the affine form $\hat{x}$, the coefficients x_i are its *partial deviations*, and the ε_i are the *noise symbols*.

Each noise symbol ε_i stands for an independent component of the total uncertainty of the ideal quantity x, while the corresponding coefficient x_i gives the magnitude of that component. The affine operations are straightforward in this form:

$$\begin{aligned} \hat{x} \pm \hat{y} &= (x_0 \pm y_0) + (x_1 \pm y_1)\varepsilon_1 + \ldots + (x_n \pm y_n)\varepsilon_n \\ \alpha\hat{x} &= (\alpha x_0) + (\alpha x_1)\varepsilon_1 + \ldots + (\alpha x_n)\varepsilon_n \\ \hat{x} \pm \gamma &= (x_0 \pm \gamma) + x_1\varepsilon_1 + \ldots + x_n\varepsilon_n \end{aligned}$$

For non-affine operations an affine approximation is used, and the possible error of the approximation is expressed in an extra term.

As an example, suppose that the quantities x and y are represented by the affine forms $\hat{x} = 20 - 3\varepsilon_1 + \varepsilon_3 + 4\varepsilon_4$ and $\hat{y} = 10 + 2\varepsilon_1 + \varepsilon_2 - \varepsilon_4$. From this data we know that x lies in the interval $\boldsymbol{x} = [12, 28]$ and y lies in $\boldsymbol{y} = [6, 14]$, i.e. the pair (x, y) lies in the light-blue rectangle of Figure 1.4. However since the two affine forms include common noise variables, they are not fully independent, so the pair $(\hat{x}, \hat{y})$ must lie in the dark red region of Figure 1.4.

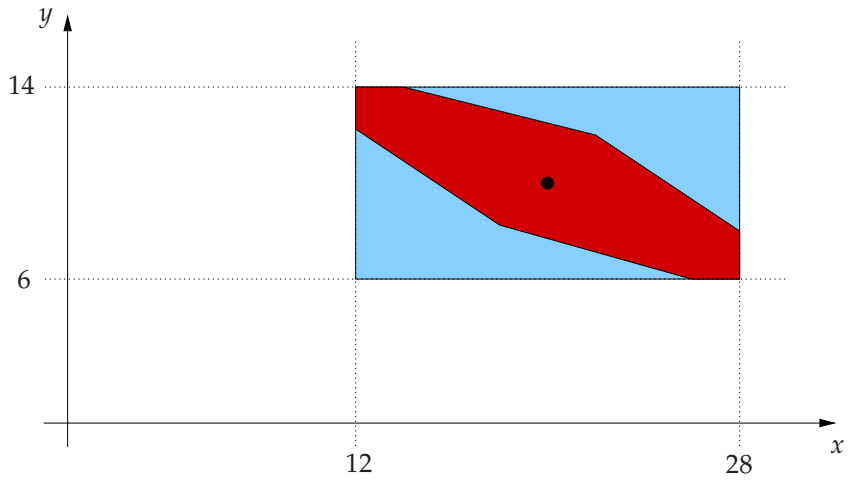


Figure 1.4: The joint range of the partially dependent affine forms $\hat{x}$ and $\hat{y}$.

All the above inclusion functions can be used to calculate bounds in a Branch and Bound method. When the bounds are obtained by an inclusion function based on interval analysis, the method is called Interval Branch and Bound method, whose general form is described in the next section.

1.4 Interval Branch and Bound Method

Interval Branch and Bound methods use boxes to define the search region and its branched subproblems, and some inclusion functions to bound the objective function over a given box. A general algorithmic description of an Interval Branch and Bound algorithm for solving (1.1) is shown in Algorithm 1.1.

Algorithm 1.1 A general interval B&B algorithm.

Funct IBB($\boldsymbol{s}$, f)

1: Set the working list $\mathcal{L}_W := \{\boldsymbol{s}\}$ and the final list $\mathcal{L}_S := \emptyset$
2: **while** ($\mathcal{L}_W \neq \emptyset$) **do**
3: Select a box $\boldsymbol{x}$ from $\mathcal{L}_W$ — *Selection rule*
4: Compute $\boldsymbol{f}(\boldsymbol{x})$ — *Bounding rule*
5: **if** $\boldsymbol{x}$ cannot be eliminated — *Elimination rule*
6: Divide $\boldsymbol{x}$ into $\boldsymbol{x}^j$, $j = 1, \ldots, p$, subintervals — *Division rule*
7: **for** $j = 1, \ldots, p$ **do**
8: **if** $\boldsymbol{x}^j$ satisfies the termination criterion — *Termination rule*
9: Store $\boldsymbol{x}^j$ in $\mathcal{L}_S$
10: **else**
11: Store $\boldsymbol{x}^j$ in $\mathcal{L}_W$
12: **return** $\mathcal{L}_S$

Of course, every concrete realization of Algorithm 1.1 depends on the available information about the objective function $f(x)$ and the feasible set S (now it is supposed that the feasible set is only bound constrained, i.e. $\boldsymbol{s} = S$). It is always assumed that f is continuous, thus, some inclusion functions can be evaluated for $f(x)$ (although it is possible to obtain inclusion functions for non-continuous functions as well [118]). In general, the objective function is also differentiable, and inclusion for its gradient vector $f'(x)$ can be computed (denoted by $\boldsymbol{g}(\boldsymbol{x})$). When the function f is two times differentiable, one can even compute an inclusion for its Hessian matrix H, denoted by $\boldsymbol{H}(\boldsymbol{x})$.

When the above information is available, the rules of a traditional realization of Algorithm 1.1 can be written more precisely. Some common rules which are frequently used are described below.

Selection rule: Among all the boxes $\boldsymbol{x}^j$ stored in the working list $\mathcal{L}_W$, select a box $\boldsymbol{x}$ such that $\underline{\boldsymbol{f}}(\boldsymbol{x}) = \min\{\underline{\boldsymbol{f}}(\boldsymbol{x}^j) \mid \boldsymbol{x}^j \in \mathcal{L}_W\}$.

Bounding rule: Some inclusion functions $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_k$ of the objective function f are available. The interval $f(\boldsymbol{x})$ is bounded by the interval $\boldsymbol{f}(\boldsymbol{x}) = \cap_{i=1}^{k} \boldsymbol{f}_i(\boldsymbol{x})$.

Elimination rules: Common discarding tests are the following:

Midpoint test: An interval $\boldsymbol{x}$ is rejected when $\underline{\boldsymbol{f}}(\boldsymbol{x}) > \tilde{f}$, where $\tilde{f}$ is the best known upper bound of the global minimum f^*. The value of $\tilde{f}$ is usually updated by $\overline{\boldsymbol{f}}(\mathrm{m}(\boldsymbol{x}))$.

Cut-off test: When $\tilde{f}$ is improved, all intervals $\boldsymbol{x}$ stored in the working and final lists satisfying the condition $\underline{\boldsymbol{f}}(\boldsymbol{x}) > \tilde{f}$ are rejected.

Monotonicity test (for differentiable functions): If for an interval $\boldsymbol{x}$ an inclusion function of the gradient, $\boldsymbol{g}(\boldsymbol{x})$ is given, and the condition $0 \notin \boldsymbol{g}(\boldsymbol{x})$ is fulfilled, then this means that the objective function is monotonous over the interval $\boldsymbol{x}$, thus it does not contain any stationary point. If $\boldsymbol{x}$ is not on the boundary of the search region, it can be rejected, otherwise it can be narrowed to the intersection of $\boldsymbol{x}$ and the boundary of the search region.

Non-convexity test (for twice differentiable functions): If for an interval $\boldsymbol{x}$ an inclusion function of the Hessian matrix $\boldsymbol{H}(\boldsymbol{x})$ is given, and $\boldsymbol{H}_{ii}(\boldsymbol{x}) < 0$ for some i, then the non-convexity of the objective function over $\boldsymbol{x}$ is guaranteed. If $\boldsymbol{x}$ is not on the boundary of the search region, it can be rejected, otherwise it can be narrowed to the intersection of $\boldsymbol{x}$ and the boundary of the search region.

Newton step (for twice differentiable functions): If for an interval $\boldsymbol{x}$ an inclusion function of the Hessian matrix, $\boldsymbol{H}(\boldsymbol{x})$, and an inclusion of the gradient, $\boldsymbol{g}(\boldsymbol{x})$, are given, one can find all the zeros of the gradient by the interval Newton method, that is, by iterating

$$\begin{aligned} N(\boldsymbol{x}^{(k+1)}) &= \mathrm{m}(\boldsymbol{x}^{(k)}) - [\boldsymbol{H}(\boldsymbol{x}^{(k)})]^{-1}\boldsymbol{g}(\mathrm{m}(\boldsymbol{x}^{(k)})) \\ \boldsymbol{x}^{(k+1)} &= N(\boldsymbol{x}^{(k+1)}) \cap \boldsymbol{x}^{(k)} \end{aligned}$$

until $\boldsymbol{x}^{(k+1)} = \emptyset$ or $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)}$ for some k. In the first case it is proved that there is no global optimizer in the box $\boldsymbol{x}$, while in the later case all local optimizers of $\boldsymbol{x}$ are included in $\boldsymbol{x}^{(k+1)}$ if any. Moreover, if $N(\boldsymbol{x}^{(k+1)}) \subset \boldsymbol{x}^{(k)}$ for any k, then it is proved that there exists a unique local optimizer in the interval $\boldsymbol{x}^{(k+1)}$. This procedure is rather costly, and even if it converges to a local optimum, there is no guarantee that it will not be deleted in the next steps by the cut-off test, therefore, it is not efficient in this form. That is why it is recommended to perform only one step of this iteration.

Furthermore, the computation of $N(\boldsymbol{x}^{(k+1)})$ is not straightforward. If $\boldsymbol{H}(\boldsymbol{x}^{(k)})$ is not sparse, or tight enough, its inverse cannot be computed, or the result is so wide that it is useless. Therefore, it is better to solve the interval linear system of equations

$$\boldsymbol{H}(\boldsymbol{x}^{(k)})(N(\boldsymbol{x}^{(k+1)}) - \mathrm{m}(\boldsymbol{x}^{(k)})) = -\boldsymbol{g}(\mathrm{m}(\boldsymbol{x}^{(k)})) \tag{1.5}$$

using the Krawczyk method [92] or the interval Gauss-Seidel method, first described in [130]. Usually, in order to obtain tighter results one should use preconditioning. The inverse of the midpoint matrix $\mathrm{m}(\boldsymbol{H}(\boldsymbol{x}^{(k)}))$ is frequently used [72]. For more details on the diverse variations on solution methods for (1.5), see [73, 88, 110].

Division rule: Usually two subintervals are generated using $\mathrm{m}(\boldsymbol{x})$ as the subdivision point (bisection) on direction k, where k is the coordinate such that $\mathrm{w}(\boldsymbol{x}_k) = \max_{i=1,\ldots,n} \mathrm{w}(\boldsymbol{x}_i)$.

Termination rule: A parameter ε determines the desired accuracy of the problem solution. Therefore, a box $\boldsymbol{x}$ with $\mathrm{w}(\boldsymbol{x}) \leq \varepsilon$ and/or $\mathrm{w}(\boldsymbol{f}(\boldsymbol{x})) \leq \varepsilon$, is moved to the final list $\mathcal{L}_S$.

The above description of the interval Branch and Bound method and its elimination rules are valid for unconstrained or bound-constrained optimization problems (when in (1.1) S is a box). If the set S is defined by some constraints, i.e.

$$S = \{x \in \mathbb{R}^n \mid h_j(x) \leq 0, j = 1, \ldots, q, (h_j : \mathbb{R}^n \to \mathbb{R}, h_j \in C^0)\}, \tag{1.6}$$

a feasibility test has to be added as an elimination rule:

Feasibility test: We say that a box $\boldsymbol{y}$ *satisfies* the constraint $h_j(x) \leq 0$ if $\overline{\boldsymbol{h}}_j(\boldsymbol{x}) \leq 0$ and that $\boldsymbol{x}$ *does not satisfy* it if $\underline{\boldsymbol{h}}_j(\boldsymbol{x}) > 0$. A box $\boldsymbol{x}$ is said to be *feasible* if it satisfies all the constraints, *infeasible* if it does not satisfy at least one of the constraints, and *undetermined* otherwise. A box $\boldsymbol{x}$ is said to be *strictly feasible* if $\overline{\boldsymbol{h}}_j(\boldsymbol{x}) < 0$, $j = 1, \ldots, q$. The Feasibility test [125] discards boxes that are infeasible. It also provides information about the feasibility of a box. Notice that to apply this test we just need an inclusion function $\boldsymbol{h}_j$ for each of the functions h_j defining the constraints.

Additionally, in the above description of the eliminating rules the following changes have to be made when solving constrained problems. When updating $\tilde{f}$ by evaluating $\boldsymbol{f}(\mathrm{m}(\boldsymbol{x}))$, one has to ensure that $\mathrm{m}(\boldsymbol{x})$ is feasible. Also, the monotonicity test, the non-convexity test and the Newton step can only be applied to feasible boxes.

Remark 1.7. Later on we will refer to Algorithm 1.1 as *traditional algorithm*, when the following rules are used: the neural and centered form as inclusion functions, the midpoint test, the cut-off test and the monotonicity test as elimination rules, bisection of the widest coordinate in the division rule, selection by the minimal lower bound of the objective, and storing the boxes with $\mathrm{w}(\boldsymbol{x}) < \varepsilon$ as solution in the termination rule. When other rules are used they will be specified accordingly.

1.5 Obtaining the region of δ-optimality

Generally, in global optimization the aim of the methods is to find the global optimizer, or optimizers, which usually means a few points. However, in many real life problems not only the global optimizer(s) are of interest, but also *near optimal* solutions, i.e. points whose objective values are within a given percent of difference from the optimal value.

If we denote the optimal value of the minimization problem (1.1) by f^*, the region of δ-optimality of (1.1) is the set

$$R_\delta = \{x \in S \mid f(x) - f^* \leq \delta \cdot |f^*|\}.$$

A Branch and Bound method for obtaining the region of δ-optimality is presented in [120], which is called the Generalized Big Square Small Square method (GBSSS). The method consists of two phases. The first one entails the determination of the optimal objective value of (1.1) up to a prespecified relative precision ε. The second phase consists of the determination of R_δ, up to a prespecified precision η. The output of the algorithm is two lists of subsets, LIR_δ and LOR_δ. The union of the subsets in the first list, $IR_\delta = \cup_{Q \in LIR_\delta} Q$, intersected with S gives an *inner approximation* of R_δ (a subset of R_δ). The union of the subsets in both lists, $OR_\delta = \cup_{Q \in LIR_\delta \cup LOR_\delta} Q$, intersected with S forms an *outer approximation* of R_δ (a superset of R_δ), guaranteed to lie entirely within $R_{\delta+\eta(1+\delta)}$ (see Figure 1.5), i.e.,

$$IR_\delta \cap S \subseteq R_\delta \subseteq OR_\delta \cap S \subseteq R_{\delta+\eta(1+\delta)}.$$

Plastria's method [120] uses the Branch and Bound schema, but not an interval based bounding rule. The only discarding test used is the cut-off test in both phases. In Sections 3.2 and 3.3 we will use a method which computes the δ-optimal region based on the above method, although we will use inclusion functions to compute bounds, as well as some new discarding tests.

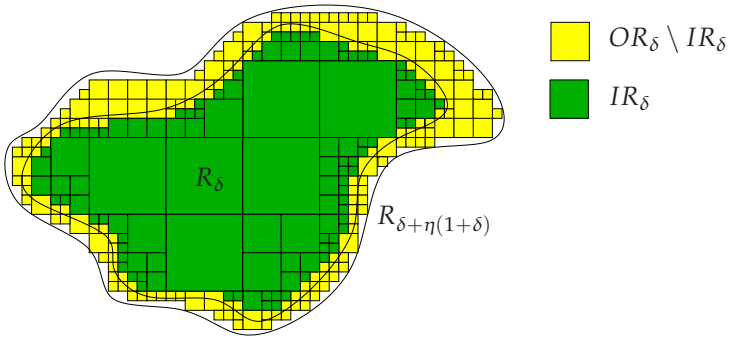


Figure 1.5: Inner and outer approximation of R_δ.

Chapter 2

Accelerating the Interval B&B Method

Finding the global optimizer(s) of a global optimization problem is NP-hard. Therefore, the interval Branch and Bound method in Section 1.4 is limited to low dimensional problems, or to *easy to solve problems* (no huge dependency problem, no cluster effect, etc.). In order to solve higher dimensional or difficult highly nonlinear problems, we need to use good accelerating devices and the best inclusion function.

This chapter is dedicated to all of the accelerating devices developed, or modified by us for the interval B&B method. In the first section we discuss how the best inclusion function for a given problem could be chosen [142, 143]. In Section 2.2 and 2.3 we describe new pruning methods to eliminate regions which do not contain any global optimizer point by the efficient use of gradient information [98, 99, 141]. Finally, in Section 2.4 modifications of existing elimination rules are presented [51].

2.1 Choosing the best inclusion function

An inclusion function generally overestimates the range of a function. The extent of the overestimation depends on the type of the inclusion function, on the considered function and on the width of the interval.

Usually, a surrogate for this problem is to find the inclusion function providing the tightest inclusions, since in this way, the different accelerating devices used in interval B&B methods are more efficient at discarding boxes, thus the problem is solved faster. The classical measure for the quality of an inclusion function is the convergence order (see Definition 1.6). In general, an inclusion function is better if it has larger convergence order, although it might also be possible that an inclusion function with larger α gives bigger overestimations for some intervals due to the constant c.

The comparison of inclusion functions is usually done by comparing their convergence orders, so that the inclusion function with higher convergence order is deemed to be better. However, the convergence order only provides a lower bound for the speed, i.e. it shows the worst case. For practical purposes one would like to know the average behavior, especially on frequently used interval sequences. Therefore, we have conducted an extensive empirical study measuring the average convergence speed of different inclusion functions so that we can decide which inclusion function should be used in a given situation.

2.1.1 Theoretical results

Let us consider that $\boldsymbol{f}_i, i = 1, \ldots, m$, are different inclusion functions of the continuously differentiable function $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$. Let us define a new inclusion function $\boldsymbol{f}$, as the intersection of $\boldsymbol{f}_i$ for all $i = 1, \ldots, m$, i.e. for every $\boldsymbol{x} \subseteq D$

$$\boldsymbol{f}(\boldsymbol{x}) = \bigcap_{i=1,\ldots,n} \boldsymbol{f}_i(\boldsymbol{x}).$$

Inclusion isotonicity (see Definition 1.2) has an important role in verified numerical methods, especially in global optimization techniques. In this context the following statements hold.

Proposition 2.1. *Assume that $\boldsymbol{f}_i$, $i = 1, \ldots, m$ are inclusion isotone. Then the inclusion function $\boldsymbol{f} = \bigcap_{i=1,\ldots,m} \boldsymbol{f}_i(\boldsymbol{x})$ is also inclusion isotone.*

Proof. We prove the statement indirectly. Assume that there exist intervals $\boldsymbol{x} \subseteq \boldsymbol{z} (\subseteq D)$ such that $\boldsymbol{f}(\boldsymbol{x}) \not\subseteq \boldsymbol{f}(\boldsymbol{z})$. Due to the assumption of the proposition, the inclusions $\boldsymbol{f}_i(\boldsymbol{x}) \subseteq \boldsymbol{f}_i(\boldsymbol{z})$, $i = 1, \ldots, m$ hold. Let us suppose that $\underline{\boldsymbol{f}}(\boldsymbol{x}) < \underline{\boldsymbol{f}}(\boldsymbol{z})$, i.e.

$$\max_{i=1,\ldots,m} \underline{\boldsymbol{f}_i}(\boldsymbol{x}) < \max_{i=1,\ldots,m} \underline{\boldsymbol{f}_i}(\boldsymbol{z}). \tag{2.1}$$

By using $\underline{\boldsymbol{f}_i}(\boldsymbol{x}) \geq \underline{\boldsymbol{f}_i}(\boldsymbol{z})$, $i = 1, \ldots, m$ we obtain that

$$\max_{i=1,\ldots,m} \underline{\boldsymbol{f}_i}(\boldsymbol{x}) \geq \max_{i=1,\ldots,m} \underline{\boldsymbol{f}_i}(\boldsymbol{z}),$$

and that contradicts (2.1). In the case $\overline{\boldsymbol{f}}(\boldsymbol{x}) > \overline{\boldsymbol{f}}(\boldsymbol{z})$, similar arguments hold. □

Definition 2.2. *An inclusion function $\boldsymbol{f}$ is called* inclusion isotone from below (above)*, if $\underline{\boldsymbol{f}}(\boldsymbol{x}) \leq \underline{\boldsymbol{f}}(\boldsymbol{y})$ ($\overline{\boldsymbol{f}}(\boldsymbol{x}) \geq \overline{\boldsymbol{f}}(\boldsymbol{y})$) holds for every $\boldsymbol{x} \subseteq \boldsymbol{y} \subseteq D$.*

Proposition 2.3. *Assume that for every $\boldsymbol{x} \subseteq D$ there exists an index $i \in \{i_1, \ldots, i_l\} \subseteq \{1, \ldots, m\}$ and $\exists\, k \in \{k_1, \ldots, k_r\} \subseteq \{1, \ldots, m\}$ such that $\underline{\boldsymbol{f}_i}(\boldsymbol{x}) \geq \underline{\boldsymbol{f}_j}(\boldsymbol{x})$ and $\overline{\boldsymbol{f}_k}(\boldsymbol{x}) \leq \overline{\boldsymbol{f}_j}(\boldsymbol{x})$ for $j = 1, \ldots, m$. Suppose also that for all $i \in \{i_1, \ldots, i_l\}$ and $k \in \{k_1, \ldots, k_r\}$ the inclusion functions $\boldsymbol{f}_i$ and $\boldsymbol{f}_k$ are inclusion isotone from below and from above, respectively. Then the inclusion function $\boldsymbol{f} = \bigcap_{i=1,\ldots,m} \boldsymbol{f}_i(\boldsymbol{x})$ is inclusion isotone.*

Proof. Due to the conditions $\boldsymbol{f}(\boldsymbol{x}) = [\underline{\boldsymbol{f}_i}(\boldsymbol{x}), \overline{\boldsymbol{f}_k}(\boldsymbol{x})]$, $i \in \{i_1, \ldots, i_l\}$ and $k \in \{k_1, \ldots, k_r\}$ for every $\boldsymbol{x} \subseteq D$ and thus, $\boldsymbol{f}$ combines the one-sided inclusion isotonicity of $\boldsymbol{f}_i$-s and $\boldsymbol{f}_k$-s. □

In interval Branch and Bound optimization methods the inclusion isotonicity can be reached easily when for all the descents $\boldsymbol{x}^k$ of $\boldsymbol{x}$ the inclusion function $\boldsymbol{f}(\boldsymbol{x}^k)$ is intersected with $\boldsymbol{f}(\boldsymbol{x})$. It would be interesting to see how much the improvement of this easy trick can be for a non-isotone inclusion function, e.g. for the general centered form.

Proposition 2.4. *Suppose that the convergence orders of $\boldsymbol{f}_i$ are α_i, $i = 1, \ldots, m$, respectively. The convergence order of the inclusion function $\boldsymbol{f} = \bigcap_{i=1,\ldots,m} \boldsymbol{f}_i(\boldsymbol{x})$ is at least $\max_{i=1,\ldots,m} \alpha_i$.*

Proof. By the construction of $\boldsymbol{f}$ it is trivial that $\boldsymbol{f}(\boldsymbol{x}) \subseteq \boldsymbol{f}_i(\boldsymbol{x})$, $\forall \boldsymbol{x} \subseteq \boldsymbol{y}$, $i = 1, \ldots, m$, therefore, $\mathrm{w}(\boldsymbol{f}(\boldsymbol{x})) \leq \mathrm{w}(\boldsymbol{f}_i(\boldsymbol{x}))$, $i = 1, \ldots, m$ and

$$\mathrm{w}(\boldsymbol{f}(\boldsymbol{x})) - \mathrm{w}(f(\boldsymbol{x})) \leq \mathrm{w}(\boldsymbol{f}_i(\boldsymbol{x})) - \mathrm{w}(f(\boldsymbol{x})) \leq c_i \mathrm{w}(\boldsymbol{x})^{\alpha_i},$$

for all $i = 1, \ldots, m$ (where c_i is the respective constant valid for the convergence speed of the inclusion function $\boldsymbol{f}_i$), and that proves our statement. □

Now let us consider the zero convergence property for the intersection of inclusion functions.

Corollary 2.5. *For the zero-convergence of $\boldsymbol{f}$ it is sufficient to have at least one $i \in \{1, \ldots, m\}$ such that $\boldsymbol{f}_i$ is α-convergent (with $\alpha_i > 0$).*

2.1.2 The empirical convergence speed

The convergence order of an inclusion function can usually be determined theoretically. It gives a lower bound for the overestimation, due to the worst case definition. Our goal in this section is different. We want to determine an approximate convergence speed which gives a good indication of which inclusion function should be used (in a given situation) in order to have the best inclusion at an acceptable cost.

The empirical convergence speed is obtained by determining the constants α and c approximating the equation

$$\mathrm{w}(\boldsymbol{f}(\boldsymbol{x})) - \mathrm{w}(f(\boldsymbol{x})) = c \cdot \mathrm{w}(\boldsymbol{x})^{\alpha} \tag{2.2}$$

for a finite set of inclusions with the smallest error. This fits the definition of the theoretical convergence order (1.3), except that the inequality is changed to equality in order to obtain information about the average behavior on a finite set of inclusions. By limiting the examined inclusions to a set of inclusions a lot of information is lost. However, if we are interested in the convergence speed of inclusion functions in global optimization, and the set of inclusions extracted by a global optimization method is examined, the result is satisfactory.

Reformulating equality (2.2) to

$$\lg\Big(\mathrm{w}\big(\boldsymbol{f}(\boldsymbol{x})\big) - \mathrm{w}\big(f(\boldsymbol{x})\big)\Big) = \lg(c) + \alpha \lg\Big(\mathrm{w}(\boldsymbol{x})\Big),$$

we can easily determine the constants α and c with linear regression. This realigning is possible because the overestimation can never be negative. Clearly, the estimation of the parameters with this new equality is not equivalent to the estimation with equality (2.2), but as we will see from the results, it is still adequate. Obviously, it is not the only way to extract suitable information from the actual overestimation figures, but the description parameters α and c are quite useful when estimating the actual convergence speed.

2.1.3 The examined sets of boxes

One of the main questions for the empirical convergence speed is how to choose the set of boxes that are applied in the computation. In our study the following sets were examined:

- a sequence of embedded boxes containing a given global minimizer point (Opt),
- several random sequences of embedded boxes converging to non-stationary points (Rand),
- a sequence of boxes increasing in their width starting from the given search interval (Up),
- boxes generated by Algorithm 1.1 (B&B).

The set obtained by Algorithm 1.1 is the most interesting one. The processed boxes depend on the used inclusion functions, accelerating tools, branching and selecting rules. In particular, as inclusion functions the natural interval extension and centered form were used together with the monotonicity and cut-off tests as accelerating tools. The bisection by the widest interval component was used for branching, and we select the subinterval with the smallest lower bound on the objective function for the next iteration.

For every box of the sequences we compute all the studied inclusion functions, the interval width, and an approximation of the range. The hardest part of the computations is the range approximation, because it means that we have to approximate the solution of an NP-hard problem for every box. One possible way of doing it is to use the interval B&B algorithm itself to approximate the lower bound by minimizing the function, and to obtain the upper bound by maximizing it. For the optimization we used all the available accelerating tools: the cut-off test, the monotonicity test, the non-convexity test, and the Newton-step. The applied stopping criterion required the relative width of the inclusion of the minimum (or maximum) to be small enough, i.e. the maximal relative error of the approximation must be below a small positive value (in this case it was set to 10^{-8}).

2.1.4 The examined inclusion function types

In this study, we examined only the most widely used inclusion functions, which are available in many interval libraries and described in Section 1.3.1. These include namely

- Natural interval extension, denoted by $\boldsymbol{f}_n(\boldsymbol{x})$.
- Centered form (or mean value form), denoted by $\boldsymbol{f}_c(\boldsymbol{x})$.
- Baumann's optimal centered form, denoted by $\boldsymbol{f}_b(\boldsymbol{x})$.
- Forward and Reverse Slope forms, denoted by $\boldsymbol{f}_{fs}(\boldsymbol{x})$ and $\boldsymbol{f}_{rs}(\boldsymbol{x})$, respectively.

2.1.5 Numerical results

The most interesting results of our experiments are shown in the following figure and tables. We have used the C-XSC-2.0 interval library [82] with the CToolbox-2.0 software that contained the optimization code [69]. A graphical representation of the results for the Goldstein-Price function [147] can be seen in Figure 2.1. The six graphs should be interpreted as follows.

The first five graphs belong to the five examined inclusion functions, while the last one shows a summary. In every graph the horizontal and vertical axes show the logarithm of the width of the box and the logarithm of the overestimation, respectively. The small black circles denote the overestimation of the inclusion function for the examined intervals. Sometimes the inclusion functions give the exact range — at least according to our estimated range. In these cases the logarithm of the overestimation is not defined. It should be $-\infty$ ($\lim_{x\to 0} \lg(x) = -\infty$), but for a reasonable result we adjust the logarithm of the overestimation to -30 (it reflects in a way the limitation of computer number representation). Every graph provides a line, which we have obtained by linear regression. From the legend one can read the approximate convergence speed α and the order of magnitude of the constant c. On the upper part of every graph we denote, with small triangles, when the given inclusion function gave the best lower ($\triangle$) or upper bounds ($\triangledown$).

Note that in the case of centered and Baumann inclusion functions for small intervals (with diameter around 10^{-10}) the overestimation is about the same. That is why the black circles form an almost constant line there. This happens because we reach the level of rounding error, which means that the inclusion differs from the range approximation only by rounding error.

Evaluating the results on Figure 2.1, we can conclude that, as the last graph indicates, the natural interval extension is the best choice if our interval is relatively large, with a width 0.1 or wider. Otherwise the forward or reverse mode slope arithmetics [7, 126] provides the best inclusions. These findings reflect only the case when the subintervals are produced by a Branch and Bound algorithm searching for the global minimum — i.e. for other interval sequences different conclusions can be drawn.

On the other hand, the figure nicely shows that the natural interval extension provides a close to linear convergence, according to our empirical convergence study.

The more sophisticated techniques which apply the inclusion of the first derivative achieve substantially better convergence speeds: the slope arithmetics around $\alpha = 3.9$, and the centered forms about $\alpha = 3.4$. The precise inclusion of the latter methods is due to the fact that for small intervals the function may be monotonous, and thus its inclusion may be precise (up to the machine representation). Notice also that the linear least squares fit seems to be an informative description.

We summarize our results in detail in Table 2.1 and Table A.1 (see Appendix A). In Table 2.1 beside the values of α one can see (in brackets) the correlation coefficients that describe the strength of linear relationships (it is the square root of the variation ratio). In most cases the correlation coefficients indicate strong linear dependency. For most of the functions we have five lines, one for each examined set of boxes. These are the boxes extracted by Algorithm 1.1 (denoted by B&B), an embedded sequence converging directly to a minimizer point (Opt), embedded interval sequences converging to random non-stationary points (Rand), an upward going sequence (Up), and the union of the above series (All), respectively.

We have tested standard global optimization test functions together with some others that are usually applied for investigating the efficiency of interval optimization methods. These were Branin,

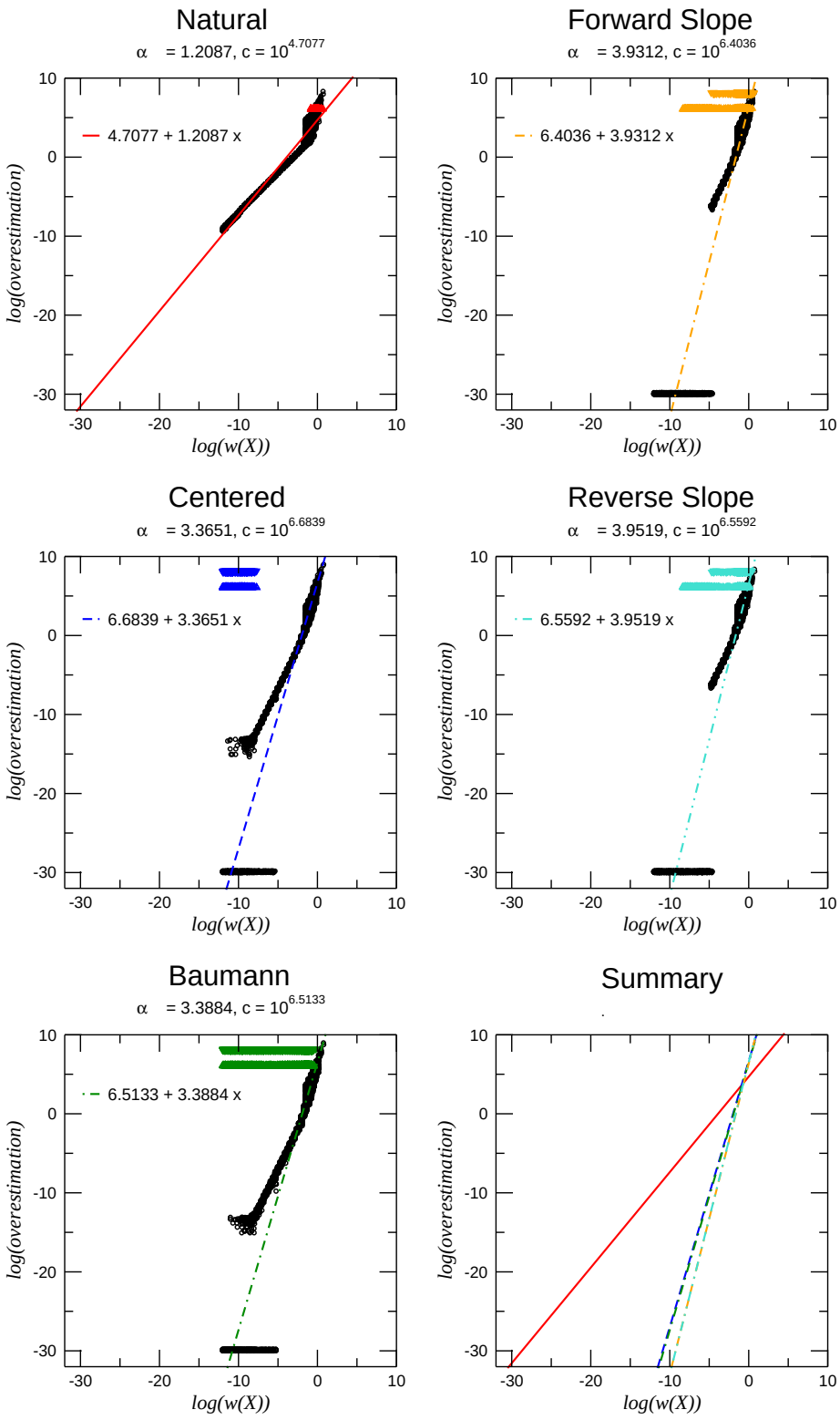


Figure 2.1: Empirical convergence speed results on the Goldstein-Price function for boxes extracted by the interval B&B algorithm.

Goldstein-Price, Griewank 5, Levy 3, 5, 8, 9, 10, 13, 14, 15, 16, Ratz 4, Rosenbrock, Shekel 5, Schwefel 21, 25, 31, 32, 37, 218, Six-Hump-Camel-Back, and Three-Hump-Camel-Back [24, 18, 128, 147, 150]. In some cases the very intensive computational requirements could not be met by standard setting. Therefore the results are missing for those interval sequences, as is the case for the Griewank-5 test function. The interesting results are **highlighted** in Table 2.1.

The empirical convergence speed of the natural extension is sometimes the best among all the inclusion functions considered and much better than the theoretical convergence order (see for instance the B&B sequence of Branin or Levy 14 (L14), or the B&B and Opt sequences of Levy 13 (L13)).

Table 2.1: The empirical convergence speed α with the correlation coefficient in brackets for the test functions with different sets of boxes.

Problem	Set	α				
name	name	Natural	Centered	Baumann	For. Slope	Rev. Slope
Branin	B&B	**2.766** (0.85)	1.545 (0.96)	1.521 (0.90)	2.782 (0.84)	2.811 (0.84)
	Opt	0.893 (0.41)	1.744 (0.89)	3.034 (0.95)	3.115 (0.88)	3.105 (0.88)
	Rand	0.943 (0.41)	2.298 (0.87)	1.144 (0.49)	2.953 (0.84)	2.964 (0.84)
	Up	0.796 (0.11)	3.850 (1.00)	3.872 (1.00)	3.793 (1.00)	3.856 (1.00)
	All	2.360 (0.73)	1.748 (0.90)	1.348 (0.53)	2.912 (0.86)	2.932 (0.86)
GP	B&B	1.209 (0.98)	3.365 (0.95)	3.388 (0.95)	3.931 (0.95)	3.952 (0.95)
	Opt	1.166 (0.98)	3.323 (0.96)	3.290 (0.96)	3.635 (0.89)	3.613 (0.89)
	Rand	1.064 (0.97)	2.427 (0.85)	1.974 (0.55)	3.576 (0.86)	3.610 (0.86)
	Up	**7.812** (0.97)	7.067 (0.99)	**7.833** (0.99)	**7.207** (0.97)	**7.590** (0.98)
	All	1.193 (0.98)	3.267 (0.94)	3.372 (0.93)	3.900 (0.94)	3.919 (0.94)
Griew5	B&B	1.661 (0.65)	1.916 (1.00)	1.790 (0.65)	1.876 (1.00)	1.876 (1.00)
L3	B&B	0.967 (1.00)	3.070 (0.96)	3.073 (0.96)	3.495 (0.92)	3.494 (0.92)
	Opt	0.928 (0.99)	2.098 (0.68)	3.373 (0.91)	3.231 (0.87)	3.241 (0.87)
	Rand	1.068 (0.47)	2.127 (0.90)	2.286 (0.70)	2.902 (0.83)	2.918 (0.83)
	Up	**0.000** (0.76)	0.908 (0.99)	0.941 (0.98)	0.982 (0.98)	0.985 (0.99)
	All	1.042 (0.75)	2.761 (0.93)	3.133 (0.89)	3.281 (0.90)	3.284 (0.90)
L5	B&B	0.947 (0.99)	2.912 (0.97)	2.918 (0.97)	3.501 (0.94)	3.498 (0.94)
	Opt	0.939 (1.00)	2.591 (0.86)	3.118 (0.95)	3.240 (0.87)	3.249 (0.87)
	Rand	0.991 (1.00)	2.168 (0.89)	2.134 (0.67)	3.083 (0.86)	3.106 (0.87)
	Up	0.010 (0.21)	1.561 (0.99)	1.480 (0.99)	1.525 (0.99)	1.513 (0.99)
	All	0.948 (0.99)	2.286 (0.92)	3.007 (0.82)	3.085 (0.90)	3.081 (0.91)
L8	B&B	1.295 (0.53)	2.062 (1.00)	2.073 (1.00)	2.047 (1.00)	2.046 (1.00)
	Opt	1.019 (0.48)	2.050 (1.00)	2.060 (1.00)	2.034 (1.00)	2.039 (1.00)
	Rand	0.990 (0.99)	2.012 (0.87)	1.441 (0.56)	2.911 (0.87)	2.911 (0.87)
	Up	6.149 (0.24)	2.808 (0.98)	2.934 (0.99)	2.812 (0.98)	2.603 (0.97)
	All	0.715 (0.25)	2.037 (0.91)	1.959 (0.64)	2.620 (0.89)	2.619 (0.89)
L9	B&B	1.325 (0.54)	2.072 (1.00)	2.083 (1.00)	2.055 (1.00)	2.055 (1.00)
	Opt	1.018 (0.48)	2.064 (1.00)	2.075 (1.00)	2.037 (1.00)	2.039 (1.00)
	Rand	1.022 (0.47)	2.139 (0.86)	1.778 (0.63)	2.849 (0.84)	2.852 (0.85)
	All	1.040 (0.33)	2.109 (0.90)	1.951 (0.63)	2.550 (0.86)	2.551 (0.86)
L10	B&B	1.343 (0.54)	2.085 (1.00)	2.091 (1.00)	2.060 (1.00)	2.057 (1.00)
	Opt	1.032 (0.48)	2.070 (1.00)	2.082 (1.00)	2.038 (1.00)	2.047 (1.00)
	Rand	1.080 (0.88)	2.056 (0.86)	1.851 (0.64)	2.912 (0.85)	2.907 (0.85)
	All	1.076 (0.34)	2.062 (0.91)	2.015 (0.65)	2.553 (0.87)	2.550 (0.87)
L11	B&B	1.377 (0.54)	2.090 (1.00)	2.098 (1.00)	2.063 (1.00)	2.066 (1.00)
L12	B&B	1.383 (0.54)	2.090 (1.00)	2.102 (1.00)	2.071 (1.00)	2.068 (1.00)

Table 2.1: *(continued)* The empirical convergence speed α with the correlation coefficient in brackets for the test functions with different sets of boxes.

Problem	Set	α				
name	name	Natural	Centered	Baumann	For. Slope	Rev. Slope
L13	B&B	5.561 (0.82)	2.030 (1.00)	2.072 (1.00)	1.977 (1.00)	1.980 (1.00)
	Opt	4.893 (0.78)	1.994 (1.00)	2.051 (1.00)	1.954 (1.00)	1.955 (1.00)
	Rand	0.857 (0.96)	1.873 (0.99)	5.819 (0.83)	3.544 (0.75)	3.546 (0.75)
	All	1.724 (0.34)	1.885 (0.98)	5.058 (0.74)	3.138 (0.75)	3.140 (0.75)
L14	B&B	**2.191** (0.70)	2.028 (1.00)	2.043 (1.00)	2.002 (1.00)	2.008 (1.00)
	Opt	1.809 (0.65)	2.037 (1.00)	2.040 (1.00)	1.998 (1.00)	1.999 (1.00)
	Rand	0.944 (0.99)	2.139 (0.87)	2.331 (0.71)	3.128 (0.88)	3.130 (0.87)
	Up	**12.570** (0.63)	3.010 (0.99)	3.046 (0.99)	3.196 (0.98)	3.206 (0.98)
	All	0.826 (0.28)	2.097 (0.91)	2.468 (0.74)	2.737 (0.88)	2.740 (0.88)
L15	B&B	2.200 (0.70)	2.036 (1.00)	2.056 (1.00)	2.007 (1.00)	2.010 (1.00)
	Opt	1.826 (0.65)	2.037 (1.00)	2.052 (1.00)	2.008 (1.00)	2.009 (1.00)
	Rand	0.935 (0.99)	2.361 (0.89)	2.442 (0.72)	3.101 (0.86)	3.104 (0.86)
	All	1.295 (0.41)	2.227 (0.92)	2.368 (0.70)	2.686 (0.86)	2.689 (0.86)
L16	B&B	2.132 (0.69)	2.031 (1.00)	2.048 (1.00)	2.010 (1.00)	2.010 (1.00)
	Opt	1.706 (0.62)	2.027 (1.00)	2.042 (1.00)	2.000 (1.00)	2.000 (1.00)
	Rand	0.945 (0.99)	2.303 (0.89)	2.513 (0.73)	3.086 (0.87)	3.083 (0.87)
	Up	4.492 (0.19)	2.680 (0.97)	2.817 (0.98)	2.668 (0.95)	2.678 (0.96)
	All	0.983 (0.31)	2.179 (0.93)	2.482 (0.74)	2.607 (0.88)	2.605 (0.88)
L18	B&B	2.146 (0.69)	2.041 (1.00)	2.057 (1.00)	2.015 (1.00)	2.015 (1.00)
Ratz4	B&B	0.983 (1.00)	2.880 (0.97)	2.981 (0.90)	0.523 (0.81)	1.004 (1.00)
	Opt	0.987 (1.00)	2.245 (0.94)	2.265 (0.94)	2.974 (0.86)	2.951 (0.85)
	Rand	1.240 (0.42)	2.131 (0.89)	1.763 (0.63)	2.833 (0.84)	2.824 (0.84)
	Up	**0.000** (0.76)	1.978 (1.00)	2.000 (1.00)	2.021 (1.00)	2.019 (1.00)
	All	1.313 (0.51)	2.166 (0.92)	2.292 (0.74)	2.847 (0.88)	2.839 (0.88)
RB	B&B	1.199 (0.46)	1.994 (1.00)	1.990 (1.00)	1.993 (1.00)	1.993 (1.00)
	Opt	0.767 (0.43)	2.005 (1.00)	1.995 (1.00)	1.992 (1.00)	2.003 (1.00)
	Rand	0.475 (0.14)	2.233 (0.88)	0.851 (0.34)	2.940 (0.80)	2.941 (0.80)
	Up	0.013 (0.33)	3.757 (0.99)	3.648 (0.99)	3.831 (1.00)	3.858 (0.99)
	All	0.472 (0.17)	2.199 (0.94)	1.707 (0.59)	2.473 (0.83)	2.474 (0.83)
S5	B&B	1.060 (1.00)	3.355 (0.92)	3.326 (0.92)	3.210 (0.89)	3.262 (0.90)
	Opt	0.997 (1.00)	2.275 (0.70)	3.281 (0.90)	3.220 (0.90)	3.223 (0.90)
	Rand	1.031 (0.99)	1.935 (0.90)	1.361 (0.56)	0.850 (0.41)	0.852 (0.41)
	Up	**0.000** (0.29)	1.984 (1.00)	2.019 (1.00)	0.062 (0.25)	0.058 (0.24)
	All	1.023 (0.98)	2.453 (0.89)	2.568 (0.77)	2.137 (0.70)	2.147 (0.70)
Sch21	B&B	2.043 (1.00)	2.072 (1.00)	2.068 (0.99)	2.068 (1.00)	1.066 (0.99)
	Opt	2.007 (1.00)	2.057 (1.00)	2.058 (1.00)	2.038 (1.00)	1.043 (0.99)
	Rand	1.027 (0.96)	2.112 (0.85)	1.204 (0.44)	3.171 (0.85)	1.026 (0.91)
	Up	2.179 (0.21)	8.154 (1.00)	8.074 (1.00)	8.140 (1.00)	8.087 (1.00)
	All	1.851 (0.90)	2.216 (0.94)	2.074 (0.76)	2.432 (0.89)	1.212 (0.88)
Sch25	B&B	1.933 (0.62)	2.000 (1.00)	2.009 (1.00)	2.012 (1.00)	2.011 (1.00)
	Opt	0.004 (0.24)	1.995 (1.00)	1.999 (1.00)	1.992 (1.00)	1.995 (1.00)
	Rand	0.429 (0.30)	1.952 (0.84)	0.447 (0.30)	2.818 (0.82)	2.823 (0.82)
	Up	1.442 (0.17)	1.992 (1.00)	1.960 (1.00)	2.002 (1.00)	1.972 (1.00)
	All	0.956 (0.37)	1.983 (0.94)	1.533 (0.52)	2.328 (0.85)	2.328 (0.85)

Table 2.1: *(continued)* The empirical convergence speed α with the correlation coefficient in brackets for the test functions with different sets of boxes.

Problem	Set	α				
name	name	Natural	Centered	Baumann	For. Slope	Rev. Slope
Sch31	B&B	2.024 (1.00)	2.041 (1.00)	2.025 (1.00)	2.034 (1.00)	2.032 (1.00)
	Opt	2.039 (1.00)	2.053 (1.00)	2.044 (1.00)	2.033 (1.00)	2.044 (1.00)
	Rand	0.768 (0.24)	1.872 (0.83)	1.235 (0.51)	2.959 (0.82)	2.962 (0.82)
	Up	**24.032** (0.63)	4.108 (0.97)	3.669 (0.98)	3.771 (0.95)	3.916 (0.97)
	All	1.057 (0.38)	1.952 (0.89)	1.712 (0.55)	2.657 (0.84)	2.661 (0.84)
Sch32	B&B	1.808 (0.61)	2.007 (1.00)	2.004 (1.00)	2.007 (1.00)	2.003 (1.00)
	Opt	0.824 (0.31)	2.012 (1.00)	2.013 (1.00)	1.998 (1.00)	1.991 (1.00)
	Rand	0.912 (0.28)	1.845 (0.85)	0.866 (0.34)	2.975 (0.85)	2.975 (0.85)
	Up	1.684 (0.34)	4.180 (1.00)	3.873 (1.00)	4.187 (1.00)	4.191 (1.00)
	All	0.868 (0.30)	2.067 (0.95)	1.867 (0.67)	2.479 (0.88)	2.476 (0.88)
$Sch37_{10}$	B&B	0.000 (0.00)	10.080 (1.00)	9.916 (1.00)	10.093 (1.00)	**10.103** (1.00)
$Sch37_5$	B&B	0.002 (0.13)	10.099 (1.00)	9.912 (1.00)	10.085 (1.00)	**10.108** (1.00)
	Opt	0.000 (0.02)	10.076 (1.00)	9.888 (1.00)	10.066 (1.00)	10.137 (1.00)
	Rand	0.000 (0.01)	1.863 (0.97)	0.795 (0.47)	2.201 (0.84)	2.204 (0.84)
	Up	0.001 (0.02)	10.089 (1.00)	10.039 (1.00)	10.039 (1.00)	10.104 (1.00)
	All	0.000 (0.03)	6.100 (0.74)	5.973 (0.75)	6.255 (0.76)	6.253 (0.76)
Sch218	B&B	2.044 (0.98)	2.744 (0.84)	2.784 (0.84)	2.846 (0.85)	2.846 (0.85)
	Opt	2.093 (1.00)	2.038 (1.00)	2.030 (1.00)	2.549 (0.89)	2.545 (0.89)
	Rand	0.746 (0.23)	2.104 (0.85)	0.866 (0.41)	2.774 (0.82)	2.772 (0.82)
	Up	1.994 (0.99)	2.004 (1.00)	1.954 (0.99)	2.045 (1.00)	2.017 (1.00)
	All	1.670 (0.66)	2.534 (0.83)	2.215 (0.72)	2.816 (0.85)	2.815 (0.85)
SHCB	B&B	1.033 (1.00)	2.985 (0.97)	3.257 (0.94)	3.421 (0.91)	3.407 (0.91)
	Opt	1.008 (1.00)	1.624 (0.88)	3.220 (0.94)	3.085 (0.86)	3.079 (0.86)
	Rand	1.005 (0.99)	2.215 (0.90)	1.696 (0.60)	3.021 (0.84)	3.008 (0.84)
	Up	4.029 (1.00)	6.046 (1.00)	5.918 (1.00)	6.031 (1.00)	5.968 (1.00)
	All	1.130 (0.96)	2.698 (0.93)	3.079 (0.83)	3.302 (0.89)	3.291 (0.89)
THCB	B&B	2.152 (1.00)	2.090 (1.00)	2.107 (1.00)	2.096 (1.00)	2.090 (1.00)
	Opt	2.072 (1.00)	2.035 (1.00)	2.028 (1.00)	2.040 (1.00)	2.049 (1.00)
	Rand	1.001 (0.99)	1.952 (0.84)	1.359 (0.48)	3.030 (0.83)	3.009 (0.82)
	Up	3.823 (0.99)	5.712 (1.00)	5.782 (1.00)	5.764 (0.99)	5.698 (0.99)
	All	1.528 (0.86)	2.215 (0.92)	2.349 (0.72)	2.883 (0.88)	2.878 (0.87)
All	B&B	2.026 (0.67)	2.818 (0.82)	2.843 (0.82)	3.237 (0.83)	3.121 (0.80)
	Opt	1.198 (0.37)	2.554 (0.64)	2.801 (0.68)	2.862 (0.68)	2.770 (0.66)
	Rand	0.975 (0.35)	2.125 (0.86)	1.607 (0.56)	2.887 (0.81)	2.684 (0.76)
	Up	2.686 (0.19)	4.672 (0.59)	4.651 (0.59)	4.479 (0.55)	4.465 (0.55)
	All	1.513 (0.51)	2.594 (0.82)	2.700 (0.72)	3.142 (0.83)	2.995 (0.79)

Comparing the results obtained from the centered form and Baumann's optimal centered form, note that theoretically Baumann's form is at least as good, but in some cases the empirical convergence speed of the plain centered form is slightly better in the parameter α (see for instance the results for the function of Branin for the B&B and Rand series). This can only happen since the constant c is smaller in these cases for Baumann's optimal centered form.

Comparing the Forward and Reverse Slope forms, we can see that they usually give very similar results. The only exception is the Schwefel 21 function, where the convergence speed for the Forward Slope form is two times bigger than for the Reverse Slope Form. In comparison with the centered form,

we can see that for the Random series the Slope forms almost always give better results, while for the B&B and Opt series this is true only in some cases. Another interesting result is that the convergence speed for the Baumann form for the Random series is quite bad, in average only 1.6. This is misleading, because the corresponding c values are always much smaller than for the other forms, therefore it is almost always the best inclusion. The reason for this strange result is that, usually, in a random interval the functions are monotonous, thus the Baumann inclusion evaluates the two endpoints of the interval. In this way it obtains the range over the box (within rounding errors), so the overestimation is set to 10^{-30}. Therefore the linear regression leans forward to the horizontal line at -30. This can also be read from the correlation coefficients.

Examining the upward going sequences, notice that the higher the α is, the worse the inclusion function is. For instance, for the Goldstein-Price function (GP) the upward going sequence gives interestingly bad results, although it fits earlier experiences that the minimization of this function is hard. The function Levy 14 (L14) is also interesting: although for the B&B sequence the natural inclusion function is the best, for the upward going sequence it has a very bad inclusion. The surprising results of the 5 and 10 dimensional functions Schwefel-37 (Sch37_5 and Sch37_{10}) are explained by the formula of the function, $f(x) = \sum_{i=1}^{n} x_i^{10}$. Here the zero convergence speed of the natural extension should be understood as ∞, because its inclusion is always exact, which gives an absolute horizontal line at -30.

At the end of the table one can read the empirical convergence speed generalized for all functions. The last line shows the empirical convergence orders for all the functions on all the boxes. These values are larger than the respective theoretical convergence orders. This can be explained by the fact that the empirical convergence speed measures the average speed, while the theoretical convergence order gives a lower bound.

The results for parameter c are less informative, but still the values are important to find out which inclusion function is the most advantageous for a given size of box. Table A.1 with the c values can be found in Appendix A.

2.1.6 Evaluation and discussion

For the usage of inclusion functions in numerical algorithms the most important question is which inclusion function provides the most precise lower and upper bounds. We have considered this question within a scenario similar to one that we can encounter when solving standard global optimization test problems.

The rule of thumb folklore in interval mathematics states that for intervals with a width larger than one, the natural interval extension should be used, while for smaller intervals it is better to use a gradient or slope based inclusion function, due to their convergence order.

In our study we have determined, for a given problem and a given interval sequence those inclusion functions which have the minimal overestimation for the interval size $\mathrm{w}(\boldsymbol{x}) = 10^{-12}$ and $\mathrm{w}(\boldsymbol{x}) = 10$. This was done on the basis of the data collected by linear regression (Tables 2.1 and A.1). The cases when one inclusion function gives the minimal overestimation everywhere are shown in Table 2.2.

Table 2.2: Test functions with only one best inclusion function in the interval $[10^{-12}, 10]$.

Natural	B&B:	L8, L9, L10, L11, L12, L13, L14, L15, L16, L18, RB, Sch21, Sch31, Sch32, Sch25
	Opt:	L10, L13, L14, L15, L16, L8, L9, RB, Sch21, Sch31, Sch32, Sch25
	Up:	GP, L16, Sch31, Sch25
	All:	Sch25
Baumann	Rand:	L8, L9, L10, L5, L3, L16, L15, L14, L13, SCHB
	All:	Sch31, Ratz4, SHCB, L5
Rev. Slope	Opt:	L5, SHCB
	B&B:	Sch218
Forw. Slope	Up:	Sch21

Table 2.3: The two most precise inclusion functions at the interval widths 10^{-12} and 10, together with the transition point.

Baumann–Natural	B&B:	Griew5 ($3.0243 \cdot 10^{-8}$)
	All:	L10 (0.245), L13 (1.074), L14 (0.0427), L15 (1.232), L16 (0.0005118), L8 (6.893), L9 (0.540), S5 (0.157)
	Up:	S5 (0.2985)
Centered–Natural	B&B:	S5 (0.0867)
	Up:	L5 (2.1007)
Forw. Slope–Natural	B&B:	L3 (2.1887), L5 (1.7504), SHCB (3.4291)
	Opt:	GP (3.0196), Branin ($8.794 \cdot 10^{-10}$)
	Up:	Sch218 (0.424), Ratz4 (1.105), SHCB (5.811), THCB (6.229)
	All:	THCB ($7.4389 \cdot 10^{-10}$)
Rev. Slope–Natural	B&B:	Sch37_{10} (0.001338), Sch37_{5} (0.001397), GP (0.2114)
	Rand:	Sch25 ($1.237 \cdot 10^{-9}$)
	Up:	L3 (0.0998), Sch32 ($1.447 \cdot 10^{-9}$), RB ($7.931 \cdot 10^{-9}$)
	All:	L3 (1.477), RB ($1.03511 \cdot 10^{-10}$)
Forw. Slope–Baumann	Rand:	Ratz4 ($1.057 \cdot 10^{-12}$), Sch21 ($1.394 \cdot 10^{-9}$), Sch218 ($7.150 \cdot 10^{-10}$), GP ($9.6332 \cdot 10^{-10}$)
	All:	Sch21 ($1.5586 \cdot 10^{-9}$), Sch218 ($4.8106 \cdot 10^{-8}$)
Rev. Slope–Baumann	Rand:	Branin ($1.597 \cdot 10^{-11}$), Sch31 ($2.3726 \cdot 10^{-12}$), THCB ($3.284 \cdot 10^{-9}$), Sch32 ($4.198 \cdot 10^{-9}$), RB ($7.468 \cdot 10^{-10}$)
	All:	Branin (0.01046), THCB ($9.087 \cdot 10^{-10}$)
Forw. Slope–Rev. Slope	Opt:	Sch218 (0.001778), Ratz4 ($3.675 \cdot 10^{-5}$)
Natural–Rev. Slope	Up:	L14 (0.1247)
	B&B:	THCB ($3.535 \cdot 10^{-7}$)
Natural–Forw. Slope	Up:	L8 (5.124)
	Opt:	THCB (8.058)
Baumann–Forw. Slope	B&B:	Ratz4 (0.3479)
	All:	S5 ($3.990 \cdot 10^{-8}$)
	Rand:	S5 ($5.823 \cdot 10^{-8}$)
Rev. Slope–Centered	Opt:	L3 (1.772)
Rev. Slope–Forw. Slope	B&B:	Branin ($1.1721 \cdot 10^{-10}$)

When the inclusion functions were different for the two different box sizes, we have also calculated the intersection point of the related lines, i.e. the transition point at which the user should switch from the second to the first inclusion function to have better precision. The corresponding results are given in Table 2.3. According to our experiences, these bounds are realistic when solving usual global optimization problems. For instance, check the line which contains the solution of the Goldstein-Price problem (Rev.Slope–Natural, B&B), also illustrated in Figure 2.1. As can be seen on the bottom right graph, the best inclusion function for small intervals is the reverse slope arithmetic, while for larger intervals the natural interval extension provides the most precise inclusions. These optimal inclusion functions for the GP, B&B case, and also the transition point of 0.2114 are realistic, and they met our expectations.

Out of the over 116 cases solved successfully, in 50 instances, a single inclusion function proved to be

Table 2.4: The two most precise inclusion functions generalized to all of the functions at interval widths 10^{-12} and 10, together with their transition point.

Forward Slope – Natural	B&B:	0.05309
	Opt:	$1.919 \cdot 10^{-7}$
	All:	0.006012
Forward Slope – Baumann	Rand:	$1.21444 \cdot 10^{-12}$
Centered – Natural	Up:	$1.22637 \cdot 10^{-9}$

the best for the whole range studied. It was mostly the natural interval extension that can be explained by the relative simplicity of the test problems. On the one hand, for small intervals, the monotonicity of the function makes the natural interval extension precise (before outward rounding). On the other hand, there is a big variety of test functions where the natural interval extension (as the only choice) is the best most often for the most interesting Branch and Bound sequence.

In Table 2.3 the Natural and the Slope forms appear the most number of times, while the Centered form only in a few cases. For the increasing interval sequence (denoted by Up) the used widths of 10^{-12} and 10 are not always realistic (the smallest box for this set has a width of 1). It is also reflected by the fact that in many cases the natural interval extension was the best inclusion function for the larger bound.

Beyond the detailed results for the test problems, Table 2.4 gives the most informative results comprising the knowledge collected on singular interval sequences when all the problems are considered together. According to the average behavior, when inclusion functions are used in a Branch and Bound method for optimization, the natural arithmetic is the best choice for intervals with a width larger than 0.05, while the forward mode slope is the best for smaller intervals. The same pair of inclusion functions should be selected for an embedded sequence of intervals converging to a global minimizer point, and also for the total set of intervals investigated (although with different transition points). For random intervals the couple forward slope and Baumann centered form is the best option (although taking the value of the intersection point into account, the Baumann form is better for most of the intervals of the sequence). In a similar way, for larger and increasing size interval sequences the couple centered–natural is the best one.

These results are obtained taking into account only the empirical convergence speed. However, some of those inclusion functions provide more useful information which can be used in global optimization algorithms. For instance, with the centered and Baumann forms, an inclusion of the gradient is available, which allows us to use the monotonicity test (see Section 1.4). Another pruning test described in Section 2.2 can be applied with Baumann, centered and slope arithmetic forms. This fact must also be taken into account when selecting an inclusion function for solving a problem. In particular, as a different study, this information can be used when solving the problems with different inclusion functions. Let us denote the examined inclusion functions as $\boldsymbol{f}_i(\boldsymbol{x}), i \in I$.

To check, in practice, the usefulness of the empirical convergence speed for selecting an inclusion function when solving optimization problems, we can make an adaptive multi-inclusion algorithm (Algorithm 2.1), and measure the time needed for solving a given problem when using different inclusion functions. In Algorithm 2.1 we apply all the mentioned accelerating devices which are available depending on the used inclusion function. The function $Ind(\mathrm{w}(\boldsymbol{x})) \to J(\subseteq I)$ in Algorithm 2.1 decides the inclusion function to be used depending on the width of the box. Such a study has been carried out for a class of facility location problems (see Section 5.4 for the computational results).

As a summary, we can say that the introduced empirical convergence speed allows a deeper understanding of the behavior of the overestimation. We can use it to give a practical answer to the following question: which is the best inclusion function for a given function at a given state in the optimization process? According to the present study, when solving an optimization problem, the best is to use the natural interval extension when the current interval's width is larger than 0.05, and forward mode

Algorithm 2.1 Adaptive multi-inclusion B&B algorithm

Input: $(\boldsymbol{s}, Ind(\mathrm{w}(\boldsymbol{x})))$ $\quad Ind(\mathrm{w}(\boldsymbol{x})) \rightarrow J(\subseteq I)$: *indexes of inclusion functions*
$\mathcal{L}_{\mathcal{W}} \leftarrow \boldsymbol{s}$, $\mathcal{L}_{\mathcal{S}} \leftarrow \emptyset$
while ($\mathcal{L}_{\mathcal{W}} \neq \emptyset$) **do**
  Select an interval $\boldsymbol{x}$ from $\mathcal{L}_{\mathcal{W}}$ *Selection Rule*
  Evaluate $\boldsymbol{f}(\boldsymbol{x}) = \bigcap_{i \in I(\mathrm{w}(\boldsymbol{x}))} \boldsymbol{f}_i(\boldsymbol{x})$, Update $\tilde{f}$
  Discard $\boldsymbol{z} \in \mathcal{L}_{\mathcal{S}}, \mathcal{L}_{\mathcal{W}}$ if $\underline{\boldsymbol{f}}(\boldsymbol{z}) > \tilde{f}$ *Cut-Off Test*
  if ($\boldsymbol{x}$ cannot be discarded or reduced) *Discarding Tests*
    Divide $\boldsymbol{x}$ into subintervals $\boldsymbol{x}^1, \boldsymbol{x}^2$ *Division Rule*
    for $i = 1, 2$ **do**
      if ($\boldsymbol{x}^i$ satisfies the termination criterion) *Termination Rule*
        Store $\boldsymbol{x}^i$ in $\mathcal{L}_{\mathcal{S}}$
      else Store $\boldsymbol{x}^i$ in $\mathcal{L}_{\mathcal{W}}$
return $\mathcal{L}_{\mathcal{S}}$

slope arithmetic for smaller intervals. The presented methodology is also applicable to other verified numerical algorithms, e.g. root finding, differential equations, etc.

2.2 Efficient use of gradient information: a pruning method

2.2.1 The one-dimensional case

In this section, the main idea of the one-dimensional pruning technique, which is the basic idea of the work in the next sections, is shown. The aim of this method is to prune out parts of a given interval, which are proved not to contain any global minimizer point. A linear lower bounding function of the objective function is built (similarly as is done in Lipschitz optimization [74]). If a good upper bound $\tilde{f}$ of the global minimum is known, then those parts of the interval, where the lower bounding function is greater than $\tilde{f}$, can be deleted.

Let us demonstrate it with an example. Suppose that for an interval $\boldsymbol{x}$ and for a point $c \in \boldsymbol{x}$ bounds on the slopes $[l, u]$ from c over $\boldsymbol{x}$ are known (see Figure 2.2). These bounds can be obtained by the inclusion of the gradient of the objective function, by slope arithmetic, or by a Lipschitz constant. Computing a lower bound of $f(c)$ one can build a linear lower bounding function as is shown in Figure 2.2.

When $\underline{\boldsymbol{f}}(c) > \tilde{f}$, as in our example, the part of the interval where the lower bounding function is above $\tilde{f}$ can be removed. In our example, the interval $\boldsymbol{y}$ can be discarded (see Figure 2.2). Pruning out an interval can be viewed also as a division rule generating one or two subintervals.

The main idea of the pruning technique has been discovered by many researchers, although all of them improved it somehow. For instance, Ratz [127] used interval slopes to build the linear lower bounding function (and also an upper bounding function), Casado et al. [12] used support values at the endpoints of the interval to get a better lower bounding function (see the next section for details), while Vinkó et al. [148] built a Kite inclusion function from the linear bounding functions.

2.2.2 The multi-dimensional pruning method based on new support functions

This section analyzes and evaluates an efficient n-dimensional discarding test, which is a natural extension of the method described in the previous section to multi-dimensional problems. This method takes advantage of all available information to estimate better bounds of the function, therefore it is able to discard whole boxes or some parts of them at an early phase.

Let us introduce some new notation that is necessary for the explanation of the method. We define $\boldsymbol{x}(i : p) = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, [p, p], \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_n)$, where p is a point in $\boldsymbol{x}_i$. Some particular cases are $\boldsymbol{x}_i^l = \boldsymbol{x}(i : \underline{x}_i)$, $\boldsymbol{x}_i^r = \boldsymbol{x}(i : \overline{x}_i)$, and $\boldsymbol{x}_i^m = \boldsymbol{x}(i : \mathrm{m}(\boldsymbol{x}_i))$. A lower bound of $f(x)$ at $\boldsymbol{x}$ will be denoted by $lbf(\boldsymbol{x})$,

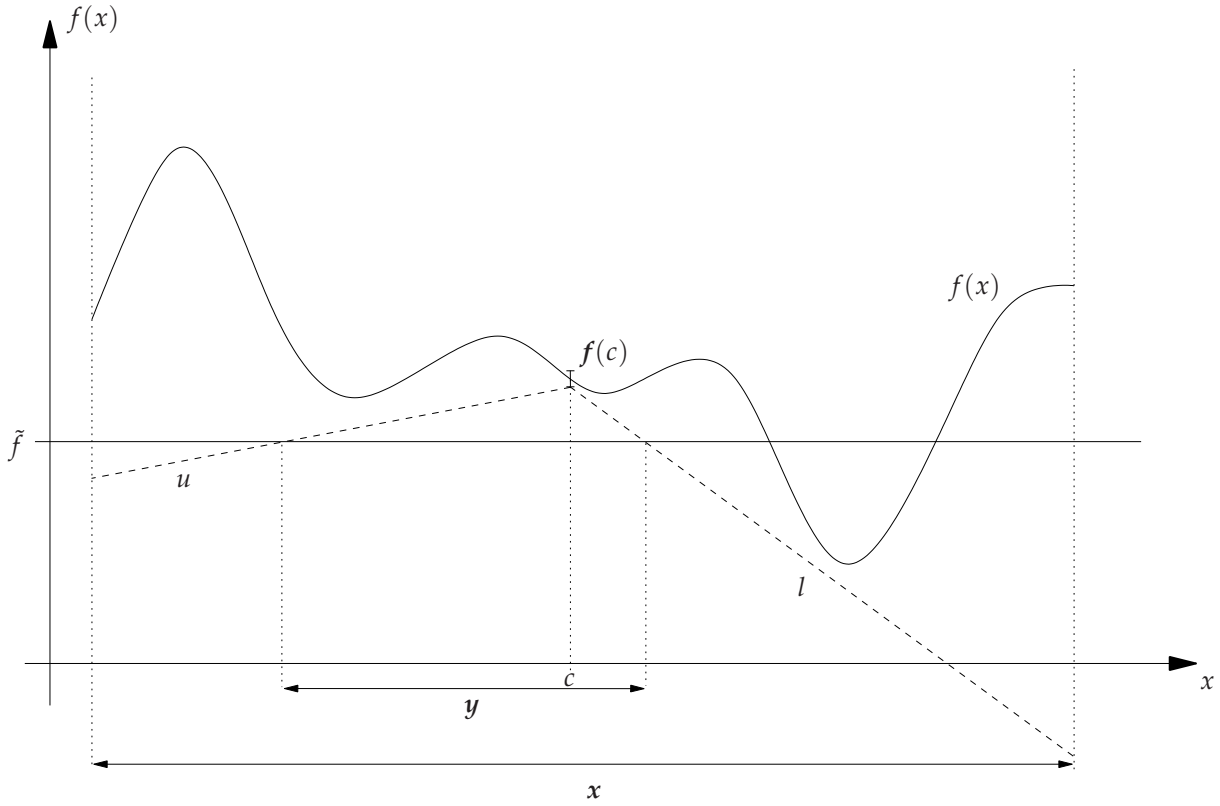


Figure 2.2: Example of pruning for a one-dimensional problem.

and the support function of $f(x)$ at the borders of the interval $\boldsymbol{x}$ by $sp(\boldsymbol{x}) = (sp(\boldsymbol{x}_1), \ldots, sp(\boldsymbol{x}_n))$, where $sp(\boldsymbol{x}_i) = \{sp(\boldsymbol{x}_i^l), sp(\boldsymbol{x}_i^r)\} = \{lbf(\boldsymbol{x}_i^l), lbf(\boldsymbol{x}_i^r)\}, i = 1 \ldots n$.

In this section it is supposed that inclusion functions can be evaluated for $f(\boldsymbol{x})$ ($\boldsymbol{f}(x)$) and for its gradient $f'(x)$ ($\boldsymbol{g}(\boldsymbol{x})$) over any box $\boldsymbol{x} \in S$.

We propose the following Advanced Multidimensional Interval analysis Global Optimization algorithm (AMIGO) to solve problem (1.1) when only bound constraints are present, i.e. $\boldsymbol{s} = S$. It uses the gradient information as the general interval B&B method (see Section 1.4) does, but, due to a more efficient usage of the available information, it constructs support functions which are sometimes closer to the objective function. It enables us to obtain better lower bounds and to diminish the width of the current interval. Hereinafter, it will be shown that this new method (AMIGO) has a quite promising performance in comparison to the traditional method.

In order to proceed with the description of the method, first we present the theoretical results which are the foundations of the new support functions and explain how the new lower bounds of $f(\boldsymbol{x})$ are evaluated. As Ratz has proposed in [126], we will analyze a feasible approach which computes enclosures based on values of particular components of the gradient vector $\boldsymbol{g}(\boldsymbol{x}) = (\boldsymbol{g}_1(x), \ldots, \boldsymbol{g}_n(x))$, i.e. this corresponds to a componentwise derivative computation of the support functions.

Theorem 2.6. *Let $\boldsymbol{x}$ and $\boldsymbol{s}$ be closed intervals such that $\boldsymbol{x} \subseteq \boldsymbol{s} \subset \mathbb{R}^n$ and let $f : \boldsymbol{s} \to \mathbb{R}$ be a continuously differentiable function. Let us suppose that, given a point $c = (c_1, \ldots, c_n) \in \boldsymbol{x}$, for an interval $\boldsymbol{x}(i : c_i) \subseteq \boldsymbol{x}$ a lower bound $lbf(\boldsymbol{x}(i : c_i))$ of $f(\boldsymbol{x}(i : c_i))$ is determined and an enclosure $\boldsymbol{g}(\boldsymbol{x})$ of $f'(\boldsymbol{x})$ is obtained. For a given current upper bound $\tilde{f}$ of the global minimum f^*, the set*

$$\boldsymbol{x}_{go}^a = \left\{x \in \boldsymbol{x} : lbf(\boldsymbol{x}(i : c_i)) + \min\left\{\boldsymbol{g}_i(\boldsymbol{x}) \cdot (x_i - c_i)\right\} \leq \tilde{f}\right\}$$

contains all the global minimizer points of $f(x)$ in $\boldsymbol{x}$, if any.

Proof. For a minimizer point $x^* \in X^*$ it applies that $f(x^*) \leq \tilde{f}$. It is easy to see from the mean-value theorem (see (1.4)) that a minimizer point $x^* \in \boldsymbol{x} \cap X^*$ has to fulfill:

$$lbf(\boldsymbol{x}(i:c_i)) + \min\{\boldsymbol{g}_i(\boldsymbol{x}) \cdot (x_i^* - c_i)\} \leq f(x^*) \leq \tilde{f},$$

i.e. the linear lower bounding function has to be below the constant line $\tilde{f}$. Thus the set where all the global minimizer points of $f(x)$ in $\boldsymbol{x}$, if any, are included is

$$\boldsymbol{x}_{go}^a = \left\{x \in \boldsymbol{x} : lbf(\boldsymbol{x}(i:c_i)) + \min\{\boldsymbol{g}_i(\boldsymbol{x}) \cdot (x_i - c_i)\} \leq \tilde{f}\right\}.$$

□

Notice that if $0 \notin \boldsymbol{g}(\boldsymbol{x})$ then x^* cannot be in the interior of $\boldsymbol{x}$. On the other hand, if $\tilde{f} < lbf(\boldsymbol{x}(i:c_i))$ then $\boldsymbol{x}(i:c_i) \not\subseteq X^*$ and if $0 \in \boldsymbol{g}(\boldsymbol{x})$ from Theorem 2.6 one can derive that $\boldsymbol{x}_{go}^a = \boldsymbol{u}_{c_i} \cup \boldsymbol{v}_{c_i}$ where

$$\boldsymbol{u}_{c_i} = \left\{x \in \boldsymbol{x} \mid x_i \leq c_i - \frac{lbf(\boldsymbol{x}(i:c_i)) - \tilde{f}}{\overline{\boldsymbol{g}_i}(\boldsymbol{x})}\right\} \quad \text{and} \quad \boldsymbol{v}_{c_i} = \left\{x \in \boldsymbol{x} \mid x_i \geq c_i - \frac{lbf(\boldsymbol{x}(i:c_i)) - \tilde{f}}{\underline{\boldsymbol{g}_i}(\boldsymbol{x})}\right\}. \quad (2.3)$$

In a simpler way it can be written as $\boldsymbol{x}_{go}^a = \boldsymbol{x} \backslash \boldsymbol{w}$ where $\boldsymbol{w}$ is given by

$$\boldsymbol{w} = \left\{x \in \boldsymbol{x} \mid x_i \in \left[c_i - \frac{lbf(\boldsymbol{x}(i:c_i)) - \tilde{f}}{\overline{\boldsymbol{g}_i}(\boldsymbol{x})}, c_i - \frac{lbf(\boldsymbol{x}(i:c_i)) - \tilde{f}}{\underline{\boldsymbol{g}_i}(\boldsymbol{x})}\right]\right\}. \quad (2.4)$$

As an example, $\boldsymbol{w}$ has been depicted in Figure 2.3 for the case $c_1 = x_1^m$ and $0 \in \boldsymbol{g}(\boldsymbol{x})$.

Theorem 2.7. *Let us consider a continuously differentiable function $f : S \to \mathbb{R}$, where $\boldsymbol{s}$ is a closed interval in $\mathbb{R}^n$ and intervals $\boldsymbol{x}$, $\boldsymbol{y}$ such that $\boldsymbol{x} \subseteq \boldsymbol{y} \subseteq \boldsymbol{s}$. If for some $i \in \{1, \ldots, n\}$:*

1. *lower bounds $lbf(\boldsymbol{x}_i^l)$ and $lbf(\boldsymbol{x}_i^r)$ of $f(\boldsymbol{x}_i^l)$ and $f(\boldsymbol{x}_i^r)$, respectively, have been evaluated;*
2. *a current upper bound $\tilde{f}$ of f^* is such that*

$$\tilde{f} \leq \min\{lbf(\boldsymbol{x}_i^l), lbf(\boldsymbol{x}_i^r)\};$$

3. *bounds $\boldsymbol{g}_i = \boldsymbol{g}_i(\boldsymbol{y})$ of $f_i'(\boldsymbol{y})$ have been obtained (so valid bounds for $f_i'(\boldsymbol{x})$), and $0 \in \boldsymbol{g}(\boldsymbol{y})$.*

Then, only the interval

$$\boldsymbol{x}_{go}^b = \left\{x \in \boldsymbol{x} : x_i \in \left[\underline{x}_i - \frac{lbf(\boldsymbol{x}_i^l) - \tilde{f}}{\underline{\boldsymbol{g}_i}}, \overline{x}_i - \frac{lbf(\boldsymbol{x}_i^r) - \tilde{f}}{\overline{\boldsymbol{g}_i}}\right]\right\} \quad (2.5)$$

can contain global minimizers and a lower bound $z(\boldsymbol{x})$ of $f(\boldsymbol{x})$ can be calculated as follows:

$$z(\boldsymbol{x}) = \max_{j \in \{1,\ldots,n\}} \left\{\frac{lbf(\boldsymbol{x}_j^l) \cdot \overline{\boldsymbol{g}_j} - lbf(\boldsymbol{x}_j^r) \cdot \underline{\boldsymbol{g}_j}}{\mathrm{w}(\boldsymbol{g}_j)} + \mathrm{w}(\boldsymbol{x}) \cdot \frac{\underline{\boldsymbol{g}_j} \cdot \overline{\boldsymbol{g}_j}}{\mathrm{w}(\boldsymbol{g}_j)}\right\}. \quad (2.6)$$

Proof. Applying Theorem 2.6 for $\boldsymbol{x}(i:c_i) = \boldsymbol{x}_i^l$ interval $\boldsymbol{u}_{\underline{x}_i}$ from (2.3) is obtained. Similarly, for $\boldsymbol{x}(i:c_i) = \boldsymbol{x}_i^r$ we obtain $\boldsymbol{v}_{\overline{x}_i}$ from (2.3). Then, interval $\boldsymbol{x}_{go}^b$ can be obtained as $\boldsymbol{u}_{\underline{x}_i} \cap \boldsymbol{v}_{\overline{x}_i}$ (see Figure 2.3).

Proof of Equation (2.6) can be obtained considering that $\boldsymbol{x} \subseteq \boldsymbol{y}$, $\boldsymbol{g}(\boldsymbol{x}) \subseteq \boldsymbol{g}(\boldsymbol{y}) = \boldsymbol{g}$ and applying the mean-value theorem, we have

$$f(x) \geq lbf(\boldsymbol{x}_i^l) + \underline{\boldsymbol{g}_i}(\boldsymbol{x}) \cdot (x_i - \underline{x}_i) \quad \text{and} \quad f(x) \geq lbf(\boldsymbol{x}_i^r) + \overline{\boldsymbol{g}_i}(\boldsymbol{x}) \cdot (x_i - \overline{x}_i), \quad i = 1, \ldots, n,\ x \in \boldsymbol{x}.$$

From these inequalities, consequently

$$f(x) \geq \max\left\{\begin{array}{l} lbf(\boldsymbol{x}_i^l) + \underline{\boldsymbol{g}_i} \cdot (x_i - \underline{x}_i) \\ lbf(\boldsymbol{x}_i^r) + \overline{\boldsymbol{g}_i} \cdot (x_i - \overline{x}_i) \end{array}\right\}, \quad i = 1, \ldots, n,\ x \in \boldsymbol{x}, \quad (2.7)$$

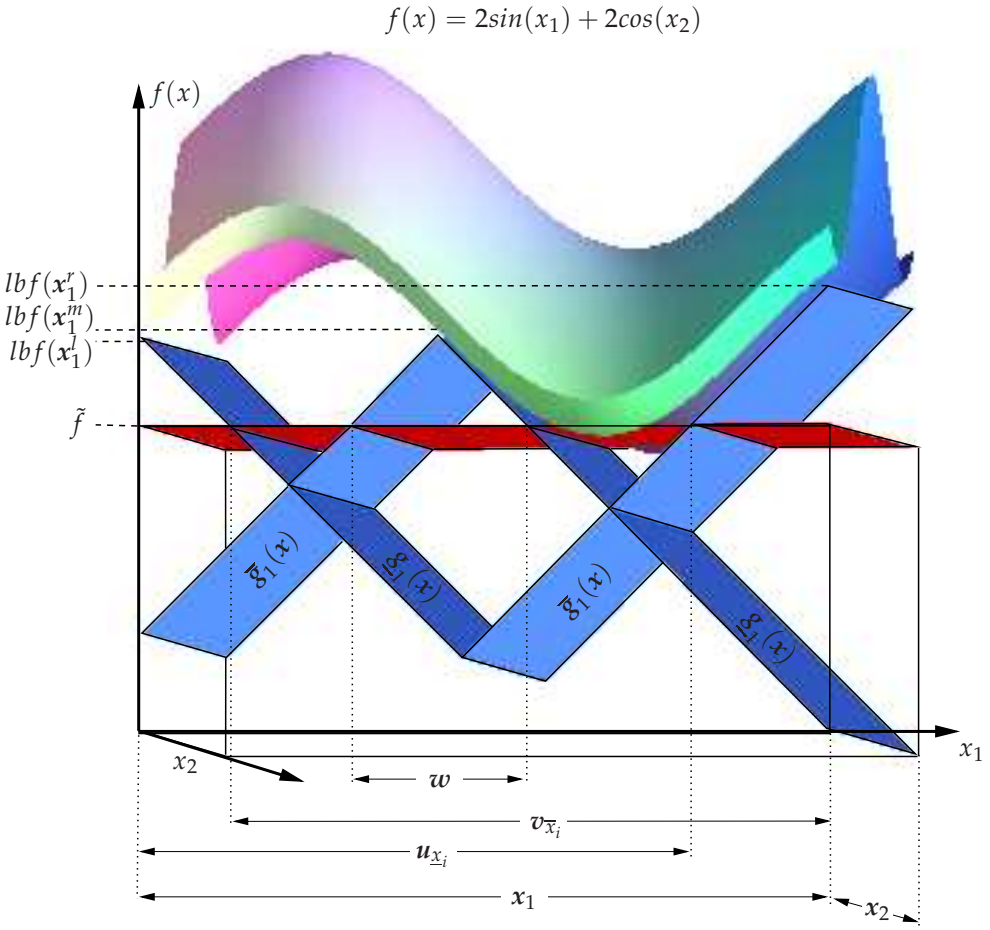


Figure 2.3: Support function using $g_1(x)$, $lbf(x_1^l)$, $lbf(x_1^m)$ and $lbf(x_1^r)$.

which takes its minimal value at the intersection of the two linear lower bounding function, that is

$$\hat{x}_i = \frac{lbf(x_i^l) - lbf(x_i^r) + \overline{g_i}\,\overline{x}_i - \underline{g_i}\,\underline{x_i}}{\mathrm{w}(g_i)}, \qquad i = 1, \ldots, n,$$

and so lower bounds of f over the box x are

$$lbf(x_i^l) + \underline{g_i} \cdot (\hat{x}_i - \underline{x}_i) = \frac{lbf(x_i^l) \cdot \overline{g_i} - lbf(x_i^r) \cdot \underline{g_i}}{\mathrm{w}(g_i)} + \mathrm{w}(x) \cdot \frac{\underline{g_i} \cdot \overline{g_i}}{\mathrm{w}(g_i)} \qquad i = 1, \ldots, n. \tag{2.8}$$

The best lower bound $z(x)$ is the maximal of the lower bounds in (2.8). □

Corollary 2.8. *If for an interval x the inequality $z(x) > \tilde{f}$ is fulfilled then x does not contain any global minimizer.*

Let us now return to problem (1.1). We can build a new lower bound $lbzf(x)$ for $f(x)$ using $\underline{f}(x)$ together with the value of $z(x)$ from (2.6), and the centered form $f_c(x)$ in the following way

$$lbzf(x) = \max\{\underline{f}(x), z(x), \underline{f_c}(x)\}. \tag{2.9}$$

Notice that for any interval w obtained from interval x according to (2.5) ($w = x_{go}^b$), the current value of $\tilde{f}$ is a lower bound of f at w_i^l and w_i^r, i.e. $\tilde{f} \leq f(w_i^l)$ and $\tilde{f} \leq f(w_i^r)$, so $lbf(w_i^l) = lbf(w_i^r) = \tilde{f}$ are

Algorithm 2.2 AMIGO algorithm

Funct AMIGO($\boldsymbol{s}, \boldsymbol{f}, \varepsilon$)

1: $sp(\boldsymbol{s}) = \{\underline{\boldsymbol{f}}(\boldsymbol{s}_i^l), \underline{\boldsymbol{f}}(\boldsymbol{s}_i^r)\},\ i = 1, \ldots, n$
2: $NaturalF(\boldsymbol{s}) = \{NaturalF(\boldsymbol{s}_i^m) = True\},\ i = 1, \ldots, n$ — *$NaturalF(\boldsymbol{x}_i)$ indicates that $\boldsymbol{f}(\boldsymbol{x}_i^m)$ or $\boldsymbol{f}_c(\boldsymbol{x}_i^m)$ is to be evaluated.*
3: Eval $\boldsymbol{f}(\boldsymbol{s}), \overline{\boldsymbol{f}}(\mathrm{m}(\boldsymbol{s})), \boldsymbol{g}(\boldsymbol{s})$, and $lbzf(\boldsymbol{s})$; $\tilde{f} = \overline{\boldsymbol{f}}(\mathrm{m}(\boldsymbol{s}))$
4: $\mathcal{L}_W = \{ListNode(\boldsymbol{s}) = (\boldsymbol{s}, lbzf(\boldsymbol{s}), sp(\boldsymbol{s}), \boldsymbol{f}(\mathrm{m}(\boldsymbol{s})), \boldsymbol{g}(\boldsymbol{s}), NaturalF(\boldsymbol{s}))\}$
5: $\mathcal{L}_S = \{\emptyset\}$ — *Initiation of the final list*
6: **while** ($\mathcal{L}_W \neq \emptyset$) **do**
7: $ListNode(\boldsymbol{x})$=PopHead($\mathcal{L}_W$) — *Extract the first element of $\mathcal{L}_W$*
8: $\{k, sp(\boldsymbol{x}_k^m)\}$ =CoordinateToSplit($\boldsymbol{x}, \boldsymbol{f}, NaturalF(\boldsymbol{x}), \tilde{f}$)
9: $\boldsymbol{w}^1 = \boldsymbol{x}$; $\boldsymbol{w}_k^1 = [\underline{x}_k, \mathrm{m}(\boldsymbol{x}_k)]$
10: $sp(\boldsymbol{w}^1) = sp(\boldsymbol{x})$; $sp(\boldsymbol{w}_k^1) = \{sp(\boldsymbol{x}_k^l), sp(\boldsymbol{x}_k^m)\}$
11: $NaturalF(\boldsymbol{w}^1) = NaturalF(\boldsymbol{x})$
12: $\boldsymbol{w}^2 = \boldsymbol{x}$; $\boldsymbol{w}_k^2 = [\mathrm{m}(\boldsymbol{x}_k), \overline{x}_k]$
13: $sp(\boldsymbol{w}^2) = sp(\boldsymbol{x})$; $sp(\boldsymbol{w}_k^2) = \{sp(\boldsymbol{x}_k^m), sp(\boldsymbol{x}_k^r)\}$
14: $NaturalF(\boldsymbol{w}^2) = NaturalF(\boldsymbol{x})$
15: **for** ($i = 1, 2$) **do**
16: GradientTest($\boldsymbol{w}_k^i, \boldsymbol{g}_k(\boldsymbol{x}), sp(\boldsymbol{w}_k^i), \tilde{f}$)
17: **if** ($\mathrm{w}(\boldsymbol{w}^i) > 0$)
18: Eval $\boldsymbol{f}(\boldsymbol{w}^i), \boldsymbol{g}(\boldsymbol{w}^i)$
19: **if** (MonotonocityTest($\boldsymbol{g}(\boldsymbol{w}^i)$)
20: **continue**
21: **if** ($\overline{\boldsymbol{f}}(\mathrm{m}(\boldsymbol{w}^i)) < \tilde{f}$)
22: $\tilde{f} = \overline{\boldsymbol{f}}(\mathrm{m}(\boldsymbol{w}^i))$; CutOffTest($\mathcal{L}_W, \mathcal{L}_S$)
23: **if** ($lbzf(\boldsymbol{w}^i) \leq \tilde{f}$)
24: **if** ($\mathrm{w}(\boldsymbol{w}^i) \leq \varepsilon$)
25: Save $ListNode(\boldsymbol{w}^i)$ in $\mathcal{L}_S$
26: **else**
27: Save $ListNode(\boldsymbol{w}^i)$ in $\mathcal{L}_W$
28: **return** $\mathcal{L}_S$

easily available bounds. This fact will be used to save function evaluations for the support function, i.e. $sp(\boldsymbol{w}_i) = \{\tilde{f}, \tilde{f}\}$ will be used.

Additionally, a lower bound of $f(\boldsymbol{x}(i : c_i))$, with $c = (c_1, c_2, \ldots, c_n) \in \boldsymbol{x}$ can be obtained applying the centered form, i.e.

$$lbf(\boldsymbol{x}(i : c_i)) = \underline{\boldsymbol{f}_c}(\boldsymbol{x}(i : c_i)). \tag{2.10}$$

In our implementation, we have to evaluate $\boldsymbol{f}(\mathrm{m}(\boldsymbol{x}))$ and $\boldsymbol{g}(\boldsymbol{x})$ to obtain $\boldsymbol{f}_c(\boldsymbol{x})$ as an inclusion function of $f(x)$. Therefore, a feasible and reliable value of $lbf(\boldsymbol{x}_i^m)$ is $\underline{\boldsymbol{f}_c}(\boldsymbol{x}_i^m)$ (as an alternative to $\underline{\boldsymbol{f}}(\boldsymbol{x}_i^m)$). This lower bound of $f(\boldsymbol{x}_i^m)$ can be obtained for all i without additional inclusion function evaluations, thus it will be used to determine the best coordinate direction for pruning.

The AMIGO algorithm follows the above theoretical results. It improves the traditional algorithm in the following ways: (i) by using $lbzf(\boldsymbol{x})$ instead of $\max\{\underline{\boldsymbol{f}}(\boldsymbol{x}), \underline{\boldsymbol{f}_c}(\boldsymbol{x})\}$, which improves the selection rule, bounding rule, midpoint test and cut-off test and (ii) by a smart use of the gradient information, through an elimination rule (called GradientTest), which is based on Theorem 2.7 and described in Algorithm 2.3.

The AMIGO Algorithm (Algorithm 2.2) uses the routine CoordinateToSplit (line 8) as the subdivision direction rule. This routine returns the coordinate k to bisect the box $\boldsymbol{x}$ and a lower bound of $f(\boldsymbol{x}_k^m)$. Then the algorithm bisects the coordinate k, generating the subboxes $\boldsymbol{w}^1$ and $\boldsymbol{w}^2$ (lines 9 and 12). The values of $sp(\boldsymbol{w}^1)$ and $sp(\boldsymbol{w}^2)$ are inherited from their parent box, with an updated value of the corresponding $sp(\boldsymbol{x}_k^m)$ (returned by CoordinateToSplit routine) in each generated subbox (lines 10 and 13 of

Algorithm 2.3 GradientTest Algorithm

Funct GradientTest $(\boldsymbol{x}, \boldsymbol{g}(\boldsymbol{x}), sp(\boldsymbol{x}), \tilde{f})$

1: **if** ($sp(\boldsymbol{x}^l) > \tilde{f}$)
2: $\quad \boldsymbol{y} = [\underline{x} - \frac{sp(\boldsymbol{x}^l) - \tilde{f}}{\underline{f}'(x)}, \infty)$
3: $\quad sp(\boldsymbol{x}^l) = \tilde{f}; \boldsymbol{x} = \boldsymbol{y} \cap \boldsymbol{x}$
4: **if** ($\mathrm{w}(\boldsymbol{x}) > 0$ **and** $sp(\boldsymbol{x}^r) > \tilde{f}$)
5: $\quad \boldsymbol{y} = (-\infty, \overline{x} - \frac{sp(\boldsymbol{x}^r) - \tilde{f}}{\overline{f}'(x)}]$
6: $\quad sp(\boldsymbol{x}^r) = \tilde{f}; \boldsymbol{x} = \boldsymbol{y} \cap \boldsymbol{x}$
7: **return** $\boldsymbol{x}, sp(\boldsymbol{x})$

Algorithm 2.4 CoordinateToSplit Algorithm

Funct CoordinateToSplit $(\boldsymbol{x}, \boldsymbol{f}, NaturalF(\boldsymbol{x}), \tilde{f})$

1: $lbf(\boldsymbol{x}_i^m) = \underline{f_c}(\boldsymbol{x}_i^m),\ i = 1 \ldots n$
2: **for** $i = 1 \ldots n$ **do**
3: $\quad RejectArea(i)$ =EvalRejectArea$(\boldsymbol{x}, i, lbf(\boldsymbol{x}_i^m), \tilde{f}, \boldsymbol{g}_i(\boldsymbol{x}))$
4: $RejectArea(k_1) = \max_i\{RejectArea(i)\}$
5: **if** ($RejectArea(k_1) > 0$)
6: $\quad sp(\boldsymbol{x}_{k_1}^m) = lbf(\boldsymbol{x}_{k_1}^m)$
7: $\quad$ **return** $k_1, sp(\boldsymbol{x}_{k_1}^m)$
8: $lbf(\boldsymbol{x}_{k_2}^m) = \max_i\{lbf(\boldsymbol{x}_i^m)\}$
9: **if** ($NaturalF(\boldsymbol{x}_{k_2}^m) = True$)
10: $\quad$ **if** ($\mathrm{w}(\boldsymbol{f}(\boldsymbol{x}_{k_2}^m)) > \mathrm{w}(\boldsymbol{f}_c(\boldsymbol{x}_{k_2}^m))$)
11: $\quad\quad NaturalF(\boldsymbol{x}_{k_2}^m) = False$
12: $\quad RejectArea(k_2)$ =EvalRejectArea$(\boldsymbol{x}, k_2, \underline{\boldsymbol{f}}(\boldsymbol{x}_{k_2}^m), \tilde{f}, \boldsymbol{g}_i(\boldsymbol{x}))$
13: **if** ($RejectArea(k_2) > 0$)
14: $\quad sp(\boldsymbol{x}_{k_2}^m) = \underline{\boldsymbol{f}}(\boldsymbol{x}_{k_2}^m)$
15: $\quad$ **return** $k_2, sp(\boldsymbol{x}_{k_2}^m)$
16: $\mathrm{w}(\boldsymbol{g}_{k_3}(\boldsymbol{x}) \cdot (\boldsymbol{x}_{k_3} - \mathrm{m}(\boldsymbol{x}_{k_3}))) = \max_i\{\mathrm{w}(\boldsymbol{g}_i(\boldsymbol{x}) \cdot (\boldsymbol{x}_i - \mathrm{m}(\boldsymbol{x}_i)))\}$
17: $sp(\boldsymbol{x}_{k_3}^m) = lbf(\boldsymbol{x}_{k_3}^m)$
18: **return** $k_3, sp(\boldsymbol{x}_{k_3}^m)$

Algorithm 2.2). After that, all the necessary information is given in order to apply the GradientTest to the k^{th}-coordinate of the generated subboxes (Algorithm 2.2, line 16).

The CoordinateToSplit routine (Algorithm 2.4) plays an important role in the AMIGO algorithm. This routine selects the coordinate which allows the GradientTest elimination rule to maximize the area to be rejected, if any. Otherwise, it applies Rule C of [19] as subdivision direction rule, i.e. the coordinate direction i is chosen for which $\mathrm{w}(\boldsymbol{g}_i(\boldsymbol{x})(x_i - c))$ is maximal. The CoordinateToSplit routine uses the EvalRejectArea procedure (Algorithm 2.5) to calculate the possible rejected area if the algorithm divides the current box perpendicularly to a given coordinate. EvalRejectArea returns the amount of search space that could be reduced from a box $\boldsymbol{x}$ by using the coordinate i. For this purpose a lower bound of $f(\boldsymbol{x}_i^m)$, $\tilde{f}$ and $\boldsymbol{g}_i(\boldsymbol{x})$ are used in equation (2.4) with $c_i = \mathrm{m}(\boldsymbol{x}_i)$.

The CoordinateToSplit routine analyzes the following possibilities:

1. Apply the EvalRejectArea procedure, (Algorithm 2.4, line 3) to evaluate the possible rejected areas using $\underline{f_c}(\boldsymbol{x}_i^m)$ as $lbf(\boldsymbol{x}_i^m)$, for each component $i = 1, \ldots, n$. Thus the computation of additional inclusion function evaluations ($\boldsymbol{f}(\boldsymbol{x}_i^m)\ i = 1 \ldots n$) can be avoided. If a reduction is feasible, the routine returns coordinate k_1 which maximizes the rejected area and $\underline{f_c}(\boldsymbol{x}_{k_1}^m)$ as $sp(\boldsymbol{x}_{k_1}^m)$ (Algorithm

Algorithm 2.5 EvalRejectArea Algorithm

Funct EvalRejectArea $(\boldsymbol{x}, i, lbf(\boldsymbol{x}_i^m), \tilde{f}, \boldsymbol{g}_i(\boldsymbol{x}))$

1: $RejectA = 0$ *Rejected Area*
2: $Vol(\boldsymbol{x}) = \prod_{i=1\ldots n} w(\boldsymbol{x}_i)$ *Volume of* $\boldsymbol{x}$
3: **if** ($lbf(\boldsymbol{x}_i^m) > \tilde{f}$)
4: **if** ($(\tilde{f} - lbf(\boldsymbol{x}_i^m))/\underline{\boldsymbol{g}_i}(\boldsymbol{x}) < w(\boldsymbol{x}_i)/2$)
5: $RejectA = (\tilde{f} - lbf(\boldsymbol{x}_i^m))/\underline{\boldsymbol{g}_i}(\boldsymbol{x}) \cdot Vol(\boldsymbol{x})/w(\boldsymbol{x}_i)$
6: **else**
7: $RejectA = Vol(\boldsymbol{x})/2$
8: **if** ($(lbf(\boldsymbol{x}_i^m) - \tilde{f})/\overline{\boldsymbol{g}_i}(\boldsymbol{x}) < w(\boldsymbol{x}_i)/2$)
9: $RejectA = RejectA + (lbf(\boldsymbol{x}_i^m) - \tilde{f})/\overline{\boldsymbol{g}_i}(\boldsymbol{x}) \cdot Vol(\boldsymbol{x})/w(\boldsymbol{x}_i)$
10: **else**
11: $RejectA = RejectA + Vol(\boldsymbol{x})/2$
12: **return** $RejectA$

2.4, line 7).

2. If the previous possibility does not work, the algorithm applies the EvalRejectArea procedure using $\underline{\boldsymbol{f}}(\boldsymbol{x}_i^m)$ instead of $\underline{\boldsymbol{f}_c}(\boldsymbol{x}_i^m)$ as $lbf(\boldsymbol{x}_i^m)$. This is motivated because the natural interval extension usually gives better bounds than the centered form for large intervals (see Section 2.1).

   In this case, for n-dimensional problems, n additional inclusion functions should be evaluated. To avoid such a large number of evaluations, the following heuristic is used: only the component $k_2 \in 1\ldots n$, with maximum $\{\underline{\boldsymbol{f}_c}(\boldsymbol{x}_i^m),\ i = 1\ldots n\}$ value (Algorithm 2.4, line 8) is evaluated, i.e, $\boldsymbol{f}(\boldsymbol{x}_{k_2}^m)$.

   Based on the studies about the convergence orders for the centered form and the natural interval extension [125, 107, 142], the natural interval extension $\boldsymbol{f}(\boldsymbol{x}_{k_2}^m)$ is evaluated only if for all its parent intervals the natural interval extension gives sharper results, i.e. $\mathrm{w}(\boldsymbol{f}(\boldsymbol{y}_{k_2}^m)) < \mathrm{w}(\boldsymbol{f}_c(\boldsymbol{y}_{k_2}^m)), \boldsymbol{x} \subseteq \boldsymbol{y}$. It means that for smaller intervals (when the centered form is better) no additional function evaluation is necessary and sharper bounds are obtained. The AMIGO algorithm takes this into account by using the $NaturalF(\boldsymbol{x})$ vector of flags, which is initiated to *True* in Algorithm 2.2, line 2. Boxes inherit this vector of flags from their father box (Algorithm 2.2, lines 11 and 14). The value of $NaturalF(\boldsymbol{x}_{k_2}^m)$ is updated in Algorithm 2.4, line 11.

   If the reduction of the interval $\boldsymbol{x}$ is possible, the CoordinateToSplit routine returns the coordinate k_2 and $\underline{\boldsymbol{f}}(\boldsymbol{x}_{k_2}^m)$ as $sp(\boldsymbol{x}_{k_2}^m)$ (Algorithm 2.4, line 15).

3. If the above possibilities do not work, no area will be rejected using equation (2.4) and the selected coordinate k_3 is given by Csendes and Ratz's Rule C (Algorithm 2.4, line 16) [19]. The Coordinate-ToSplit routine returns the coordinate k_3 and $\underline{\boldsymbol{f}_c}(\boldsymbol{x}_{k_3}^m)$ as $sp(\boldsymbol{x}_{k_3}^m)$ (Algorithm 2.4, line 15).

The AMIGO algorithm, as the traditional algorithm, uses a work list ($\mathcal{L}_W$) and a final list ($\mathcal{L}_S$). In the AMIGO algorithm, $\mathcal{L}_W$ and $\mathcal{L}_S$ store the following elements:

$$ListNode(\boldsymbol{x}) = \big(\boldsymbol{x}, lbzf(\boldsymbol{x}), sp(\boldsymbol{x}), \boldsymbol{f}(\mathrm{m}(\boldsymbol{x})), \boldsymbol{g}(\boldsymbol{x}), NaturalF(\boldsymbol{x})\big).$$

2.2.3 Numerical results

The new algorithm AMIGO has been numerically compared to the traditional method (see Remark 1.7) on a set of forty multi-dimensional test functions. This set of test functions has been taken from [19, 80, 128, 147, 150]. The reference where every function is described (*Ref*) and the dimension of the function (n) are shown for all the functions in Table 2.5 and 2.6.

Table 2.5: Efficiency comparison between the traditional and AMIGO algorithms.

Name	Ref	n	Eff_1	Eff_2	$SpUp$	ε
Schwefel 3.1	[128]	3	874	1340	0.7	10^{-8}
Price	[25]	2	5322	6023	0.9	10^{-8}
Schwefel 3.7	[150]	2	1762	1907	0.9	10^{-8}
Shekel 10	[128]	4	1365	1424	1.0	10^{-8}
Shekel 7	[128]	4	1365	1401	1.0	10^{-8}
Schwefel 2.1 (Beale)	[150]	2	5560	5350	1.0	10^{-8}
Levy 8	[128]	3	851	788	1.1	10^{-8}
Levy 5	[128]	2	1587	1466	1.1	10^{-8}
Schwefel 2.18 (Matyas)	[150]	2	5144	4595	1.1	10^{-8}
Levy 3	[128]	2	7116	6336	1.1	10^{-8}
EX1	[19]	2	488	429	1.1	10^{-8}
Chichinadze	[25]	2	653	547	1.2	10^{-8}
Schwefel 1.2	[150]	4	27975	22341	1.3	10^{-8}
Three-Hump-Camel-Back	[24]	2	2482	1957	1.3	10^{-8}
Six-Hump-Camel-Back	[147]	2	3484	2747	1.3	10^{-8}
Branin	[128]	2	4869	3742	1.3	10^{-8}
Rosenbrock 2	[24]	3	1279	972	1.3	10^{-8}
Goldstein-Price	[128]	2	41839	30493	1.4	10^{-8}
Schwefel 2.5 (Booth)	[150]	2	1993	1376	1.4	10^{-8}
Ratz 4	[128]	2	6792	4391	1.5	10^{-8}
Hartman 6	[128]	6	20996	12998	1.6	10^{-8}
Schwefel 3.2	[150]	3	3170	1951	1.6	10^{-8}
Henriksen-Madsen 3	[80]	2	14961	8892	1.7	10^{-8}
Treccani	[24]	2	2430	1439	1.7	10^{-8}
Griewank 2	[147]	2	1952	1113	1.8	10^{-8}
Rastrigin	[147]	2	1564	867	1.8	10^{-8}
Hartman 3	[128]	3	4463	2020	2.2	10^{-8}
Henriksen-Madsen 4	[80]	3	63639	28726	2.2	10^{-8}
Simplified Rosenbrock	[24]	2	2386	780	3.1	10^{-8}
Schwefel 2.14 (Powell)	[128]	4	387176	139335	2.8	10^{-5}
Schwefel 3.1p	[128]	3	1090616	640275	1.7	10^{-4}
Ratz 5	[128]	3	917495	364215	2.5	10^{-3}
Ratz 6	[128]	5	2162657	502237	4.3	10^{-3}
Ratz 7	[128]	7	3932091	645129	6.1	10^{-3}
Schwefel 2.10 (Kowalik)	[150]	4	672219	520749	1.3	10^{-2}
Griewank 10	[147]	10	3875828	2436103	1.6	10^{-2}
Rosenbrock 10	[24]	10	2708380	1524310	1.8	10^{-2}
Neumaier 2	[110]	4	898506	449367	2.0	10^{-2}
EX2	[19]	5	1010335	261241	3.9	10^{-2}
Ratz 8	[128]	9	388997	75231	5.2	10^{-2}
$\sum$			18282661	7716603	2.37	
Average $SpUp$					1.84	

Table 2.6: Time comparison between the traditional and AMIGO algorithms.

Name	Ref	n	T_1	T_2	$SpUp$	ε
Schwefel 3.1	[128]	3	0.01	0.01	1	10^{-8}
Price	[25]	2	0.04	0.06	0.67	10^{-8}
Schwefel 3.7	[150]	2	0.02	0.03	0.67	10^{-8}
Shekel 10	[128]	4	0.11	0.12	0.92	10^{-8}
Shekel 7	[128]	4	0.07	0.08	0.88	10^{-8}
Schwefel 2.1 (Beale)	[150]	2	0.05	0.06	0.83	10^{-8}
Levy 8	[128]	3	0.02	0.02	1	10^{-8}
Levy 5	[128]	2	0.06	0.06	1	10^{-8}
Schwefel 2.18 (Matyas)	[150]	2	0.02	0.03	0.67	10^{-8}
Levy 3	[128]	2	0.23	0.24	0.96	10^{-8}
EX1	[19]	2	0.00	0.00	1	10^{-8}
Chichinadze	[25]	2	0.01	0.01	1	10^{-8}
Schwefel 1.2	[150]	4	0.24	0.24	1	10^{-8}
Three-Hump-Camel-Back	[24]	2	0.02	0.02	1	10^{-8}
Six-Hump-Camel-Back	[147]	2	0.04	0.03	1.33	10^{-8}
Branin	[128]	2	0.04	0.04	1	10^{-8}
Rosenbrock 2	[24]	3	0.01	0.01	1	10^{-8}
Goldstein-Price	[128]	2	1.20	1.05	1.14	10^{-8}
Schwefel 2.5 (Booth)	[150]	2	0.01	0.01	1	10^{-8}
Ratz 4	[128]	2	0.05	0.05	1	10^{-8}
Hartman 6	[128]	6	1.31	0.93	1.41	10^{-8}
Schwefel 3.2	[150]	3	0.02	0.02	1	10^{-8}
Henriksen-Madsen 3	[80]	2	0.52	0.36	1.44	10^{-8}
Treccani	[24]	2	0.01	0.01	1	10^{-8}
Griewank 2	[147]	2	0.01	0.01	1	10^{-8}
Rastrigin	[147]	2	0.01	0.01	1	10^{-8}
Hartman 3	[128]	3	0.13	0.07	1.86	10^{-8}
Henriksen-Madsen 4	[80]	3	4.68	1.86	2.52	10^{-8}
Simplified Rosenbrock	[24]	2	0.01	0.01	1	10^{-8}
Schwefel 2.14 (Powell)	[128]	4	42.79	15.85	2.7	10^{-5}
Schwefel 3.1p	[128]	3	1635.83	1302.67	1.26	10^{-4}
Ratz 5	[128]	3	790.18	667.35	1.18	10^{-3}
Ratz 6	[128]	5	1626.63	823.49	1.98	10^{-3}
Ratz 7	[128]	7	1535.34	903.75	1.7	10^{-3}
Schwefel 2.10 (Kowalik)	[150]	4	364.05	405.54	0.9	10^{-2}
Griewank 10	[147]	10	299.85	334.04	0.9	10^{-2}
Rosenbrock 10	[24]	10	286.62	199.19	1.44	10^{-2}
Neumaier 2	[110]	4	290.41	84.41	3.44	10^{-2}
EX2	[19]	5	189.36	44.38	4.27	10^{-2}
Ratz 8	[128]	9	36.40	10.65	3.42	10^{-2}
$\sum$			7106.41	4796.77	1.48	
Average $SpUp$					1.36	

We have tested both algorithms with stopping criterion $w(\boldsymbol{x}) \leq \varepsilon = 10^{-8}$ and a limit for the runtime equal to one hour. For most of the functions both algorithms ended the execution, but for a set of eleven functions the traditional method was not able to finish. For these functions we show the values of ε for which both algorithms have finished the execution, although AMIGO was able to provide a solution with higher precision for functions Schwefel 2.14 ($\varepsilon = 10^{-6}$) and Ratz 8 ($\varepsilon = 10^{-3}$).

Table 2.5 shows the numerical comparison in evaluations between the two algorithms. If FE represents the number of interval function evaluations and GE the number of interval function evaluations of the gradient $\boldsymbol{g}(\boldsymbol{x})$, then columns Eff_1 and Eff_2 represent $FE + n \cdot GE$ for the traditional and AMIGO algorithm, respectively. Column $SpUp$ shows the values of $SpUp = Eff_1/Eff_2$, providing information on the relative speedup of the AMIGO algorithm compared to the traditional one. The last two rows show summary information about the achieved improvement. The first one sums up the required evaluations for every test function and shows the speedup for the whole set of test functions ($\sum Eff_1/\sum Eff_2$), while the last line shows the average of the speedups.

The value of $SpUp$ is less than one only for three out of 40 functions as can be seen in Table 2.5. In average (see the last row of Table 2.5) AMIGO is 1.84 times faster. In the second to last line of the table can be seen that the total speedup is 2.37. It is much better than the average speedup, which means that the new algorithm is better for harder problems.

Table 2.6 provides information about the CPU time spent on the optimization of the test functions in seconds. The columns T_1 and T_2 correspond to the CPU time in seconds of the traditional algorithm and AMIGO, respectively. The column $SpUp$ shows the speedup T_1/T_2. For easy problems, the algorithms were very fast so the difference is negligible. In these cases, the $SpUp$ is equal to 1. In the last two lines, the total and average speedup are given. The achieved improvement is not as much in time as in the evaluations due to the fact that in the new algorithm few additional computations and additional information storage were necessary. However, the algorithm AMIGO was 1.48 times faster for the total set of test functions.

As the numerical results show, the AMIGO algorithm improves the traditional method using the gradient information in an efficient way. The improvement is reached by pruning the search region smartly by exploiting the already available information. The effectiveness of the pruning process mainly depends on how good the upper bound of the minimum is, and on the sharpness of the derivative. In the cases where both are quite accurate, the effectiveness of the pruning can make AMIGO 4 to 6 times better than the traditional method, as e.g. for some of Ratz's functions.

2.2.4 Simplified multi-dimensional pruning method

Although the above pruning technique, and the AMIGO algorithm built around it, is very efficient, the storage of the gradient and support vectors for every box can increase the memory requirements significantly. In later studies, it was necessary to have a pruning method, which is easier to plug in the interval B&B method as a discarding test, without additional storage requirements. Thus, a simplified version of the above pruning method is created.

Let us suppose that for a box $\boldsymbol{x}$, the enclosure of its gradient vector, $\boldsymbol{g}(\boldsymbol{x})$, and the inclusion of its midpoint, $\boldsymbol{f}(\mathrm{m}(\boldsymbol{x}))$ are known. By Theorem 2.6, the interval

$$\boldsymbol{v} = \left\{ x \in \boldsymbol{x} : x_i \in \left[c_i - \frac{lbf(\boldsymbol{x}(i:c_i)) - \tilde{f}}{\overline{g_i}(\boldsymbol{x})}, c_i - \frac{lbf(\boldsymbol{x}(i:c_i)) - \tilde{f}}{\underline{g_i}(\boldsymbol{x})} \right] \right\}$$

cannot contain any global optimizer, and can be discarded. Here, c is the midpoint of $\boldsymbol{x}$, and $lbf(\boldsymbol{x}_i^m)) = \underline{f_c}(\boldsymbol{x}_i^m)$, thus the computation of $\boldsymbol{v}$ does not need any additional function evaluation. In this way one can choose, for pruning, the direction i for which $\boldsymbol{x} \setminus \boldsymbol{v}$ is the smallest, so that the largest possible pruning from the midpoint is made.

It is easy to see that this simplified pruning test (that only uses the midpoint and not the endpoints) does not need any additional function evaluation, nor additional memory storage, provided that $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{f}(\mathrm{m}(\boldsymbol{x}))$ are available.

2.3 Baumann Tent Pruning-Dividing Method

In minimization problems, interval B&B methods use a good upper bound of the global minimum value and good lower bounds of the objective function at the subproblems to discard many of them, but efficient pruning methods to discard regions of the subproblems that do not contain global minimizer points are also necessary to accelerate the convergence. The new pruning method presented in this section is based on the application of derivative information from the Baumann point.

For this section a new type of notation will be introduced. We denote either the lower or the upper bound of a box $\boldsymbol{x}$ at a same time by $\llcorner x \urcorner = \{\underline{x}, \overline{x}\}$ and $\llcorner \cdot \urcorner = \underline{\cdot}, \overline{\cdot}$ in general. It will be used in formulas, where by changing all occurrences of $\llcorner x \urcorner$ by $\underline{x}$ (or by $\overline{x}$) the statement is true.

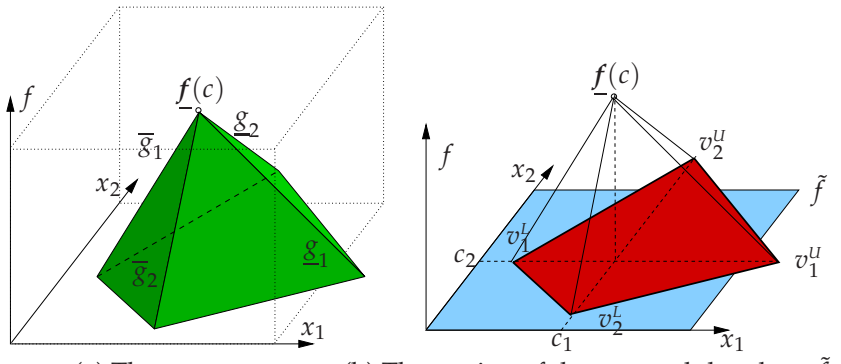
Let us recall the definition of the Baumann point [4], used by Baumann's optimal centered form (see Section 1.3.1). For a box $\boldsymbol{x}$ it is given by

$$b_i = \begin{cases} \dfrac{\overline{g}_i \underline{x}_i - \underline{g}_i \overline{x}_i}{\overline{g}_i - \underline{g}_i}, & \text{if } \underline{g}_i < 0 < \overline{g}_i \\ \overline{x}_i & \text{if } \overline{g}_i \leq 0 \\ \underline{x}_i & \text{if } \underline{g}_i \geq 0 \end{cases} \qquad i = 1, \ldots, n, \tag{2.11}$$

where $\boldsymbol{g}(\boldsymbol{x}) = ([\underline{g}_1, \overline{g}_1], \ldots, [\underline{g}_n, \overline{g}_n])$ is an inclusion for the gradient over box $\boldsymbol{x}$.

2.3.1 The two-dimensional case

Before explaining the general method, let us demonstrate it on a two-dimensional problem. Let us examine the visualization of the centered form for a box $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$. We suppose that $0 \in \boldsymbol{g}(\boldsymbol{x}) = ([\underline{g}_1(\boldsymbol{x}), \overline{g}_1(\boldsymbol{x})], [\underline{g}_2(\boldsymbol{x}), \overline{g}_2(\boldsymbol{x})])$ (abbreviated to $([\underline{g}_1, \overline{g}_1], [\underline{g}_2, \overline{g}_2])$), because otherwise the objective function is monotonous in the box and it could be discarded or reduced. A *tent* can be drawn from point $(c, \underline{f}(c))$ using the bounds of the derivatives (see Figure 2.4(a)). It can be seen that the function has to be above the tent. If we have a good upper bound $\tilde{f}$ on the global minimum which is smaller than $\underline{f}(c)$, we know that the minimum cannot be attained in the region defined by the intersection of the tent and the plane $\tilde{f}$. This region is drawn in dark-red in Figure 2.4(b), and will be denoted by *PR* (Pruneable Region). The use of this pruneable region was not recommended in [126] because the edges of *PR* are not parallel to the coordinate axes and the division of $\boldsymbol{x}$ by rejecting part of *PR* would generate a lot of boxes. Here is shown that *PR* is still of interest.



(a) The tent. (b) The section of the tent and the plane $\tilde{f}$.

Figure 2.4: The two-dimensional case.

PR can be defined by its vertices:

$$v_1^U = \left\{(x_1, c_2) \mid \tilde{f} = \underline{f}(c) + \underline{g}_1(x_1 - c_1)\right\} = \left(c_1 + \left(\tilde{f} - \underline{f}(c)\right)/\underline{g}_1, c_2\right), \tag{2.12}$$

$$v_1^L = \left(c_1 + \left(\tilde{f} - \underline{f}(c)\right)/\overline{g}_1, c_2\right), \tag{2.13}$$

$$v_2^U = \left(c_1, c_2 + \left(\tilde{f} - \underline{f}(c)\right)/\underline{g}_2\right), \tag{2.14}$$

$$v_2^L = \left(c_1, c_2 + \left(\tilde{f} - \underline{f}(c)\right)/\overline{g}_2\right), \tag{2.15}$$

if $\underline{g}_i \neq 0$ and $\overline{g}_i \neq 0, i \in \{1, 2\}$. Otherwise $v_i^L = -\infty$ or $v_i^U = \infty$ for the appropriate vertex, as the limits of the fractions suggest. Let us introduce the following notation to simplify the formulas:

$$\overline{pr}_i = \frac{\tilde{f} - \underline{f}(c)}{\underline{g}_i}, \qquad \underline{pr}_i = \frac{\tilde{f} - \underline{f}(c)}{\overline{g}_i}, \qquad i = 1, 2. \tag{2.16}$$

Thus $v_1^L = (c_1 + \underline{pr}_1, c_2)$, $v_1^U = (c_1 + \overline{pr}_1, c_2)$ and $v_2^L = (c_1, c_2 + \underline{pr}_2)$, $v_2^U = (c_1, c_2 + \overline{pr}_2)$ (notice that $\underline{pr}_1$ and $\underline{pr}_2$ are negative values).

Since our interval B&B algorithm works with boxes (with sides parallel to the axes), we cannot use other shapes (different from boxes) to divide the non-rejected area, i.e. the remaining region cannot be approximated in the way shown on the left hand side of Figure 2.5. On the other hand, if we approximate the non-rejected regions by boxes, too many boxes can be generated, and/or only a small part of the pruneable region can be discarded (see Figure 2.5). In general, more than four generated subboxes are not desired because the computational cost of the algorithm can increase accordingly.

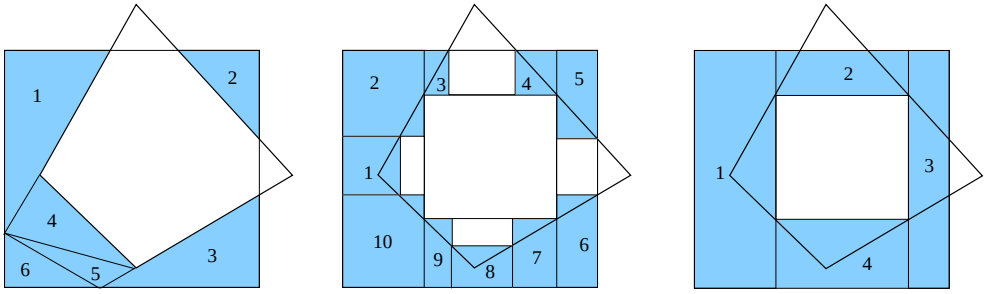


Figure 2.5: Examples of the different cutting choices.

One possible goal for pruning is to obtain the largest box to be removed from the original box. This can be done by computing the largest rectangle in the triangle defined by its vertices v_1^U, v_2^U, c (see Figure 2.6). The area of the rectangle is $A = a \cdot b$, where $b = \overline{pr}_2 - (\overline{pr}_2/\overline{pr}_1)a$ (notice that the

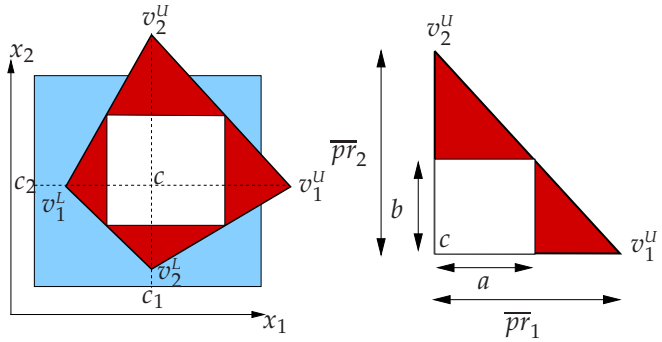


Figure 2.6: The largest box which can be cut from PR.

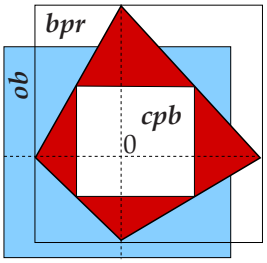


Figure 2.7: The new notation for the easier formulation.

coordinate geometrical form of the hypotenuse is $y = \overline{pr}_2 - (\overline{pr}_2/\overline{pr}_1)x, (x, y) \in \mathbb{R}^2_+$). Thus, the largest area can be computed by maximizing A with respect to a and b. The maximal area is $\overline{pr}_1\overline{pr}_2/4$ with $a = \overline{pr}_1/2$ and $b = \overline{pr}_2/2$. One can see that similar results can be obtained for all of the four triangles. Fortunately, every pair of adjacent rectangles share one edge, thus the resulting box can be given by $c + ([\underline{pr}_1/2, \overline{pr}_1/2], [\underline{pr}_2/2, \overline{pr}_2/2])$ (remember that $\underline{pr}_1$ and $\underline{pr}_2$ are negative).

Sometimes we can find better choices for pruning than the largest box in the pruneable region, since the latter is not always inside box $\boldsymbol{x}$. To calculate the pruneable box, the only important thing is the intersection between the pruneable region and the original box, independent of their position in the coordinate system. So, to obtain easier formulas and notation let us center the whole problem at the point c, i.e. let us change c to be the origin. Thus, we introduce the following notation.

$$\begin{array}{lrcl} \textbf{\textit{ob}} \text{ (Original Box):} & \textbf{\textit{ob}} & = & \boldsymbol{x} - c, \\ \textbf{\textit{bpr}} \text{ (Box containing } PR\text{):} & \textbf{\textit{bpr}} & = & ([\underline{pr}_1, \overline{pr}_1], [\underline{pr}_2, \overline{pr}_2]), \\ \textbf{\textit{pb}} \text{ (Pruneable Box):} & \textbf{\textit{pb}} & = & ([\underline{pb}_1, \overline{pb}_1], [\underline{pb}_2, \overline{pb}_2]), \end{array} \tag{2.17}$$

where ***bpr*** is the smallest box which contains the pruneable region (PR) (see Figure 2.7). In our new notation the Centered Pruneable Box (***cpb***) is

$$\textbf{\textit{cpb}} = ([\underline{pr}_1/2, \overline{pr}_1/2], [\underline{pr}_2/2, \overline{pr}_2/2]) \tag{2.18}$$

and the area of the ***cpb*** is

$$A(\textbf{\textit{cpb}}) = \left(\frac{\overline{pr}_1}{2} - \frac{\underline{pr}_1}{2}\right)\left(\frac{\overline{pr}_2}{2} - \frac{\underline{pr}_2}{2}\right) = \frac{1}{4}A(\textbf{\textit{bpr}}),$$

i.e. half of the area of PR, which is half of the area of ***bpr***.

2.3.1.1 Shifting the Centered Pruneable Box

As can be seen in Figure 2.8, sometimes it is better to shift ***cpb*** to the edge of ***ob***. This can improve the method by reducing the number of the generated subboxes at the cost of the reduction of the rejected area compared to the area of ***cpb***. Moreover, when ***cpb*** is not inside the original box, by shifting ***cpb*** the pruneable area can become larger (see the third case in Figure 2.8). To distinguish the centered pruneable box (***cpb***) from the shifted pruneable box, the latter will be denoted by ***spb***.

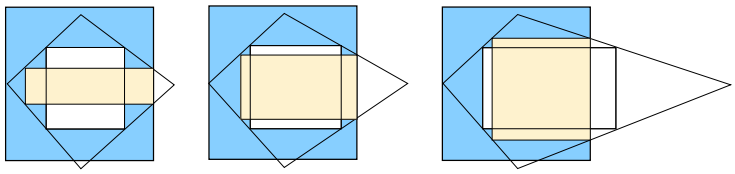


Figure 2.8: Shifting the pruneable box to an edge of ***ob***.

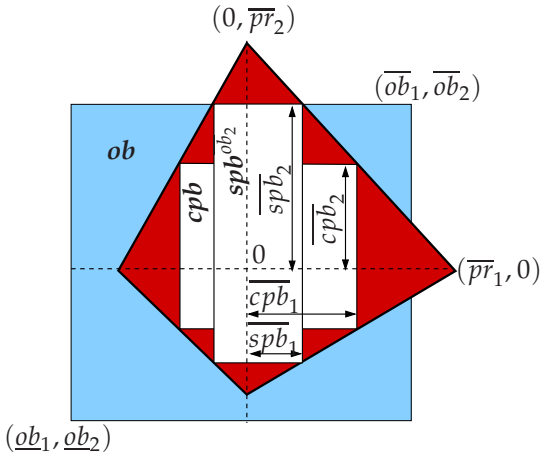


Figure 2.9: Shifting example.

Let us suppose that we want to shift ***cpb*** to the upper side of ***ob***, as $\boldsymbol{spb}^{\overline{ob}_2}$ in Figure 2.9. This is possible only if $\overline{ob}_2 < \overline{pr}_2$, and so $\overline{spb}_2 = \overline{ob}_2$ can be set. We only have to determine the box coordinates $(\underline{spb}_1, \overline{spb}_1, \underline{spb}_2, \overline{spb}_2)$ (note that $\underline{spb}_1, \underline{spb}_2$ are negative since the origin is inside the box). It is easy to see that the corner $(\overline{spb}_1, \overline{spb}_2)$ is on the edge of the pruneable region, iff

$$1 = \frac{\overline{spb}_1}{\overline{pr}_1} + \frac{\overline{spb}_2}{\overline{pr}_2}.$$

We know that $\overline{spb}_2 = \overline{ob}_2$, thus $\overline{spb}_1 = \overline{pr}_1 \left(1 - \frac{\overline{ob}_2}{\overline{pr}_2}\right)$. For the other corners,

$$1 = \frac{\overline{spb}_1}{\overline{pr}_1} + \frac{\underline{spb}_2}{\underline{pr}_2}, \qquad 1 = \frac{\underline{spb}_1}{\underline{pr}_1} + \frac{\overline{spb}_2}{\overline{pr}_2}, \qquad \text{and} \qquad 1 = \frac{\underline{spb}_1}{\underline{pr}_1} + \frac{\underline{spb}_2}{\underline{pr}_2}$$

have to hold, thus

$$\underline{spb}_1 = \underline{pr}_1 \left(1 - \frac{\overline{ob}_2}{\overline{pr}_2}\right), \quad \underline{spb}_2 = \underline{pr}_2 \frac{\overline{ob}_2}{\overline{pr}_2}.$$

Now we have a new problem to solve: which side should be chosen for the shifting, and moreover, is it worth shifting, or not? Both questions can be answered by comparing the areas that can be pruned for each case.

The areas of the boxes – supposing that the resulting ***spb*** is inside of the original one (***ob***) – can be computed as

$$\begin{aligned} A(\boldsymbol{spb}^{\overline{ob}_2}) &= \left(\overline{pr}_1 \left(1 - \frac{\overline{ob}_2}{\overline{pr}_2}\right) - \underline{pr}_1 \left(1 - \frac{\overline{ob}_2}{\overline{pr}_2}\right)\right) \left(\overline{ob}_2 - \underline{pr}_2 \frac{\overline{ob}_2}{\overline{pr}_2}\right) \\ &= \left(\overline{pr}_1 - \underline{pr}_1\right) \left(1 - \frac{\overline{ob}_2}{\overline{pr}_2}\right) \frac{\overline{ob}_2}{\overline{pr}_2} \left(\overline{pr}_2 - \underline{pr}_2\right) \\ &= \left(1 - \frac{\overline{ob}_2}{\overline{pr}_2}\right) \frac{\overline{ob}_2}{\overline{pr}_2} A(\boldsymbol{bpr}) \end{aligned} \tag{2.19}$$

that is, in general

$$A(\boldsymbol{spb}^{\llcorner ob \urcorner_i}) = \left(1 - \frac{\llcorner ob \urcorner_i}{\llcorner pr \urcorner_i}\right) \frac{\llcorner ob \urcorner_i}{\llcorner pr \urcorner_i} A(\boldsymbol{bpr}), \qquad i = 1, 2, \; \llcorner \cdot \urcorner = \underline{\cdot}, \overline{\cdot}. \tag{2.20}$$

If $\boldsymbol{spb}^{\llcorner ob \urcorner_i}$ is inside the original box then its area only depends on the value $\frac{\llcorner ob \urcorner_i}{\llcorner pr \urcorner_i}, i = 1, 2, \llcorner \cdot \urcorner = \underline{\cdot}, \overline{\cdot}$, obtaining a larger area if it is nearer $1/2$. If it equals $1/2$ we obtain that $A(\boldsymbol{spb}^{\llcorner ob \urcorner_i}) = A(\boldsymbol{cpb})$, which also means that $\boldsymbol{spb}^{\llcorner ob \urcorner_i} = \boldsymbol{cpb}$.

These equations show that by knowing the vector $\left(\frac{\overline{ob}_1}{\overline{pr}_1}, \frac{\underline{ob}_1}{\underline{pr}_1}, \frac{\overline{ob}_2}{\overline{pr}_2}, \frac{\underline{ob}_2}{\underline{pr}_2}\right)$ one can choose the largest Shifted Pruneable Box. If the largest $\boldsymbol{spb}$ is inside the original box we only have to decide which is better to prune, the largest $\boldsymbol{spb}$ or $\boldsymbol{cpb}$.

The other cases, when one or more corners of the box is inside the pruneable region (when no $\boldsymbol{spb}$ can be inside the box), do not differ too much, but have to be treated differently. This would lead to a case analysis that is out of the scope of this study and cannot be extended to the multi-dimensional case easily.

2.3.1.2 Simplification using the Baumann point

In this section we will see that the use of the Baumann point [4] instead of the center c simplifies our computations, avoiding the previous case analysis.

Theorem 2.9. *Consider a differentiable function $f : \mathbb{R}^2 \to \mathbb{R}$, and let us suppose that the inclusion function of its derivative, $\boldsymbol{g}$, can be computed over a given box $\boldsymbol{x}$. The pruneable region defined by its vertices, as in equations (2.12) to (2.15), centered in the Baumann point (i.e. $c = b$) includes either all the corners of $\boldsymbol{x}$ or none of them.*

Proof. First we show that if one corner is exactly on the edge of the pruneable region, then the other corners are also on the edges of the pruneable region (see Figure 2.10). That is, if $\underline{x}$ is such that

$$1 = \frac{\underline{x}_1 - b_1}{\underline{pr}_1} + \frac{\underline{x}_2 - b_2}{\underline{pr}_2}, \tag{2.21}$$

then

$$1 = \frac{\overline{x}_1 - b_1}{\overline{pr}_1} + \frac{\underline{x}_2 - b_2}{\underline{pr}_2}, \quad 1 = \frac{\underline{x}_1 - b_1}{\underline{pr}_1} + \frac{\overline{x}_2 - b_2}{\overline{pr}_2}, \quad 1 = \frac{\overline{x}_1 - b_1}{\overline{pr}_1} + \frac{\overline{x}_2 - b_2}{\overline{pr}_2}. \tag{2.22}$$

It is easy to see that equations (2.21-2.22) hold if and only if

$$\frac{\overline{x}_1 - b_1}{\overline{pr}_1} = \frac{\underline{x}_1 - b_1}{\underline{pr}_1}, \qquad \frac{\overline{x}_2 - b_2}{\overline{pr}_2} = \frac{\underline{x}_2 - b_2}{\underline{pr}_2}. \tag{2.23}$$

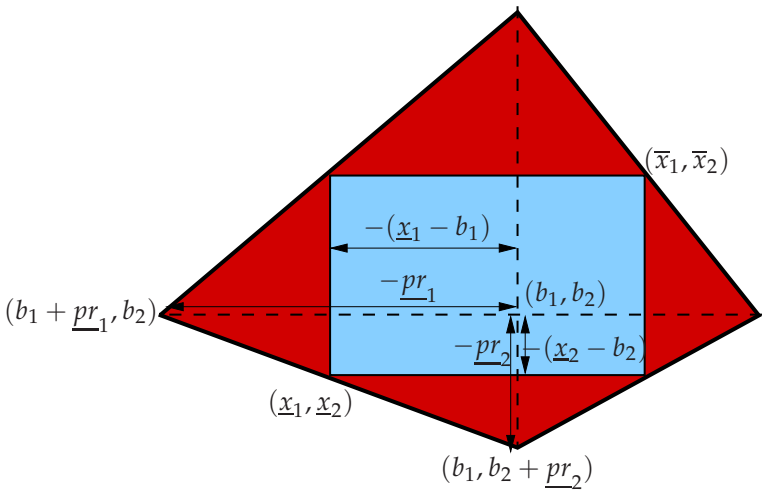


Figure 2.10: Example of the case when all the corners of the box $\boldsymbol{x}$ are on the edges of the *PR* using the Baumann point.

Recalling the definitions of $\overline{pr}_i, \underline{pr}_i, i = 1, 2$ (see (2.16)) and b_1, b_2 (see (2.11)), the equations are

$$\frac{\overline{x}_i - \frac{\overline{g}_i\underline{x}_i - \underline{g}_i\overline{x}_i}{\overline{g}_i - \underline{g}_i}}{\frac{\tilde{f} - \underline{F}(b)}{\underline{g}_i}} = \frac{\underline{x}_i - \frac{\overline{g}_i\underline{x}_i - \underline{g}_i\overline{x}_i}{\overline{g}_i - \underline{g}_i}}{\frac{\tilde{f} - \underline{F}(b)}{\overline{g}_i}}, \quad i = 1, 2$$

which can be simplified to

$$\underline{g}_i\overline{x}_i - \underline{g}_i\frac{\overline{g}_i\underline{x}_i - \underline{g}_i\overline{x}_i}{\overline{g}_i - \underline{g}_i} = \overline{g}_i\underline{x}_i - \overline{g}_i\frac{\overline{g}_i\underline{x}_i - \underline{g}_i\overline{x}_i}{\overline{g}_i - \underline{g}_i}, \quad i = 1, 2,$$

and finally to

$$\underline{g}_i\overline{x}_i - \overline{g}_i\underline{x}_i = \frac{-\overline{g}_i(\overline{g}_i\underline{x}_i - \underline{g}_i\overline{x}_i) + \underline{g}_i(\overline{g}_i\underline{x}_i - \underline{g}_i\overline{x}_i)}{\overline{g}_i - \underline{g}_i}, \quad i = 1, 2,$$

which reduces to an identity. Consequently, the equations (2.23) are always true. Thus, if we change the equality in (2.21) to inequality, then the same type of inequality will occur in (2.22), which proves the theorem. □

Remark 2.10. The theorem also holds if $\underline{g}_i = 0$ or $\overline{g}_i = 0$ for $i \in \{1, 2\}$. In this case either $\overline{pr}_i = \infty$ and $b_i = \underline{x}_i$ or $\underline{pr}_i = -\infty$ and $b_i = \overline{x}_i$. Thus,

$$\frac{\overline{x}_i - b_i}{\overline{pr}_i} = \frac{\underline{x}_i - b_i}{\underline{pr}_i} = 0.$$

Corollary 2.11. *Since* $\frac{\overline{x}_i - b_i}{\overline{pr}_i} = \frac{\underline{x}_i - b_i}{\underline{pr}_i}$ $\forall i$, *it can be deduced that the shifted pruneable box* $\boldsymbol{spb}^{\overline{ob}_i}$ *is exactly the same as* $\boldsymbol{spb}^{\underline{ob}_i}$ *(see Figure 2.11). Thus, it can be denoted as* $\boldsymbol{spb}^i$. *Therefore, after shifting there will be only two newly generated boxes (the enclosures of the non-rejected areas), and only the better shifting direction has to be determined.*

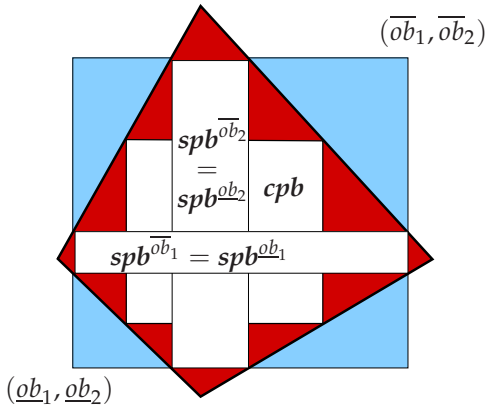


Figure 2.11: Using the Baumann point, the shifted pruneable box $\boldsymbol{spb}^{\overline{ob}_i}$ is exactly the same as $\boldsymbol{spb}^{\underline{ob}_i}$, thus denoted by $\boldsymbol{spb}^i$.

Remark 2.12. The above results suggest the notation of the Shifting Factor

$$sf_i = \frac{\overline{x}_i - b_i}{\overline{pr}_i}\left(= \frac{\underline{x}_i - b_i}{\underline{pr}_i}\right), \text{i.e. } sf_i = \frac{\overline{ob}_i}{\overline{pr}_i}\left(= \frac{\underline{ob}_i}{\underline{pr}_i}\right), \quad i = 1, 2,$$

and the Opposite Shifting Factor

$$osf_i = 1 - \frac{\overline{x}_i - b_i}{\overline{pr}_i}, \quad \text{i.e.} \quad osf_i = 1 - \frac{\overline{ob}_i}{\overline{pr}_i}, \quad i = 1, 2.$$

Therefore, $A(\boldsymbol{spb}^i) = sf_i osf_i A(\boldsymbol{bpr})$.

Theorem 2.13. *If $\boldsymbol{cpb} \not\subseteq \boldsymbol{ob}$ and $\boldsymbol{ob} \not\subseteq \boldsymbol{cpb}$, then there exists an $\boldsymbol{spb}$ such that $A(\boldsymbol{spb}) > A(\boldsymbol{cpb} \cap \boldsymbol{ob})$.*

Proof. If $\boldsymbol{cpb} \not\subseteq \boldsymbol{ob}$, then there exists an $i \in \{1, 2\}$ such that $sf_i < 1/2$, i.e. in the ith direction $\overline{ob}_i < \overline{pr}_i/2 = \overline{cpb}_i$ thus $\underline{cpb}_i < \underline{ob}_i$. As $\boldsymbol{ob} \not\subseteq \boldsymbol{cpb}$, for the other direction $j \neq i$, $\overline{cpb}_j < \overline{ob}_j$ and $\underline{cpb}_j > \underline{ob}_j$. The area of the region, which can be pruned using the $\boldsymbol{cpb}$ is

$$A(\boldsymbol{cpb} \cap \boldsymbol{ob}) = \frac{1}{2} \frac{\overline{ob}_i - \underline{ob}_i}{\overline{pr}_i - \underline{pr}_i} A(\boldsymbol{bpr}),$$

while the area of $\boldsymbol{spb}^i$ is

$$A(\boldsymbol{spb}^i) = sf_i osf_i A(\boldsymbol{bpr}).$$

One can easily see that

$$\begin{aligned} A(\boldsymbol{cpb} \cap \boldsymbol{ob}) < A(\boldsymbol{spb}^i) \quad &\iff \frac{1}{2} \frac{\overline{ob}_i - \underline{ob}_i}{\overline{pr}_i - \underline{pr}_i} < sf_i osf_i \\ \iff \frac{1}{sf_i} \frac{1}{2} \frac{\overline{ob}_i - \underline{ob}_i}{\overline{pr}_i - \underline{pr}_i} < osf_i \quad &\iff \frac{1}{2} \frac{\frac{\overline{ob}_i}{sf_i} - \frac{\underline{ob}_i}{sf_i}}{\overline{pr}_i - \underline{pr}_i} < osf_i \\ \iff \frac{1}{2} \frac{\overline{pr}_i - \underline{pr}_i}{\overline{pr}_i - \underline{pr}_i} < osf_i \quad &\iff \frac{1}{2} < osf_i \end{aligned}$$

which is assured by $sf_i < 1/2$. □

Remark 2.14. In Theorem 2.13 if $\boldsymbol{spb}^i \not\subseteq \boldsymbol{ob}$, then $\boldsymbol{ob} \subset \boldsymbol{spb}^i$. Therefore, the whole box can be deleted. It also means that although the computed area ($A(\boldsymbol{spb}^i)$) is larger than the possible pruned one, we still prune the maximal area, $A(\boldsymbol{cpb} \cap \boldsymbol{ob}) < A(\boldsymbol{ob}) = A(\boldsymbol{spb}^i \cap \boldsymbol{ob}) < A(\boldsymbol{spb}^i)$.

The advantages of the usage of the Baumann point are that it makes our computation easier, and it balances the pruneable region over the box.

2.3.2 The n-dimensional case

To generalize the above results for the multi-dimensional case, let us suppose that $c = (c_1, c_2, \ldots, c_n)$ is an arbitrary point of the n-dimensional box $\boldsymbol{x}$. Before any computations are done, let us center the problem at point c as we did in the two-dimensional case. Thus, we will use the same notation:

$$\boldsymbol{ob} \text{ (Original Box):} \quad \boldsymbol{ob} \quad = \quad \boldsymbol{x} - c, \tag{2.24}$$

It is easy to see that the vertices of the pruneable region are

$$\begin{aligned} v_i^L &= (0, \ldots, 0, \underline{pr}_i, 0, \ldots, 0), \\ v_i^U &= (0, \ldots, 0, \overline{pr}_i, 0, \ldots, 0), \quad i = 1, \ldots, n, \end{aligned} \tag{2.25}$$

where

$$\overline{pr}_i = \frac{\tilde{f} - \underline{f}(c)}{\underline{g}_i}, \quad \underline{pr}_i = \frac{\tilde{f} - \underline{f}(c)}{\overline{g}_i}, i = 1, \ldots, n.$$

From (2.25) we know that in the three-dimensional case the shape of PR is as it is shown in Figure 2.12. To see the properties of this body, let us take a little side-track into geometry. Let us first introduce the notion of orthopeder which appears to be a new class of geometrical objects.

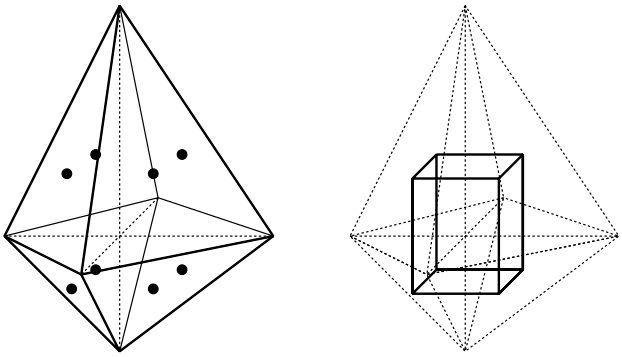


Figure 2.12: A pruneable region (an orthopeder) in the three-dimensional case with the largest cuboid inside.

Definition 2.15. *An n-dimensional polytope is called* orthopeder, *if the diagonals intersect in one point and are orthogonal to each other.*

Proposition 2.16. *The PR defined by its vertices (2.25) is an orthopeder.*

Proof. The diagonals of the *PR* are the lines between $v_i^L, v_i^U, i = 1, \ldots, n$, i.e. between the non-adjacent vertices (2.25). As all the lines pass through the origin, this is the intersection point. The ith diagonal v_i^L, v_i^U is parallel to the ith axis for all i, thus all the diagonals are orthogonal. □

Definition 2.17. *The* cross polytope *is the regular polytope in n dimensions corresponding to the convex hull of the points formed by permuting the coordinates* $(\pm 1, 0, 0, \ldots, 0)$.

An orthopeder is a generalization of the cross polytope, with different scaling on all directions (the corresponding scaling vectors are $-\underline{pr}$ and $\overline{pr}$). The cross polytope is the dual of the n-dimensional hypercube, i.e. its vertices are the centers of the facets of the hypercube $[-1, 1]^n$. As is known in geometry, the largest hyperrectangle within the cross polytope is its dual, i.e. the hypercube which has its vertices on the center of the faces of the cross polytope.

Proposition 2.18. *In an orthopeder (a scaled cross polytope) the largest hyperrectangle is the hypercube (inside the cross polytope) scaled with the scale vectors of the orthopeder.*

Proof. Let us suppose that the scale vectors of the orthopeder are $-\underline{pr}$ and $\overline{pr}$. The largest hyperrectangle inside the orthopeder is a $\boldsymbol{pb}$ which is the optimum of

$$\begin{aligned} \max \quad & V(\boldsymbol{pb}) = \prod_{i=1,\ldots,n} \overline{pb}_i - \underline{pb}_i \\ \text{s.t.} \quad & 1 = \sum_{i=1,\ldots,n} \frac{J_i(\boldsymbol{pb}_i)}{J_i(\boldsymbol{bpr}_i)} \qquad J_i(\cdot) = \underline{\cdot}, \overline{\cdot}, \quad i = 1, \ldots, n. \end{aligned}$$

The conditions hold only if $\frac{\overline{pb}_i}{\overline{pr}_i} = \frac{\underline{pb}_i}{\underline{pr}_i}\ \forall i$. Thus, $\overline{pb}_i - \underline{pb}_i = \frac{\underline{pb}_i}{\underline{pr}_i}(\overline{pr}_i - \underline{pr}_i)$, and so denoting $x_i = \frac{\underline{pb}_i}{\underline{pr}_i}$, the above problem can be converted to

$$\begin{aligned} \max \quad & \prod_{i=1,\ldots,n} x_i \prod_{i=1,\ldots,n} \overline{pr}_i - \underline{pr}_i \\ \text{s.t.} \quad & 1 = \sum_{i=1,\ldots,n} x_i, \end{aligned}$$

where the solution is $x_i = \frac{1}{n}$, $\forall i$. Therefore, $\underline{pb}_i = \frac{\underline{pr}_i}{n}, \overline{pb}_i = \frac{\overline{pr}_i}{n}$, $\forall i$, which is what we wanted to prove. □

Corollary 2.19. *The largest box in the orthopeder PR defined by its vertices (2.25) is the Centered Pruneable Box (**cpb**),*

$$\boldsymbol{cpb} = (\left[\underline{pr}_1/n, \overline{pr}_1/n\right], \ldots, \left[\underline{pr}_n/n, \overline{pr}_n/n\right]).$$

Corollary 2.20. *The volume of **cpb** is*

$$V(\boldsymbol{cpb}) = \prod_{i=1,..,n} \frac{\overline{pr}_i - \underline{pr}_i}{n} = \frac{1}{n^n} V(\boldsymbol{bpr}).$$

Theorem 2.21. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a given differentiable function, **f**, its inclusion function and **g** the inclusion function of its gradient over a given box **x**. The pruneable region defined by its vertices as in (2.25) centered in the Baumann point (i.e. $c = b$) includes either all the corners of **x** or none of them.*

Proof. It can be proved in the same way as Theorem 2.9. We first prove that when one corner of $\boldsymbol{x}$ is exactly on one facet of the pruneable region, the other corners are also on the corresponding facets of the pruneable region. For this we will use the notation $J^k(\cdot) = (J_1^k(\cdot), \ldots, J_n^k(\cdot))$ where $J_i^k(\cdot) \in \{\underline{\cdot}, \overline{\cdot}\}, i = 1, \ldots, n$, and so the kth vertex of $\boldsymbol{x}$ can be written as $w_k = (J_1^k(\boldsymbol{x}_1), \ldots, J_n^k(\boldsymbol{x}_n)), k = 1, \ldots, 2^n$.

Now we will show that if w_k is on the facet of PR, i.e.

$$1 = \sum_{i=1}^{n} \frac{J_i^k(\boldsymbol{x}_i) - b_i}{J_i^k(\boldsymbol{bpr}_i)}, \tag{2.26}$$

then it is true for the other corners,

$$1 = \sum_{i=1}^{n} \frac{J_i^j(\boldsymbol{x}_i) - b_i}{J_i^j(\boldsymbol{bpr}_i)}, \qquad \forall j = 1, \ldots, 2^n. \tag{2.27}$$

It is easy to see that equations (2.26)-(2.27) can hold iff

$$\frac{J_i^k(\boldsymbol{x}_i) - b_i}{J_i^k(\boldsymbol{bpr}_i)} = \frac{J_i^j(\boldsymbol{x}_i) - b_i}{J_i^j(\boldsymbol{bpr}_i)} \qquad \forall j = 1, \ldots, 2^n;\ i = 1, \ldots, n,$$

i.e.

$$\frac{\overline{x}_i - b_i}{\overline{pr}_i} = \frac{\underline{x}_i - b_i}{\underline{pr}_i}, \qquad i = 1, \ldots, n. \tag{2.28}$$

As the Baumann point is defined piecewise, the proof of equations (2.28) does not differ from the demonstration of equations (2.23) which appears in the proof of Theorem 2.9.

The equations (2.28) are always true. Thus, changing the equality in (2.26) to inequality implies the same inequality change in (2.27). It means that when one vertex of $\boldsymbol{x}$ is inside (or outside) PR, the other vertices are also inside (or outside), which proves the theorem. □

2.3.2.1 Shifting the n-dimensional *cpb*

When ***cpb*** is inside ***ob*** and ***cpb*** is pruned, the number of generated boxes is $2n$. If $n > 3$ the performance of the algorithm may worsen due to the large number of boxes to be evaluated. Therefore, it is important to find the best direction(s) to shift.

To choose the best ***pb***, we are going to construct a simple and powerful method that determines the ***spb*** with the *largest volume/fewer new boxes* index. We already know that by using the Baumann point the equations

$$\frac{\overline{ob}_i}{\overline{pr}_i} = \frac{\underline{ob}_i}{\underline{pr}_i},\ i = 1, \ldots, n$$

hold (which are equivalent to (2.28)). In the following formulas these values will appear many times, thus, let us introduce the notation of Shifting Factor (sf_i) and Opposite Shifting Factor (osf_i) as we did in the 2D case,

$$sf_i = \frac{\overline{ob}_i}{\overline{pr}_i} = \frac{\underline{ob}_i}{\underline{pr}_i}, \quad osf_i = 1 - sf_i, \quad i = 1, \dots, n.$$

Shifting in the dimension i is possible only if $sf_i < 1$ ($osf_i > 0$), i.e. when $\underline{ob}_i > \underline{pr}_i$ and $\overline{ob}_i < \overline{pr}_i$. The corner $(\overline{spb}_1, \dots, \overline{spb}_n)$ is on the corresponding facet of the pruneable region, iff

$$1 = \frac{\overline{spb}_1}{\overline{pr}_1} + \frac{\overline{spb}_2}{\overline{pr}_2} + \dots + \frac{\overline{spb}_n}{\overline{pr}_n}.$$

Let us suppose that $\overline{spb}_j = \overline{ob}_j$ (and so $\underline{spb}_j = \underline{ob}_j$), i.e. we shift in the jth dimension. Therefore, if the other coordinates of ***cpb*** have to be scaled with the same factor r, i.e. $\lfloor spb \rceil_i = r \cdot \lfloor cpb \rceil_i = r\frac{\overline{pr}_i}{n}$, $i = 1, \dots, n, i \neq j, \lfloor \cdot \rceil = \underline{\cdot}, \overline{\cdot}$, then

$$\begin{aligned} 1 &= \frac{\overline{ob}_j}{\overline{pr}_j} + \sum_{\substack{i=1,\dots,n \\ i \neq j}} \frac{r\frac{\overline{pr}_i}{n}}{\overline{pr}_i}, \qquad \text{where} \qquad r = \frac{n}{n-1}\left(1 - sf_j\right) = \frac{n}{n-1} osf_j, \qquad \text{and so} \\ \lfloor spb \rceil_i &= \frac{\lfloor pr \rceil_i}{n-1} osf_j, \qquad i = 1, \dots, n \quad \lfloor \cdot \rceil = \underline{\cdot}, \overline{\cdot}. \end{aligned}$$

Thus, if $\boldsymbol{spb}^j$ is inside ***ob***, its volume can be computed as

$$\begin{aligned} V(\boldsymbol{spb}^j) &= \left(\overline{ob}_j - \underline{ob}_j\right) \prod_{\substack{i=1,\dots,n \\ i \neq j}} \left(\frac{\overline{pr}_i}{n-1} osf_j - \frac{\underline{pr}_i}{n-1} osf_j\right) \\ &= \left(\overline{ob}_j - \frac{\underline{pr}_j}{\overline{pr}_j}\overline{ob}_j\right) \left(\frac{n}{n-1} osf_j\right)^{n-1} \prod_{\substack{i=1,\dots,n \\ i \neq j}} \frac{\overline{pr}_i - \underline{pr}_i}{n} \\ &= n \left(\frac{n}{n-1}\right)^{n-1} \frac{\overline{ob}_j}{\overline{pr}_j} osf_j^{n-1} \prod_{i=1,\dots,n} \frac{\overline{pr}_i - \underline{pr}_i}{n} \\ &= \frac{n^n}{(n-1)^{n-1}} sf_j \, osf_j^{n-1} \, V(\boldsymbol{cpb}). \end{aligned}$$

One can see that, the nearer sf_j is $1/n$, the greater the volume is. It also implies that by knowing the vector $(sf_1, \dots, sf_n)$ we can easily choose the best direction to shift.

In some cases it is possible to shift in several dimensions. To shift in $j_1, \dots, j_k \in \{1, \dots, n\}$ dimensions, we generalize the notation of the Opposite Shifting Factor: $osf_{j_1,\dots,j_k} = 1 - \sum_{i=j_1,\dots j_k} sf_i$. Shifting is possible only if $osf_{j_1,\dots,j_k} > 0$. Thus, we can compute the coordinates of $\boldsymbol{spb}^{j_1,\dots,j_k}$ in the following way:

$$\begin{aligned} 1 &= \sum_{i=j_1,\dots j_k} \frac{\overline{ob}_i}{\overline{pr}_i} + \sum_{\substack{i=1,\dots,n \\ i \neq j_1,\dots,j_k}} \frac{r\frac{\overline{pr}_i}{n}}{\overline{pr}_i}, \qquad r = \frac{n}{n-k}\left(1 - \sum_{i=j_1,\dots j_k} sf_i\right) = \frac{n}{n-k} osf_{j_1,\dots,j_k}, \\ \lfloor spb \rceil_i &= \frac{\lfloor pr \rceil_i}{n-k} osf_{j_1,\dots,j_k}, \qquad i = 1, \dots, n \quad \lfloor \cdot \rceil = \underline{\cdot}, \overline{\cdot}. \end{aligned} \tag{2.29}$$

If $\boldsymbol{spb}^{j_1,\ldots,j_k}$ is inside $\boldsymbol{ob}$, the pruneable volume can be computed as

$$\begin{aligned} V(\boldsymbol{spb}^{j_1,\ldots,j_k}) &= \prod_{i=j_1,\ldots j_k}\left(\overline{ob}_i - \underline{ob}_i\right)\prod_{\substack{i=1,\ldots,n\\ i\neq j_1,\ldots,j_k}}\left(\frac{\overline{pr}_i}{n-k}\,osf_{j_1,\ldots,j_k} - \frac{\underline{pr}_i}{n-k}\,osf_{j_1,\ldots,j_k}\right)\\ &= \prod_{i=j_1,\ldots,j_k}\left(\overline{ob}_i - \frac{\underline{pr}_i}{\overline{pr}_i}\overline{ob}_i\right)\left(\frac{n}{n-k}osf_{j_1,\ldots,j_k}\right)^{n-k}\prod_{\substack{i=1,\ldots,n\\ i\neq j_1,\ldots,j_k}}\frac{\overline{pr}_i - \underline{pr}_i}{n}\\ &= n^k\prod_{i=j_1,\ldots,j_k}\frac{\overline{ob}_i}{\overline{pr}_i}\left(\frac{n}{n-k}\right)^{n-k}osf_{j_1,\ldots,j_k}^{n-k}\prod_{i=1,\ldots,n}\left(\overline{pr}_i - \underline{pr}_i\right)/n\\ &= \frac{n^n}{(n-k)^{n-k}}\left(\prod_{i=j_1,\ldots,j_k}sf_i\right)osf_{j_1,\ldots,j_k}^{n-k}V(\boldsymbol{cpb}). \end{aligned}$$

Remark 2.22. If we have the volume for $\boldsymbol{spb}$, shifted in the dimensions $j_1,\ldots,j_{k-1}\in\{1,\ldots,n\}$, and we want to shift it in one more dimension, $j_k\in\{1,\ldots,n\}$ as well, the new volume can be computed from the known $V(\boldsymbol{spb}^{j_1,\ldots,j_{k-1}})$ in the following way:

$$V(\boldsymbol{spb}^{j_1,\ldots,j_k}) = sf_{j_k}\frac{(n-k+1)^{n-k+1}}{(n-k)^{n-k}}\,\frac{osf_{j_1,\ldots,j_k}^{n-k}}{osf_{j_1,\ldots,j_{k-1}}^{n-k+1}}V(\boldsymbol{spb}^{j_1,\ldots,j_{k-1}}).$$

Similar to Theorem 2.13, Theorem 2.23 shows that if $\boldsymbol{cpb}$ is not inside $\boldsymbol{ob}$ we can always obtain an $\boldsymbol{spb}$ with larger volume than $\boldsymbol{cpb}\cap\boldsymbol{ob}$.

Theorem 2.23. *Let us suppose that $\boldsymbol{cpb}\not\subseteq\boldsymbol{ob}$ such that $\boldsymbol{cpb}_i\not\subseteq\boldsymbol{ob}_i$ exactly for the dimensions $i\in\{j_1,\ldots,j_k\}\subseteq\{1,\ldots,n\}$, i.e. $\overline{ob}_i<\overline{cpb}_i$ (and so $\underline{ob}_i>\underline{cpb}_i$) for all $i\in\{j_1,\ldots,j_k\}$, but only for those dimensions. Then $V(\boldsymbol{spb}^{j_1}\cap\boldsymbol{ob})>V(\boldsymbol{cpb}\cap\boldsymbol{ob})$, and $V(\boldsymbol{spb}^{j_1,\ldots,j_l})>V(\boldsymbol{spb}^{j_1,\ldots,j_{l-1}}\cap\boldsymbol{ob})$, $l\leq k$.*

Proof. First we will prove that $V(\boldsymbol{spb}^{j_1}\cap\boldsymbol{ob})>V(\boldsymbol{cpb}\cap\boldsymbol{ob})$. From the assumptions one can obtain that $sf_{j_1}<1/n$, and so $osf_{j_1}>(n-1)/n$.

The volume of the region which can be pruned using $\boldsymbol{cpb}$ is

$$V(\boldsymbol{cpb}\cap\boldsymbol{ob}) = \frac{1}{n^{n-k}}\prod_{i=j_1,\ldots,j_k}\frac{\overline{ob}_i - \underline{ob}_i}{\overline{pr}_i - \underline{pr}_i}\,V(\boldsymbol{bpr}).$$

The volume of the region which can be pruned using $\boldsymbol{spb}^{j_1}$ is

$$\begin{aligned} V(\boldsymbol{spb}^{j_1}\cap\boldsymbol{ob}) &= \prod_{\substack{i=1,\ldots,n\\ i\neq j_1,\ldots,j_k}}\frac{osf_{j_1}}{n-1}\left(\overline{pr}_i - \underline{pr}_i\right)\prod_{i=j_1,\ldots,j_k}(\overline{ob}_i - \underline{ob}_i)\\ &= \frac{osf_{j_1}^{n-k}}{(n-1)^{n-k}}\prod_{i=j_1,\ldots,j_k}\frac{\overline{ob}_i - \underline{ob}_i}{\overline{pr}_i - \underline{pr}_i}\,V(\boldsymbol{bpr}). \end{aligned}$$

Thus,

$$\begin{aligned} V(\boldsymbol{cpb}\cap\boldsymbol{ob}) < V(\boldsymbol{spb}^{j_1}\cap\boldsymbol{ob}) &\iff \frac{1}{n^{n-k}} < \frac{1}{(n-1)^{n-k}}\,osf_{j_1}^{n-k}\\ &\iff \left(\frac{n-1}{n}\right)^{n-k} < osf_{j_1}^{n-k} \iff \frac{n-1}{n} < osf_{j_1}, \end{aligned}$$

which is assured by $sf_{j_1}<1/n$.

Let us suppose now, that we have already shifted in $j_1, \ldots, j_{l-1}$ directions, but $\boldsymbol{spb}^{j_1,\ldots,j_{l-1}} \not\subseteq \boldsymbol{ob}$. We know that

$$V(\boldsymbol{spb}^{j_1,\ldots,j_l} \cap \boldsymbol{ob}) = \frac{osf_{j_1,\ldots,j_l}^{n-k}}{(n-l)^{n-k}} \prod_{i=j_1,\ldots,j_k} \frac{\overline{ob}_i - \underline{ob}_i}{\overline{pr}_i - \underline{pr}_i} V(\boldsymbol{bpr}).$$

Then

$$\begin{aligned}
V(\boldsymbol{spb}^{j_1,\ldots,j_{l-1}} \cap \boldsymbol{ob}) &< V(\boldsymbol{spb}^{j_1,\ldots,j_l} \cap \boldsymbol{ob}) \\
\iff \frac{osf_{j_1,\ldots,j_{l-1}}^{n-k}}{(n-l+1)^{n-k}} &< \frac{osf_{j_1,\ldots,j_l}^{n-k}}{(n-l)^{n-k}} \\
\iff (n-l)osf_{j_1,\ldots,j_{l-1}} &< (n-l+1)osf_{j_1,\ldots,j_l} \\
\iff (n-l)osf_{j_1,\ldots,j_{l-1}} &< (n-l+1)(osf_{j_1,\ldots,j_{l-1}} - sf_{j_l}) \\
\iff (n-l+1)sf_{j_l} &< osf_{j_1,\ldots,j_{l-1}} \\
\iff (n-l+1)sf_{j_l} + \sum_{i=j_1,\ldots,j_{l-1}} sf_i &< 1.
\end{aligned}$$

Since $sf_i < 1/n$, $i = j_1, \ldots, j_k$, the above equation holds. Thus, $V(\boldsymbol{spb}^{j_1,\ldots,j_{l-1}} \cap \boldsymbol{ob}) < V(\boldsymbol{spb}^{j_1,\ldots,j_l} \cap \boldsymbol{ob})$ whenever $l \leq k$. □

Corollary 2.24. *The largest pruneable box inside* $\boldsymbol{ob}$ *is* $\boldsymbol{spb}^{j_1,\ldots,j_k}$ *if* $\overline{ob}_i < \overline{cpb}_i$ *for all* $i \in \{j_1, \ldots, j_k\}$*, and only for those dimensions.*

2.3.3 Integrating the BTPD method into the interval B&B algorithm

The Baumann Tent Pruning Dividing (BTPD) method is described in Algorithm 2.6, and can be incorporated into Algorithm 1.1 by changing the Division Rule in line 6 by a call to Algorithm 2.6.

The BTPD method prunes and divides the current box or just divides it depending on several factors. Firstly, a Pruning Index (PI) of the best $\boldsymbol{pb}$, i.e. the *relative volume/number of new boxes* is calculated. If $PI \cdot n^n$ is smaller than a given input parameter α, the usual division rule is applied, avoiding the unpromising pruning of $\boldsymbol{pb}$ (see Algorithm 2.6 line 12). The multiplication by n^n comes from the fact that the volume of $\boldsymbol{cpb}$ is n^n-part of the volume of $\boldsymbol{bpr}$, thus, to give a shifted $\boldsymbol{pb}$ a chance to be pruned, we multiply the Pruning Index by n^n. Secondly, if the box is badly shaped, i.e. the ratio between the minimal and maximal width is less than a parameter β, we just divide it by the usual division rule in order to avoid needle shapes (see Algorithm 2.8, line 4). Finally, the usual division rule is applied as well if the width of any component of the box is less than ε in order to avoid the division of the dimensions in which the termination criterion is fulfilled. The second and third conditions are controlled in Algorithm 2.7 which returns -1 if either of them is satisfied (see Algorithm 2.8, line 4).

If the previous conditions are not satisfied, Algorithm 2.6 prunes the $\boldsymbol{pb}$ from the current box (and so divides it too) (See Algorithm 2.6, lines 13 to 22).

In steps 2-4 of Algorithm 2.6 we check whether the box is inside the pruneable region in order to discard the whole box if possible. From the other extreme, if PR is inside the box, only $\boldsymbol{cpb}$ can be pruned, and so PI (Pruning Index) is computed by the *relative volume/number of new boxes* (lines 5-6 of Algorithm 2.6).

If $\boldsymbol{cpb}$ can be shifted, we call the ChoosePB procedure (See Algorithm 2.7) which returns the number of shifted directions, the best $\boldsymbol{pb}$, and the value of PI.

Procedure ChoosePB decides which directions should be shifted in order to obtain the best $\boldsymbol{pb}$. It starts with a call to GetShiftFactorAndVol procedure (see Algorithm 2.8) where the relative volume vol_0 is calculated by shifting only in the directions i where $\boldsymbol{cpb}_i$ is outside $\boldsymbol{ob}_i$. PI of $\boldsymbol{spb}$ and the shifting factors are also calculated. The volume vol_0 is the maximal relative volume of any $\boldsymbol{pb}$ (see Corollary 2.24). Variable *out* returns the number of directions where $\boldsymbol{cpb}$ was outside the current box and therefore shifted.

Algorithm 2.6 The Baumann Tent Pruning Division method (BTPD)

Funct BTPD($\boldsymbol{x}, \boldsymbol{g}(\boldsymbol{x}), b, \boldsymbol{f}(b), \tilde{f}, \alpha, \beta$)

1: $\boldsymbol{ob} = \boldsymbol{x} - b$; $\underline{pr} = \frac{\tilde{f} - \underline{f}(b)}{\overline{g}(\boldsymbol{x})}$; $\overline{pr} = \frac{\tilde{f} - \underline{f}(b)}{\underline{g}(\boldsymbol{x})}$
2: in = $1 - \sum_{i=1}^{n} \frac{\underline{ob_i}}{\underline{pr_i}}$
3: **if** (in > 0) *The box is inside the pruneable region,*
4: **return** *so we can discard the whole box*
5: **else if** ($\boldsymbol{bpr} \subseteq \boldsymbol{ob}$)
6: $\boldsymbol{pb} = \boldsymbol{bpr}/n$; $PI = Vol(\boldsymbol{pb})/(2n \cdot Vol(\boldsymbol{ob}))$; $nshift = 0$;
7: **else**
8: *nshift* = ChoosePB($\boldsymbol{ob}, \boldsymbol{bpr}, \boldsymbol{pb}, PI, \beta$)
9: **if** (*nshift* == -1) *The box is badly shaped and the new*
10: DivideBox($\boldsymbol{x}$); *subboxes would inherit it*
11: **return**
12: **if** ($PI \cdot n^n > \alpha$) *It is worth pruning*
13: **for** ($i = 1; i \leq n; i++$) **do**
14: $d_i = \min\{\underline{ob_i} - \underline{pb_i}, \overline{pb_i} - \overline{ob_i}\}$ *In order to obtain more cube-like boxes*
15: $sort = Sort(d)$; $\boldsymbol{y} = \boldsymbol{x}$; j=1; *the widths of the fixed new sides are ordered,*
16: **for** ($k = 1; k \leq n; k++$) **do** *to always divide the longest available width.*
17: $i = sort_k$;
18: **if** ($c_i + \underline{pb_i} > \underline{x_i}$)
19: $\boldsymbol{x}^j = \boldsymbol{y}; \boldsymbol{x}_i^j = [\underline{x_i}, c_i + \underline{pb_i}]$; j++ *Generating*
20: **if** ($c_i + \overline{pb_i} < \overline{x}_i$) *new boxes*
21: $\boldsymbol{x}^j = \boldsymbol{y}; \boldsymbol{x}_i^j = [c_i + \overline{pb_i}, \overline{x}_i]$; j++
22: $\boldsymbol{y}_i = c_i + \boldsymbol{pb}_i$;
23: **else**
24: DivideBox($\boldsymbol{x}$);
25: **return**

Algorithm 2.7 checks if more shifting is possible. For this, the *sf* vector is sorted in increasing order. Since the first *out* directions are already shifted, only the remaining ones are checked (see lines 7-14). Step 10 computes a new relative volume from the previous one (see Remark 2.22). If the *relative volume/number of new boxes* is better than the previous PI, we update it. Finally, Steps 17-21 compute the best $\boldsymbol{pb}$, and return it. If the best $\boldsymbol{pb}$ has a high enough PI, Algorithm 2.6 computes the new boxes in a way that more cube-like boxes are generated (see Steps 13-22).

2.3.4 Numerical results

The experiments have been carried out with a dual processor Intel Xeon 2.6GHz with 1GB memory. The programs were compiled with g++ under Linux using automatic differentiation of the C-XSC library [82] and the interval arithmetic of the Profil/Bias libraries [91]. The results were obtained by running the traditional method (Algorithm 1.1) with and without the BTPD method. In order to check the level of improvement of the computational cost, our experiments have been carried out on a wide set of well known test functions in Global Optimization literature.

We present two setups for the BTPD method, which correspond to two kinds of optimal settings of the parameters α and β. These settings were optimized taking into account the computational effort used by the algorithms. By computational effort we mean $\#F + n\#G$, where $\#F$ and $\#G$ are the number of inclusion function evaluations and inclusion gradient evaluations, respectively. Denoting by E_0 the effort of Algorithm 1.1, and by E the effort of Algorithm 1.1 with BTPD, the optimization of the parameters was carried out with respect to the *total effort* (summing up the effort for all the problems) and with respect to the *average efficiency rate* (the average of E_0/E for all the problems).

Algorithm 2.7 : Choose the best $\boldsymbol{pb}$.

Funct ChoosePB($\boldsymbol{ob}, \boldsymbol{bpr}, \boldsymbol{pb}, \beta$)

$n = Dimension(\boldsymbol{ob})$;
if (!GetShiftFactorAndVol($\boldsymbol{ob}, \boldsymbol{bpr}, sf, vol_0, out, \beta$)) *The box is badly shaped*
  return -1; *and the new boxes would inherit it*
$sort = Sort(-sf)$;
$nshift = out$; *nshift will be the number of directions to shift*
$PI = vol_0/(2n - 2out)$; *PI is the pruning index*
for $(k = out + 1; k < n; k++)$ **do**
  if $(osf - sf(sort_k) < 0)$ *We cannot shift more*
    break
  $vol_k = vol_{k-1}\frac{sf(sort_k)}{osf}\left(\frac{osf - sf(sort_k)}{osf}\right)^{n-k}\frac{(n-k+1)^{n-k+1}}{(n-k)^{n-k}}$;
  if $(vol_k/(2n - 2k) \geq PI)$
    $osf- = sf(sort_k)$;
    $PI = vol_k/(2n - 2k)$;
    $nshift = k$;
  else
    break ;
for $(i = 1; i <= nshift; i++)$ **do** *In the shifted dimensions*
  $\boldsymbol{pb}(sort_i) = \boldsymbol{ob}(sort_i)$; $\boldsymbol{pb}_i = \boldsymbol{ob}_i$,
for $(i = nshift + 1; i \leq n; i++)$ **do** *otherwise*
  $\boldsymbol{pb}(sort_i) = osf * \boldsymbol{bpr}(sort_i)/(n - nshift)$; *use* (2.29).
return *nshift*;

Algorithm 2.8 : Compute the vector sf and the maximal volume of $\boldsymbol{spb}$.

Funct GetShiftFactorAndVol($\boldsymbol{ob}, \boldsymbol{bpr}, sf, vol_0, out, \beta$)

$w_{min} = min\{w(\boldsymbol{ob}_1), \ldots, w(\boldsymbol{ob}_n)\}$; *The minimal width*
$osf = 1.0; vol_0 = 1.0$; vol_0 *is the maximal relative volume of any* $\boldsymbol{pb}$
for $(i = 1; i \leq n; i++)$ **do**
  if $(w_{min}/w(\boldsymbol{ob}_i) > \beta$ && $w(\boldsymbol{ob}_i) > \varepsilon)$ *It is not badly shaped*
    $sf_i = \underline{ob_i}/\underline{pr_i}$;
    if $(sf_i > 1/n)$
      $vol_0* = w(\boldsymbol{bpr}_i/n)/w(\boldsymbol{ob}_i)$;
    else *We shift the dimensions*
      $out++$; *where* $\boldsymbol{cpb}$ *is outside by default.*
      if $(\overline{pr}_i > -\underline{pr}_i)$ *To avoid division by tiny number*
        $vol_0* = \overline{ob}_i(1 - \underline{pr}_i/\overline{pr}_i)$;
      else
        $vol_0* = \underline{ob_i}(\overline{pr}_i/\underline{pr}_i - 1)$;
      $osf- = sf(i)$;
  else *It is badly shaped*
    if $(w(\boldsymbol{bpr}_i) == \infty)$ *we cannot divide in this direction using any* $\boldsymbol{pb}$
      return 0;
    else
      $sf_i = 100$; *To avoid shifting in this direction*
if $(out > 0)$ *If there are already shifted dimensions*
  $vol_0* = (n/(n - out))^{n-out}$;
  return 1;

For the optimization of the parameters we used a naive grid search algorithm. In fact, the search was performed in only one parameter at a time, holding the other fixed, and we used a loop to optimize both parameters in turns until they converge to a local optimum. Optimizing for the total effort the best obtained results are $\alpha = 1.45$ and $\beta = 0.028$ while for the average efficiency rate $\alpha = 0.38$ and $\beta = 0.027$. They will be called setting 1 and 2, respectively. The results for the *badly shaped* parameter β are very similar in both cases. This happens because this parameter has an effect only on a few problems. However, for the parameter α the results are quite different. This is because most of the test problems can be solved easily, while only a few of them take more time and computational effort. Thus, when minimizing the total effort those easy problems do not have much influence on the result. Unfortunately this means that one cannot decide the best values for α and β for a given problem beforehand. On the other hand, the results are not so different for both settings, as one can see it in Tables 2.7 and 2.8. In those tables for column headings the following notation has been used:

Problem:	The name of the test function.
Ref.:	Reference where the problem is described.
n:	Dimension of the problem.
ε:	Termination criterion ($\mathrm{w}(X) \leq \varepsilon$).
E_0:	Computational effort of Algorithm 1.1.
E_1:	Effort of Algorithm 1.1 with BTPD with $\alpha = 1.45$ and $\beta = 0.028$.
E_2:	Effort of Algorithm 1.1 with BTPD with $\alpha = 0.38$ and $\beta = 0.027$.
T_0:	Computational time in seconds of Algorithm 1.1.
T_1:	Time of Algorithm 1.1 with BTPD with $\alpha = 1.45$ and $\beta = 0.028$.
T_2:	Time of Algorithm 1.1 with BTPD with $\alpha = 0.38$ and $\beta = 0.027$.

For every problem we have chosen a termination criterion that ensures successful termination within one hour. Table 2.7 shows that the new BTPD method reduced the computational effort in 25 and 26 cases out of the 42 test problems for setting 1 and 2, respectively. The computational cost is the same (with and without the BTPD method) in 12 and 11 problems for setting 1, 2 respectively, and for 5 problems the effort is greater when the new pruning technique is applied (for both settings). The worst results were obtained for the Rosenbrock-10 function (see Table 2.8). The reasons can be that the use of the BTPD method changes the set of evaluated points that improve the $\tilde{f}$ value, and that the pruning method produces the generation of too many subproblems. For the original Griewank-10 problem we have obtained an extreme speed-up (64.05) when the parameter α was very small, but it was a bit slower when the parameter was higher. The reason is that the minimizer point is in the center of the search region, therefore, when only traditional division is used, the minimizer point is shared by several boxes that cannot be rejected. The asymmetric division generated by the BTPD method avoids this problem, and smaller α provide a greater possibility to do it sooner. This problem was overwhelming the results of the parameter optimization as well, therefore we slightly modified the problem by changing the search region from $[-600, 600]^{10}$ to $[-599, 601]^{10}$.

The average efficiency rate is 1.13 and 1.14, while the efficiency rate on the total effort is 1.83 and 1.82 for settings 1 and 2, respectively. This shows that on harder problems we can achieve better results. We can also see that the average results obtained by the two settings are almost equal, and the individual results do not differ too much for the two settings either. Therefore, in general, any value in the same order of magnitude may be useful, even if the optimal setting for the given problem is not the one used.

Examining the results of the computational time we can see that the time ratio is always smaller than the corresponding efficiency rate, due to the time spent on the calculation of the best pruneable box. For instance, the Ratz-5 and Ratz-6 problems need more time with BTPD method (relatively small difference compared to the others) even though the effort is less. Therefore, we can conclude that the application of the new BTPD method needs a non-negligible computational effort but in average, its application improves the performance of the interval B&B method, especially for hard to solve problems.

Future work has to be carried out to tune the variable α accordingly to the problem at hand, or even to the box being examined depending on how promising the actual box is to contain the global minimum. As it was shown in [11], every subproblem needs a different level of division that has to be taken into account to save computations.

Table 2.7: Computational efficiency comparison

Problem	n	ε	E_0	E_0/E_1	E_0/E_2
Goldstein-Price	2	10^{-10}	33694	1.49	1.54
Six-Hump-Camel-Back	2	10^{-10}	3592	1.26	1.31
Three-Hump-Camel-Back	2	10^{-10}	2328	1.15	1.22
Griewank-2	2	10^{-10}	1238	1.00	1.00
Branin-2	2	10^{-10}	2718	1.01	1.01
Rosenbrock	2	10^{-10}	1319	1.04	1.05
Simplified-Rosenbrock	2	10^{-10}	1319	1.04	1.05
Price	2	10^{-10}	3200	1.00	1.00
Treccani	2	10^{-10}	1506	1.00	1.00
Matyas	2	10^{-10}	2070	1.16	1.16
EX1	2	10^{-10}	468	1.01	0.97
Branin	2	10^{-10}	5527	1.02	1.03
Ratz-4	2	10^{-10}	6210	1.08	1.12
Henriksen-Madsen-3	2	10^{-10}	8631	1.01	1.02
Beale	2	10^{-10}	2755	1.02	1.03
Chichinadze	2	10^{-10}	597	1.00	1.03
Levy-13	2	10^{-10}	281	1.04	1.04
Neumaier3-2	2	10^{-10}	2718	1.29	1.29
Schwefel-2.1	2	10^{-10}	3121	0.97	0.97
Levy-3	2	10^{-10}	5492	1.00	1.00
Levy-5	2	10^{-10}	974	1.00	1.00
Henriksen-Madsen-4	3	10^{-10}	26725	1.02	1.00
Schwefel-3.1	3	10^{-10}	527	1.08	1.08
Hartman-3	3	10^{-10}	4715	1.13	1.18
Levy-8	3	10^{-10}	475	1.03	1.03
Shekel-5	4	10^{-10}	1311	1.00	1.00
Shekel-7	4	10^{-10}	1376	1.00	1.00
Shekel-10	4	10^{-10}	1431	1.00	1.00
Rosenbrock-5	5	10^{-10}	17921	0.88	0.87
Hard-Problem	6	10^{-10}	25	1.00	1.00
Hartman-6	6	10^{-10}	17796	0.97	0.94
Levy-18	7	10^{-10}	1800	1.12	1.12
Levy-12	10	10^{-10}	3111	1.08	1.08
Rosenbrock-10	10	10^{-10}	923896	0.57	0.54
Griewank-10	10	10^{-10}	5361	1.00	1.00
Ratz-5	3	10^{-3}	633508	1.18	1.18
Ratz-6	5	10^{-3}	1201476	1.16	1.16
Ratz-7	7	10^{-3}	1900512	1.14	1.14
Ratz-8	9	10^{-3}	3183190	1.31	1.31
Kowalik	4	10^{-4}	10086339	2.75	2.59
EX2	5	10^{-6}	13851668	2.72	2.82
Rastrigin-10	10	10^{-6}	2319366	0.94	0.94
Average			816007	1.83	1.82
Average of ratios				1.13	1.14

Table 2.8: Computational time comparison

Problem	n	Ref.	ε	T_0	T_0/T_1	T_0/T_2
Goldstein-Price	2	[147]	10^{-10}	0.31	1.15	1.24
Six-Hump-Camel-Back	2	[147]	10^{-10}	0.03	1.50	1.50
Three-Hump-Camel-Back	2	[147]	10^{-10}	0.01	1.00	1.00
Griewank-2	2	[147]	10^{-10}	0.02	1.00	1.00
Branin-2	2	[147]	10^{-10}	0.05	1.25	1.00
Rosenbrock	2	[24]	10^{-10}	0.01	1.00	1.00
Simplified-Rosenbrock	2	[24]	10^{-10}	0.00	1.00	1.00
Price	2	[25]	10^{-10}	0.02	1.00	1.00
Treccani	2	[24]	10^{-10}	0.00	1.00	1.00
Matyas	2	[150]	10^{-10}	0.00	1.00	1.00
EX1	2	[19]	10^{-10}	0.00	1.00	1.00
Branin	2	[147]	10^{-10}	0.06	1.00	1.00
Ratz-4	2	[128]	10^{-10}	0.10	1.11	1.25
Henriksen-Madsen-3	2	[80]	10^{-10}	0.43	1.00	0.98
Beale	2	[150]	10^{-10}	0.02	1.00	1.00
Chichinadze	2	[25]	10^{-10}	0.01	1.00	1.00
Levy-13	2	[150]	10^{-10}	0.00	1.00	1.00
Neumaier3-2	2	[111]	10^{-10}	0.01	1.00	1.00
Schwefel-2.1	2	[150]	10^{-10}	0.02	0.67	0.67
Levy-3	2	[150]	10^{-10}	0.29	1.00	0.97
Levy-5	2	[150]	10^{-10}	0.05	0.83	1.00
Henriksen-Madsen-4	3	[80]	10^{-10}	1.87	0.97	0.95
Schwefel-3.1	3	[150]	10^{-10}	0.01	1.00	1.00
Hartman-3	3	[147]	10^{-10}	0.08	1.14	1.00
Levy-8	3	[150]	10^{-10}	0.03	1.50	3.00
Shekel-5	4	[147]	10^{-10}	0.03	1.50	1.50
Shekel-7	4	[147]	10^{-10}	0.03	1.00	1.00
Shekel-10	4	[147]	10^{-10}	0.04	1.00	1.00
Rosenbrock-5	5	[111]	10^{-10}	0.16	0.70	0.70
Hard-Problem	6	[128]	10^{-10}	0.00	1.00	1.00
Hartman-6	6	[147]	10^{-10}	0.38	0.97	0.93
Levy-18	7	[150]	10^{-10}	0.08	1.60	1.60
Levy-12	10	[150]	10^{-10}	0.23	1.35	1.21
Rosenbrock-10	10	[111]	10^{-10}	11.94	0.47	0.45
Griewank-10*	10	[147]	10^{-10}	0.20	0.91	0.91
Ratz-5	3	[128]	10^{-3}	125.24	0.94	0.95
Ratz-6	5	[128]	10^{-3}	166.79	0.95	0.95
Ratz-7	7	[128]	10^{-3}	226.56	1.01	0.97
Ratz-8	9	[128]	10^{-3}	582.01	2.18	2.18
Kowalik	4	[150]	10^{-4}	251.29	2.51	2.34
EX2	5	[19]	10^{-6}	279.71	2.18	2.31
Rastrigin-10	10	[111]	10^{-6}	87.74	0.82	0.81
Average				41.33	1.49	1.48
Average of ratios					1.12	1.15

* The search region of this problem is modified to $[-599, 601]^{10}$.

2.4 More Discarding Tests

2.4.1 Monotonicity test for feasible and undetermined boxes

The monotonicity test commonly applied in the interval B&B methods (see Section 1.4) is used to decide whether the objective function f is monotonous in a feasible box $\boldsymbol{x}$. If $\boldsymbol{g}(\boldsymbol{x}) = (\boldsymbol{g}_1(\boldsymbol{x}), \ldots, \boldsymbol{g}_n(\boldsymbol{x}))$ is an inclusion function for the gradient of the objective function, and for a feasible box $\boldsymbol{x}$ we have that $0 \notin \boldsymbol{g}(\boldsymbol{x})$, then $\boldsymbol{x}$ cannot contain a global minimizer in its interior. If $\boldsymbol{x}$ is not on the boundary of the search region, it can be rejected, otherwise it can be narrowed to the intersection of $\boldsymbol{x}$ and the boundary of the search region. However, we can only apply this test to boxes which are guaranteed to be feasible. We propose the following extension. If $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ is an undetermined box for which $\boldsymbol{g}(\boldsymbol{x}) \geq 0$ for some variable x_i, and the upper-facet of box $\boldsymbol{x}$ associated to x_i, $\boldsymbol{x}'$ satisfies the constraints in which variable x_i appears, then box $\boldsymbol{x}$ can either be discarded or reduced to $\boldsymbol{x}'$ (if $\boldsymbol{x}'$ is on the boundary of the search region). To see this, notice that the feasible point(s) in $\boldsymbol{x}$ which has the best objective value lies in the facet $\boldsymbol{x}'$, and therefore the box $\boldsymbol{x}$ can be reduced to it. Furthermore, if $\boldsymbol{g}_i(\boldsymbol{x}) > 0$, and $h_j(\boldsymbol{x}') < 0$ for all the constraints $h_j(x) \leq 0$ in which variable x_i appears, and also $\boldsymbol{x}'$ is not on the border of the search region then the whole box can be removed. A similar result can be obtained if $\boldsymbol{g}_i(\boldsymbol{x}) \leq 0$, considering the corresponding lower-facet. We will refer to this test as the monotonicity test for undetermined boxes.

2.4.2 Projected one-dimensional interval Newton method

The multi-dimensional interval Newton step (see Section 1.4) is not always useful (see [46]). However, in those cases the following 'projected one-dimensional interval Newton method' can be used to discard or reduce boxes. Let $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$ be an undetermined (or feasible) box, and suppose that $\boldsymbol{x}$ certainly satisfies the constraints in which the variable x_i appears (if any). We propose to apply the one-dimensional interval Newton method (see [71]) to the one-dimensional equation $\frac{\partial f}{\partial x_i} = 0$, considering $\boldsymbol{x}_i$ as the input interval, i.e., we consider x_j, $j = 1, \ldots, n$, $j \neq i$ as fixed variables, with values equal to $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$, respectively. In this way only one element of the Hessian matrix has to be computed, and the computations become much easier. Notice that our proposal is to perform not only one step (or iteration) of the interval Newton method, as it is usually done in interval global optimization algorithms, but let the method run longer (that is, till the box is discarded or the last reduction is less then a tiny number (we used the parameter of the stopping criteria)). This method is specially suitable when the objective function is convex along variable x_i. Notice that to apply this method f must be C^2.

2.4.3 Projected one-dimensional non-convexity test

The non-convexity of a function is guaranteed if the Hessian matrix is not positive definite. Similar to the previous test, we want to avoid the computation of the whole Hessian matrix. The test works as follows: Let $\boldsymbol{x}$ be an undetermined (or feasible) box, and suppose that an inclusion of the diagonal of the Hessian matrix H of the objective function at $\boldsymbol{x}$ can be computed. If $\overline{H_{ii}} < 0$ for some i and both the upper and lower-facets of $\boldsymbol{x}$ associated to the coordinate i are feasible, then f cannot be convex at $\boldsymbol{x}$ and there cannot be any global minimum in the interior of $\boldsymbol{x}$. If neither the lower nor the upper-facet is on the boundary of the search region, box $\boldsymbol{x}$ can be discarded, otherwise it can be diminished to the intersection of the boundary and the lower-, upper-facets associated to the coordinate i.

The computation of the diagonal of the Hessian matrix is done element by element. More precisely, for the computation of $\boldsymbol{H}_{ii}$ the only variable considered is x_i, and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_n$ are taken as input intervals. Then, an inclusion for the second derivative is computed, that is, the element $\boldsymbol{H}_{ii}$ of the Hessian. Therefore, the elements of the diagonal of the Hessian matrix are computed one by one until for one of them the above condition holds, or there are no more variables to check.

2.5 Conclusions

In this chapter we have introduced new methods which try to accelerate the standard interval B&B method. It was shown that the best inclusion function can vary depending on the optimization problem considered, but as a rule, for a general problem, the natural interval extension should be used for intervals with a width larger than 0.05, and the forward slope arithmetic, otherwise.

Different pruning techniques were also studied, which can significantly speed up the traditional method. Two different generalizations of a one-dimensional pruning test were discussed. First, a projected pruning technique, which was used with support functions in order to achieve even better results was examined. In the second generalization, the aim was to use a multi-dimensional linear lower bounding function, despite the fact that it generates a rejectable region with a difficult shape to handle. It was shown that using the Baumann point the best pruneable box is easy to compute, and thus an efficient technique can be derived.

In the last section some modifications of existing discarding tests are discussed. All of them are extensions of well-known techniques for handling constraint optimization problems in the sense that they can be used not only for feasible boxes, but for undetermined boxes as well. Two of them, the Newton method and the Non-convexity test use the Hessian matrix, which is expensive to compute for high-dimensional problems. With our modifications one can avoid the evaluation of the whole Hessian matrix while still being able to use the strength of the original methods.

Chapter 3

Biobjective methods

Multiobjective optimization problems are ubiquitous. Many real-life problems require taking into account several conflicting points of view. In fact, although the origins of multiobjective optimization literature are linked to Utility Theory, Game Theory, Linear Production Theory and Economics (see [63]), we can now find applications in many and diverse fields, such as portfolio optimization [36], jury selection [133], airline operations [37], radiation therapy [94], manpower planning [135] or reservoir management [2], among others. In [153], White mentions more that 500 applications between 1955 and 1986. Classical references on multiobjective optimization are the books [14, 15, 137, 156, 157]. Other more recent books are [35, 103].

Optimization of continuous nonlinear, nonconvex function is an NP-hard problem. Therefore, obtaining a single efficient point of a problem, where more than one such objective is present is NP-hard as well. Hence finding all the efficient points of such a problem seems to be unreachable. This is the reason why methods finding the whole efficient set are lacking in literature.

The chapter is organized as follows. Section 3.2 describes a Lexicographical-Like method [50] that is able to find any horizontally (or vertically) efficient point. In Section 3.3 a Constraint-Like method [54] is presented, which obtains an outer approximation of the whole efficient set. A biobjective interval B&B method [53, 56] is developed with the same aim in Section 3.4. Convergence properties are also showed for the biobjective interval B&B method. Finally, in Section 3.5, some conclusions of the chapter are derived.

3.1 Basic notions of biobjective optimization

We restrict ourselves to the biobjective case in this chapter, that is, to the problem

$$\begin{array}{ll} \min & \{f_1(x), f_2(x)\} \\ & x \in S \subset \mathbb{R}^n \end{array} \tag{3.1}$$

where S denotes the feasible set of the problem, which can be written through some analytical constraints. We will further denote the vector of objective functions by $f(x) = (f_1(x), f_2(x))$, and the image of the feasible region by $Z = f(S) \subset \mathbb{R}^2$.

When dealing with multiobjective problems we need to clarify what *solving* a problem means. Some widely known definitions to explain the concept of solution of (3.1) follow. The most popular one for such a problem is efficiency (or Pareto optimality).

Definition 3.1. *A feasible vector $x \in S$ is said to be* efficient *iff there does not exist another feasible vector $y \in S$ such that $f_j(y) < f_j(x)$ for at least one objective $j \in \{1, 2\}$ and $f_i(y) \leq f_i(x)$ for the other objective $i \in \{1, 2\}, i \neq j$. The set S_E of all the efficient points is called the* efficient set.

Efficiency is defined in the decision space. The corresponding definition in the criterion space is as follows.

Definition 3.2. *An objective vector $z = f(x) \in Z$ is said to be* nondominated *(or also* efficient*) iff x is efficient. The set of all nondominated vectors will be denoted by $Z_N (= f(S_E))$.*

Another related concept widely used is weak efficiency.

Definition 3.3. *A feasible vector $x \in S$ is said to be* weakly efficient *iff there does not exist another feasible vector $y \in S$ such that $f_i(y) < f_i(x)$ for $i = 1, 2$. The set S_{WE} of all the weakly efficient points is called the* weakly efficient set.

Definition 3.4. *An objective vector $z = f(x) \in Z$ is said to be* weakly nondominated *(or also* weakly efficient*) iff x is weakly efficient. The set of all weakly nondominated vectors will be denoted by $Z_{WN} (= f(S_{WE}))$.*

Note that any efficient point is weakly efficient, but the converse does not hold in general. At this point, we want to introduce a new definition, in which we consider two (overlapping) groups of weakly efficient points so that the intersection of these two groups is the set of efficient points (see Figure 3.1).

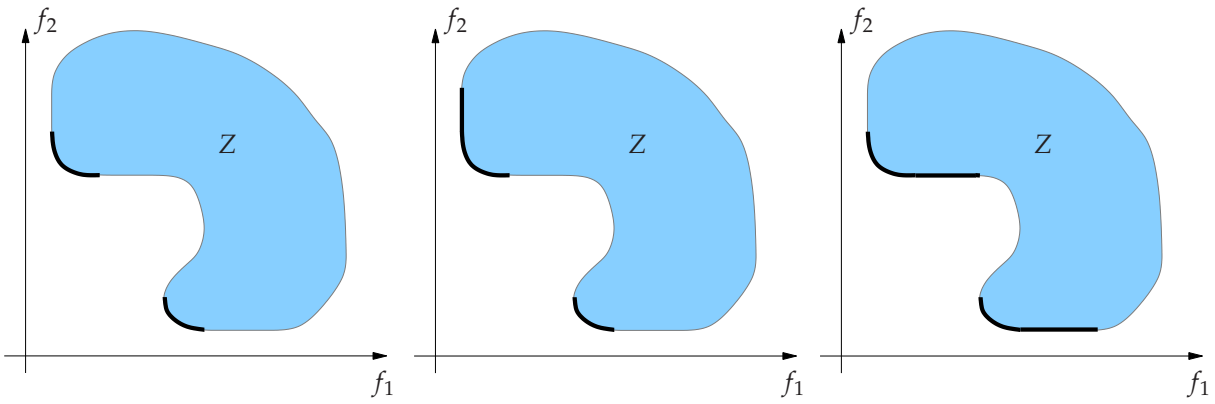


Figure 3.1: In thick lines we have: a) Efficient points, b) vertically weakly efficient points, c) horizontally weakly efficient points.

Definition 3.5. *A feasible vector $x \in S$ is said to be* vertically weakly efficient *iff there does not exist another feasible vector $y \in S$ such that $f_1(y) < f_1(x)$ and $f_2(y) \leq f_2(x)$. A feasible vector $x \in S$ is said to be* horizontally weakly efficient *iff there does not exist another feasible vector $y \in S$ such that $f_2(y) < f_2(x)$ and $f_1(y) \leq f_1(x)$.*

Thus, a point is efficient if and only if it is both vertically and horizontally weakly efficient, and weakly efficient points are not efficient as soon as their image lies vertically above another one (vertically weakly efficient points), or horizontally at the left of another one (horizontally weakly efficient points).

Other special vectors that will be used later on are the ideal and the nadir objective vectors, which are defined as follows.

Definition 3.6. *The* ideal objective vector *of problem (3.1) is a vector $z^* \in \mathbb{R}^2$ whose components z_i^* are obtained by minimizing each of the objective functions individually subject to the original constraints of problem (3.1).*

Definition 3.7. *The* nadir objective vector *of problem (3.1) is a vector $z^{nad} \in \mathbb{R}^2$ giving the upper bounds of Z_N, i.e. $z_i^{nad} = \max\{f_i(x) \mid x \in S_E\}$.*

Ideally, solving (3.1) means obtaining the whole efficient set, although in practice we will be satisfied if we obtain some representative (in a sense) efficient points. There is a great and rich variety of methods

with that aim, as can be seen in the references mentioned above (weighting method, constraint method, lexicographic method,...). However, most of the literature on multiobjective optimization deals with either *discrete* problems or with continuous multiobjective *linear* problems, whereas the interest in this chapter is in *nonlinear* multiobjective optimization. Although less studied, we can also find many references dealing with this last topic in literature (see [103] and the references therein).

3.2 Lexicographical-like method

In this section we present a modification of the classical lexicographic method [157]. As will be shown, our method solves some of the drawbacks of the classical lexicographic method, and has very appealing theoretical properties. It may also be seen as a special type of constraint method.

Sometimes one of the objectives of the problem is much more important for the decision maker than the other one. Thus, when obtaining (weakly) efficient solutions for problem (3.1) we should use a method which takes that fact into account: we do not want to present the decision maker the whole efficient set, but just a small part of it which reflects his/her preferences. This suggests that we should use the classical lexicographic method (see [58]). Applied to the biobjective problem (3.1), the method is as follows.

Classical lexicographic method

1. The decision maker arranges the objective functions according to their absolute importance f_1, f_2.
2. The most important objective function is optimized subject to the original constraints, that is, we solve the problem

$$(P_1) \qquad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & x \in S \end{array}$$

3. If (P_1) has a unique solution $x^*_{(1)}$, then STOP: $x^*_{(1)}$ is the *lexicographic solution*, to be presented to the decision maker as the solution for the whole biobjective optimization problem (3.1).
4. Otherwise, the second objective function is optimized subject to the original constraints and to an additional one to guarantee that f_1 preserves its optimal value, $f_1^* = f_1(x^*_{(1)})$, that is, we solve the problem

$$\begin{array}{ll} \min & f_2(x) \\ \text{s.t.} & x \in S \\ & f_1(x) = f_1^* \end{array}$$

   Any solution to this second problem is a lexicographic solution, to be presented to the decision maker as a solution for (3.1).

Although the method has good properties (any lexicographic solution is efficient), it presents an important drawback. In particular, it is very likely that the less important objective f_2 is not taken into consideration at all: if problem (P_1) has a unique solution, the second objective does not have any influence on the solution. Therefore, the method does not allow a small increment of the first objective to be traded off with a possibly important decrement of the second objective.

We propose below a new method, which we have named δ-lexicographic method, that circumvents the disadvantage of the classical method. It needs a procedure to obtain some regions of δ-optimality, for which we used a modification of the method described in Section 1.5.

δ-lexicographic method

1. The decision maker arranges the objective functions according to their absolute importance, f_1, f_2, and indicates a fraction $\delta \geq 0$ by which he/she allows the optimal value of the primary objective f_1 to deteriorate in order to improve upon the second objective f_2.
2. The region R_δ^1 of δ-optimality of problem

$$(P_1) \quad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & x \in S \end{array}$$

   is calculated, where $R_\delta^1 = \{x \in S \mid f_1(x) - f_1^* \leq \delta f_1^*\}$.
3. The second objective function is optimized subject to the original constraints and to an additional one guaranteeing δ-optimality for f_1, that is, we solve the problem

$$(P_2^\delta) \quad \begin{array}{ll} \min & f_2(x) \\ \text{s.t.} & x \in R_\delta^1 (\subseteq S) \end{array}$$

   Any solution to this second problem is a *δ-lexicographic solution*, to be presented to the decision maker as a solution for (3.1).

As we can see, in addition to solving problem (P_1) to optimality, we now obtain its region of δ-optimality as well. In this way, we allow a relative increment of δ% in the first objective, to be traded off with a decrement in the second objective (notice that we use relative precision when obtaining the region of δ-optimality; in particular, this means that we are assuming that f_1 and f_2 are strictly positive on S; however, a similar construction can be developed using absolute errors, and in this case, the sign of the objective function is arbitrary). Also, as soon as $\delta > 0$ we take both objectives into consideration. In our opinion, this kind of trading-off is very appealing to the decision maker. In this sense, the δ-lexicographic solutions obtained with the δ-lexicographic method are good alternative solutions for the decision maker, since they can quite accurately represent his/her preferences, and might even be used in an interactive way, in case δ cannot be fixed in advance.

It is important to highlight here that, regardless the use of relative or absolute measures, the choice of δ has to be made very carefully. For instance, when using relative measures, if for a given problem the exact range of f_1 on the feasible set S is $f_1(S) = [100, 1000]$, then setting $\delta = 0.1$ implies that in the region of δ-optimality the range of f_1 will be the interval $f_1(R_\delta^1) = [100, 110]$, which can be a good range compared to $f_1(S)$. However, if $f_1(S) = [100, 105]$ and we again set $\delta = 0.1$, then the range of f_1 on R_δ^1 will again be $f_1(R_\delta^1) = [100, 110]$, which is obviously not acceptable at all. This means that in order to choose an adequate δ, the range $f_1(S)$ has to be taken into account. In the same way, the global shape of f_1 is also important, and in particular its flatness around its optimal solutions: if f_1 is flat around an optimal solution, then R_δ^1 can become much larger than expected.

The δ-lexicographic solutions also have very good properties, as can be seen from the following results.

Theorem 3.8. *A feasible vector $\check{x} \in S$ is horizontally weakly efficient for problem (3.1) if and only if there exists a $\delta \geq 0$ such that $\check{x}$ is a δ-lexicographic solution of (3.1).*

Proof. Firstly, let us suppose that $\check{x}$ is horizontally weakly efficient for problem (3.1). We have to prove that there exists a $\delta \geq 0$ such that $\check{x}$ is an optimal solution of (P_2^δ). If we set

$$\delta = \frac{f_1(\check{x}) - f_1^*}{f_1^*}$$

then $\check{x}$ is a feasible vector for problem (P_2^δ). To prove that $\check{x}$ is an optimal solution for (P_2^δ), let us suppose that there exists a feasible vector x for problem (P_2^δ) with a strictly better objective value, $f_2(x) < f_2(\check{x})$.

Since $x \in R^1_\delta$, then $f_1(x) - f_1^* \leq \delta f_1^*$ and substituting δ by its value, we obtain that $f_1(x) \leq f_1(\check{x})$. But this means that $\check{x}$ is not horizontally weakly efficient (it is dominated by x), which is a contradiction.

Secondly, let us consider now that $\check{x}$ is an optimal solution to (P_2^δ) for some $\delta \geq 0$. We have to prove that $\check{x}$ is horizontally weakly efficient for problem (3.1). Suppose that it is not. Then there must exist a feasible vector $x \in S$ such that $f_1(x) \leq f_1(\check{x})$ and $f_2(x) < f_2(\check{x})$. However, $\check{x} \in R^1_\delta$ and $f_1(x) \leq f_1(\check{x})$. Thus, x belongs to R^1_δ too. But then, x is a feasible vector of problem (P_2^δ) whose objective value is strictly less than the one at $\check{x}$. But this is a contradiction, since $\check{x}$ is an optimal solution to (P_2^δ). □

A direct consequence is that any horizontally weakly efficient solution can be obtained by the δ-lexicographic method by an adequate choice of δ (notice that this is not true for the classical lexicographic method). Since the efficient set is included in the set of horizontally weakly efficient solutions, the δ-lexicographic method can obtain any efficient solution.

In a similar way, if we take $f_2(x)$ as the main objective, we obtain that a feasible vector $\check{x} \in S$ is vertically weakly efficient if and only if there exists a $\delta \geq 0$ such that $\check{x}$ is an optimal solution of a problem in the form:

$$(P_1^\delta) \quad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & x \in R^2_\delta \end{array},$$

where R^2_δ is the region of δ-optimality of the problem

$$(P_2) \quad \begin{array}{ll} \min & f_2(x) \\ \text{s.t.} & x \in S \end{array}.$$

Therefore, the following corollary is obtained.

Corollary 3.9. *A feasible vector $x \in S$ is efficient if and only if there exist $\delta_1, \delta_2 \geq 0$ such that x is an optimal solution for both $(P_1^{\delta_1})$ and $(P_2^{\delta_2})$.*

To sum up, we can see that the δ-lexicographic method is rather appealing, since, in principle it is able to obtain an enclosure of the whole efficient set. For our purposes it is even more important that it is able to find efficient points from any part of the efficient set. We will be mainly interested in the upper part with respect to the first objective.

It is worth highlighting here that our δ-lexicographic method is closely related to the well known α-constraint [35] (or ε-constraint [103]) method , with $\alpha = (1+\delta)f_1^*$, since (P_2^δ) is in fact the constrained problem

$$\begin{array}{ll} \min & f_2(x) \\ \text{s.t.} & f_1(x) \leq (1+\delta)f_1^* \\ & x \in S \end{array}.$$

The only, but important difference is that the bound α on f_1 is described in terms of the optimal value f_1^*, and not a constant given in advance: it is for this reason that two stages (P_1) and (P_2^δ) are needed in our algorithm. Also note the importance of being able to ensure that (P_1) delivers a correct global optimum value f_1^*, and not some value only known to be a local optimum.

On the other hand, in some applications for a decision maker, it is better to have a small region from which to choose the final solution (taking into account other aspects not included in the formulation of the problem) than just a single efficient point. In order to be able to offer the decision maker such a small region, we propose that in addition to solving (P_2^δ) to optimality, we obtain its region of θ-optimality, i.e.

$$R^2_{\theta(\delta)} = \{x \in R^1_\delta \mid f_2(x) - f_2^*(\delta) \leq \theta f_2^*(\delta)\},$$

where $f_2^*(\delta)$ denotes the optimal value of (P_2^δ) and $\theta > 0$ is a given value. Let us call this modified method the (δ, θ)-lexicographic method.

Definition 3.10. *A feasible vector $x \in S$ is said to be* (δ, θ)-efficient *iff there exists an efficient vector y such that $f_1(x) - f_1(y) \leq \delta f_1(y)$ and $f_2(x) - f_2(y) \leq \theta f_2(y)$.*

Notice that the points in $R^2_{\theta(\delta)}$ are not necessarily efficient, but they are (δ,θ)-efficient. For any fixed δ, as used in the δ-lexicographic method, the closeness of a (δ,θ)-efficient point to an efficient point is determined by the parameter θ alone. Thus, the smaller θ, the better.

The algorithmic description of the δ-lexicographic method, and its modification, the (δ,θ)-lexicographic method are given in the next sections.

3.2.1 Obtaining (a subset of) R^1_δ

The main procedure of the interval Branch and Bound algorithm that we have implemented to carry out the δ-lexicographic method is described in pseudo-code form in Algorithm 3.1. In that algorithm, $\boldsymbol{s}$ denotes a box containing the feasible set S, $\mathcal{L}_W^{min}$ is the working list during the first phase of the algorithm and $\mathcal{L}_S^{min}$ is the solution list for the first phase. $\mathcal{L}_W^{\delta}$ and $\mathcal{L}_S^{\delta}$ are the corresponding lists for the second phase. The algorithm is similar to the two-phase GBSSS method (see Section 1.5). The aim of the first phase is to obtain the optimal value of (P_1) within a tolerance ε (see Step 26 of Phase 1 in Algorithm 3.1), whereas the aim of the second is to obtain the region of δ-optimality R^1_δ within a tolerance η (see Steps 4,7,19 of Phase 2). However, Algorithm 3.1 differs from GBSSS in several points. Firstly, the bounds are computed here with the help of interval analysis. Secondly, in addition to the feasibility test (Feasible, in the code) and the δ-cut-off test (CutOffTest in the code, a modification of the classical cut-off test used in GBSSS (see Section 1.4), we also use a δ-monotonicity test for feasible and undetermined boxes (MonoTestFeas), and a δ-pruning test (Prune). Thirdly, Algorithm 3.1 does *not* compute the whole region R^1_δ, but just the part which is of interest for solving (P_2^δ) by way of a cut-off test due to the second objective function (RejectBySecObj).

A brief description of the new discarding tests used follows.

δ-cut-off Test: Every time a box $\boldsymbol{x}$ is chosen from the list $\mathcal{L}_W^{min}$, and provided that its midpoint c is feasible, we use $\overline{\boldsymbol{f}}_j(c)$ (previously computed to evaluate the centered form) to update (if possible, i.e. when $\overline{\boldsymbol{f}}_j(c) < \tilde{f}_j$) the best upper bound $\tilde{f}_j$ of the global minimum of (P_1). If updated, then the test sends $\mathcal{L}_W^{\delta}$ all the boxes $\boldsymbol{x}$ in $\mathcal{L}_W^{min}$ and $\mathcal{L}_S^{min}$ such that $\underline{\boldsymbol{f}}_j(\boldsymbol{x}) > \tilde{f}_j$ (since they cannot contain the optimal value of (P_1)), and then remove from $\mathcal{L}_W^{\delta}$ all the boxes $\boldsymbol{x}$ for which $\underline{\boldsymbol{f}}_j(\boldsymbol{x}) > \tilde{f}_j(1+\delta)$ (since they cannot be in R^1_δ).

δ-pruning Test: This test is a slight modification of the pruning method in Section 2.2.4: since we do not want to remove δ-optimal points, for the lower bound $\tilde{f}_j(1+\delta)$ is used instead of $\tilde{f}_j$.

δ-monotonicity test (for feasible and undetermined boxes): In Section 2.4.1 a monotonicity test (for feasible and undetermined boxes) is described to decide whether the objective function f_j is monotonous in a box, which allows either to discard the box or to reduce it to one of its facets. Since we are now computing R^1_δ and not solving (P_1) till optimality, we cannot discard monotonous boxes. Instead of discarding them, the δ-monotonicity test sends them to $\mathcal{L}_W^{\delta}$ (since they cannot contain the optimal value of (P_1)), provided that the box is not discarded by the next test. Note that this test is only useful in Phase 1.

Cut-off test due to the second objective function: Since our final purpose is to solve (P_2^δ), we can discard a box $\boldsymbol{x}$ if $\underline{\boldsymbol{f}}_2(\boldsymbol{x}) > \tilde{f}_2$, where $\tilde{f}_2$ is the best upper bound of the optimal value of (P_2^δ) known so far by the algorithm. Notice that this test can discard boxes in R^1_δ which are not of interest when solving (P_2^δ), but not boxes which can contain the minimum until they do not enter in the solution list. The updating of $\tilde{f}_2$ is done within this test, by the smallest $\overline{\boldsymbol{f}}_2(\boldsymbol{x})$ $(< \tilde{f}_2)$, provided that $\boldsymbol{x} \in R^1_\delta$ (that is, if $\boldsymbol{x}$ is to be sent to either $\mathcal{L}_S^{min}$ or $\mathcal{L}_S^{\delta}$). Furthermore, if $\tilde{f}_2$ is updated, then the test tries to discard any box in $\mathcal{L}_W^{\delta}$ and $\mathcal{L}_S^{\delta}$, and not only $\boldsymbol{x}$.

A few comments on Algorithm 3.1 are as follows. The box $\boldsymbol{x}$ to be chosen from $\mathcal{L}_W^{min}$ and $\mathcal{L}_W^{\delta}$ (Step 3 of Phase 1 and Phase 2) is the one with the lowest $\underline{\boldsymbol{f}}_j(\boldsymbol{x})$. We also want to point out that if we remove the cut-off test due to the second objective function, then the algorithm obtains the whole R^1_δ. In fact, if in

Algorithm 3.1 Phase 1.

1: Eval $f_j(s)$; $\tilde{f}_j = \overline{f}_j(s)$; $s \rightarrow \mathcal{L}_W^{min}$;
2: **while** ($\mathcal{L}_W^{min} \neq \emptyset$) **do**
3: $\mathcal{L}_W^{min} \rightarrow x$;
4: **if** (MonoTestFeas(x, s))
5: **if** (**not** RejectBySecObj(x, $\tilde{f}_2$, NoUpd))
6: $x \rightarrow \mathcal{L}_W^{\delta}$;
7: **continue**;
8: Eval CenteredForm(x, c);
9: CutOffTest (c, $\mathcal{L}$, $\tilde{f}_j$, δ);
10: **if** ($f_j(x) > \tilde{f}_j$)
11: **if** ($\underline{f}_j(x) \leq \tilde{f}_j * (1+\delta)$)
12: **if** (**not** RejectBySecObj(x, $\tilde{f}_2$, NoUpd))
13: $x \rightarrow \mathcal{L}_W^{\delta}$;
14: **continue**;
15: Prune(x, $\tilde{f}_j$, ε, δ) $\rightarrow x^1, x^2$;
16: **for** $i = 1, 2$ **do**
17: **if** (**not** Feasible(x^i))
18: **continue**;
19: Eval $f_j(x^i)$;
20: $f_j(x^i) \cap = f_j(c) + (x^i - c)^T g^j(x)$;
21: **if** ($f_j(x^i) > \tilde{f}_j$)
22: **if** ($\underline{f}_j(x^i) \leq \tilde{f}_j * (1+\delta)$)
23: **if** (**not** RejBySecObj(x^i, $\tilde{f}_2$,NoUpd))
24: $x^i \rightarrow \mathcal{L}_W^{\delta}$;
25: **continue**;
26: **if** (w(x^i) $> \varepsilon$ & w($f_j(x^i)$) $> \varepsilon$)
27: $x^i \rightarrow \mathcal{L}_W^{min}$;
28: **else**
29: **if** (**not** RejectBySecObj(x^i, $\tilde{f}_2$, Upd))
30: $x^i \rightarrow \mathcal{L}_S^{min}$;
31: **endfor**
32: **endwhile**

Phase 2.

1: $\mathcal{L}_S^{min} \rightarrow \mathcal{L}_S^{\delta}$
2: **while** ($\mathcal{L}_W^{\delta} \neq \emptyset$) **do**
3: $\mathcal{L}_W^{\delta} \rightarrow x$;
4: **if** ($\overline{f}_j(x) < \tilde{f}_j * (1+\delta)(1+\eta)$)
5: $x^i \rightarrow \mathcal{L}_S^{\delta}$; **continue**;
6: Eval CenteredForm(x, c);
7: **if** ($\overline{f}_j(x) < \tilde{f}_j * (1+\delta)(1+\eta)$)
8: $x^i \rightarrow \mathcal{L}_S^{\delta}$; **continue**;
9: **if** ($\underline{f}_j(x) > \tilde{f}_j * (1+\delta)$)
10: **continue**;
11: Prune(x, $\tilde{f}_j$, ε, δ) $\rightarrow x^1, x^2$;
12: **for** i=1,2 **do**
13: **if** (**not** Feasible(x^i))
14: **continue**;
15: Eval $f_j(x^i)$;
16: $f_j(x^i) \cap = f_j(c) + (x^i - c)^T g^j(x)$;
17: **if** ($\underline{f}_j(x^i) > \tilde{f}_j * (1+\delta)$))
18: **continue**;
19: **if** ($\overline{f}_j(x^i) < \tilde{f}_j * (1+\delta)(1+\eta)$)
20: **if** (**not** RejectBySecObj(x^i, $\tilde{f}_2$, Upd))
21: $x^i \rightarrow \mathcal{L}_S^{\delta}$;
22: **continue**;
23: **if** (w(x^i) $> \varepsilon$ & w($f_j(x^i)$) $> \varepsilon$)
24: **if** (**not** RejectBySecObj(x^i, $\tilde{f}_2$,NoUpd))
25: $x^i \rightarrow \mathcal{L}_W^{\delta}$;
26: **continue**;
27: **else**
28: **if** (**not** RejectBySecObj(x^i, $\tilde{f}_2$, Upd))
29: $x^i \rightarrow \mathcal{L}_S^{\delta}$;
30: **continue**;
31: **endfor**
32: **endwhile**

addition we also remove the δ-pruning test (and we instead perform a simple bisection perpendicular to the direction of maximum width) and the δ-monotonicity test, we have an interval version of the GBSSS method.

3.2.2 Obtaining δ-lexicographic solutions

The algorithm for obtaining the δ-lexicographic solutions is now rather simple to explain. It consists of two steps. The first step is to call the full Algorithm 3.1 using $j = 1$, i.e. f_1 as the objective function. Then all the resulting boxes in $\mathcal{L}_S^{\delta}$ are moved to $\mathcal{L}_W^{min}$, and δ is set equal to 0 and as a second step, one again calls the first phase of Algorithm 3.1 (without its first line), now using $j = 2$, i.e. f_2 as the objective function. The solution list $\mathcal{L}_S^{min}$ resulting after this second call is the desired list of boxes containing any δ-lexicographic solution of problem (3.1).

3.2.3 Obtaining (δ, θ)-lexicographic solutions

The above presented δ-lexicographic method can be used to obtain $R^2_{\theta(\delta)}$. The only changes to be made to obtain the (δ, θ)-lexicographic method are the following:

1. In the 'cut-off test due to the second objective function', now we can discard a box $\boldsymbol{x}$ provided that $\underline{f}_2(\boldsymbol{x}) > \tilde{f}_2(1+\theta)$ (where $\tilde{f}_2$ is the best upper bound on the optimal value of (P_2^δ) known by the algorithm), and

2. In the second call of Phase 1 we have to pass the parameter $\delta = \theta$ instead of 0 as the size for the region of θ-optimality for problem (P_2^δ), and here the second phase has to be called again as well. The solution list $\mathcal{L}_S^\delta$ contains any (δ, θ)-lexicographic solution of problem (3.1).

Computational experiments on the δ-lexicographic and (δ, θ)-lexicographic method can be seen in Section 7.2 and 8.2 on some facility location problems.

3.3 Constraint-like method

In this section a new method for obtaining an outer approximation of the complete efficient set of nonlinear biobjective optimization problems is presented. It is based on the well known *constraint method*, and obtains a superset of the efficient set by computing the regions of δ-optimality of a finite number of single objective constraint problems.

The approach in this section is completely different from the one in the previous section. Instead of offering a (small) subset of the efficient set to the decision maker, we offer a *superset* which tightly contains the *complete* efficient set. The efficient set might be described analytically as a closed formula, numerically as a set of points, or in mixed form as a parametrized set of points. Unfortunately, as pointed out in [131], for the majority of multiobjective optimization problems, it is not easy to obtain such a description, since the efficient set typically includes a very large number or infinite number of points. The methods proposed in literature with that purpose are specialized either for particular problems (for instance, in [113], it is shown how to obtain the whole efficient set of some location problems) or for a particular class of multiobjective problems (for instance, the multiobjective simplex methods for the linear case [57]). To the extent of our knowledge, this is the only general method proposed in literature with that purpose (apart from the biobjective interval B&B method described in the next section). The reason for this lack of methods is that even obtaining a single efficient point of a nonlinear biobjective problem can be a difficult task. That is why some authors have proposed to present to the decision maker a 'representative set' of efficient points which suitably represent the whole efficient set (either by modifying the definition of efficiency [9] or by selecting a finite set of efficient points with the criteria of coverage, uniformity and cardinality as quality measures [132]) or an *approximation* of the efficient set by means of sets with a simpler structure (see [131] for a survey of methods with that aim).

The two methods presented in this thesis which obtain the whole (weakly) efficient set make use of interval tools (see Section 1.4). They offer a *superset* which tightly contains the (weakly) efficient set. By drawing in the image space that superset, the decision maker can easily see the trade-off between the two objectives, i.e., how one objective improves as the other gets worse. Something similar can be done in the decision space, by drawing the superset in a color scale depending on the objective value of one of the objectives.

3.3.1 The constraint method

There is a variety of methods for finding efficient points of nonlinear multiobjective optimization problems (see [38, 103] and the references therein), e.g., weighting method, lexicographic method,..., but among them, only a few (e.g., the constraint method, reference point methods,...) are able to detect *all* nondominated points. The constraint method is probably the most famous among them. The rationale behind the constraint method is rather simple. One of the objective functions, say f_1, is selected

to be minimized, whereas the other one, f_2, is converted into a constraint by setting an upper bound f_2^{ub} to it (we always minimize f_1 subject to a constraint on f_2, but a similar process can be developed interchanging the functions). The single objective problem to be solved, called *constraint problem*, is then

$$\begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & f_2(x) \leq f_2^{ub} \\ & x \in S \end{array} \tag{3.2}$$

The constraint method has very good properties which can be seen in the following theorems (for a proof, see for instance [103]).

Theorem 3.11. *The solution for the constraint problem (3.2) is weakly efficient.*

Theorem 3.12. *A feasible vector $x \in S$ is efficient if and only if it is a solution of the constraint problems*

$$\begin{array}{ll} \min & f_1(y) \\ s.t. & f_2(y) \leq f_2(x) \\ & y \in S \end{array} \quad \textit{and} \quad \begin{array}{ll} \min & f_2(y) \\ s.t. & f_1(y) \leq f_1(x) \\ & y \in S \end{array} .$$

Instead of having to solve those two constraint problems in Theorem 3.12, efficiency can also be guaranteed with only one of them, provided that it has a unique solution, as the following theorem shows.

Theorem 3.13. *A feasible vector $x \in S$ is efficient if it is a unique solution of any of the constraint problems in Theorem 3.12.*

From the previous theorems it can be concluded that it is possible to find *all* the efficient solutions of *any* biobjective optimization problem by the constraint method. However, we need to solve one or two problems to find *one* nondominated point, which means that if the nondominated set is not a discrete set (as is usually the case in continuous multiobjective optimization) then the method is not practical for finding the complete efficient set.

3.3.2 The constraint-like method

As explained above, with the classical constraint method we have to solve one (or two) constraint problems of type (3.2) in order to be able to obtain a single nondominated point. However, if in addition to solving (3.2) to optimality we compute its region of δ-optimality (see Section 1.5), then R_δ contains a *portion* of the efficient set whose values in the criterion space are close to the corresponding non-dominated point. The idea of the constraint-like method to obtain a superset containing the whole efficient set is simply this: considering that Z_N is plotted in the criterion space, we sweep the nondominated set from, (say), top to bottom by obtaining the regions of δ-optimality of a finite sequence of constraint problems, whose type is a modification of (3.2), by choosing appropriate upper bounds f_2^{ub} for the f_2 function. Next, we give the details.

3.3.2.1 The method

The basic type of constraint problems that we will use in our method is exactly the same as the one used in the classical constraint method. We will rewrite it as

$$(P_i) \quad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & f_2(x) \leq f_2^{(i)} \\ & x \in S \end{array}$$

where $f_2^{(i)}$ is a given constant defined below. Let $\hat{x}^{(i)}$ denote an optimal solution of (P_i), $i \geq 0$, and let $R_\delta^{(i)}$ be the region of δ-optimality of (P_i), that is,

$$R_\delta^{(i)} = \{x \in S : f_2(x) \leq f_2^{(i)}, f_1(x) - f_1(\hat{x}^{(i)}) \leq \delta \cdot |f_1(\hat{x}^{(i)})|\}.$$

The feasible set of (P_i) is $Y^{(i)} = \{x \in S : f_2(x) \leq f_2^{(i)}\}$.

In the first problem that we will consider, (P_0), we set $f_2^{(0)} = +\infty$. Thus, problem (P_0) is in fact the single objective problem

$$\begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & x \in S \end{array}.$$

Once we have solved problem (P_i) and have obtained an outer approximation $OR_\delta^{(i)}$ of $R_\delta^{(i)}$ with the help of the procedure mentioned in Section 1.5, the constant $f_2^{(i+1)}$ for the next problem (P_{i+1}) is given by

$$f_2^{(i+1)} = \min\{\overline{F}_2(Q) : Q \in LIR_\delta^{(i)} \cup LOR_\delta^{(i)}, Q \cap R_\delta^{(i)} \neq \emptyset\}, \tag{3.3}$$

where $\overline{F}_2(Q)$ is an upper bound on all objective values of f_2 found within the subset Q (see Figure 3.2), $LIR_\delta^{(i)}$ and $LOR_\delta^{(i)}$ are lists of subsets such that $LIR_\delta^{(i)}$ contains an inner approximation of $R_\delta^{(i)}$, while $LIR_\delta^{(i)} \cup LOR_\delta^{(i)}$ contains an outer approximation of $R_\delta^{(i)}$ (see Section 1.5). However, from a computational point of view, it can be better to set

$$f_2^{(i+1)} = \min\{\overline{F}_2(Q) : Q \in LIR_\delta^{(i)}, Q \cap Y^{(i)} \neq \emptyset\} \tag{3.4}$$

although this is a worse (higher) value than the one obtained with (3.3). Using (3.4) we only have to check whether a subset Q in $LIR_\delta^{(i)}$ contains at least one feasible point. If so, we take that set into account for calculating the minimum in (3.4). Moreover, if we know a feasible point y in Q, and also an upper bound of its function value $f_2(y)$, we can use that in (3.4) instead of $\overline{F}_2(Q)$.

Let us denote by $Q^{(i)}$ the subset at which the previous minimum (either (3.3) or (3.4)) is attained, i.e, $f_2^{(i+1)} = \overline{F}_2(Q^{(i)})$.

Lemma 3.14. $f_1(\hat{x}^{(i+1)}) \leq f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$.

Proof. Since $f_2^{(i+1)} = \overline{F}_2(Q^{(i)})$, we have, in particular, that all the points in $Q^{(i)} \cap S$ are feasible for (P_{i+1}). On the other hand, $Q^{(i)}$ contains at least one point $\check{x}$ in $R_\delta^{(i)}$, and thus, its objective value $f_1(\check{x})$ is less than or equal to $f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$. Therefore, the optimal value of (P_{i+1}), $f_1(\hat{x}^{(i+1)})$, must be less than or equal to $f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$. □

Unfortunately, the use of this type of constraint problems is not enough to guarantee that the algorithm obtains an outer approximation of the whole efficient set. We also need a second type of constraint problems, namely,

$$(\check{P}_{i+j}) \quad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & f_2(x) \leq f_2^{(i)} \\ & x \in S \\ & f_1(x) \geq f_1(\hat{x}^{(i)}) + j \cdot \delta|f_1(\hat{x}^{(i)})|. \end{array}$$

Notice that whereas the constraint in the problems of type (P_i) forces to the image of the feasible set of the problems to go down in the criterion space as i increases, the additional constraint in the problems of type $(\check{P}_{i+j})$ forces it to go to the right as j increases (see Figure 3.2).

The method that we propose to obtain an outer approximation of the efficient set of (3.1) is the following (see Figure 3.2).

Constraint-like method (CLM)

1. $i \leftarrow 0$, $f_2^{(0)} \leftarrow +\infty$.
2. While (P_i) is feasible
   (a) Solve (P_i) and obtain an outer approximation $OR_\delta^{(i)}$ of $R_\delta^{(i)}$ using the procedure mentioned in Subsection 1.5.
   (b) Calculate $f_2^{(i+1)}$ as given by (3.3) or (3.4).
   (c) If $f_2^{(i+1)} \not< f_2^{(i)}$ then
      i. $j \leftarrow 0$.
      ii. Repeat
         A. $j \leftarrow j+1$.
         B. Solve the problem $(\check{P}_{i+j})$ and obtain an outer approximation $OR_\delta^{(i+j)}$ of its region of δ-optimality $R_\delta^{(i+j)}$ using the procedure mentioned in Subsection 1.5.
         C. Calculate $f_2^{(i+j+1)}$ as given by (3.3) or (3.4).
      until $f_2^{(i+j+1)} < f_2^{(i)}$ or the problem $(\check{P}_{i+j})$ is infeasible.
      iii. If $f_2^{(i+j+1)} < f_2^{(i)}$ then set $i \leftarrow i+j+1$.
      iv. If the problem $(\check{P}_{i+j})$ is infeasible then set $i \leftarrow i+j$ and go to step 3.
   else $i \leftarrow i+1$.
3. $\bigcup_{j=0}^{i-1} OR_\delta^{(j)}$ contains the efficient set of (3.1).

Theorem 3.15. *If CLM stops after a finite number of iterations, then it obtains all the efficient solutions of (3.1).*

Proof. From Theorem 3.12, a necessary condition for a feasible vector to be efficient is that it must be a solution for a problem in the form

$$(P_{aux}) \quad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & f_2(x) \leq f_2^{(aux)} \\ & x \in S \end{array}$$

We shall prove that all the optimal solutions for problems of that type lie in a region of δ-optimality of one of the problems of type (P_i) or $(\check{P}_{i+j})$ solved by CLM. Let $\hat{x}^{(aux)}$ be an optimal solution of (P_{aux}).

A: Let us suppose that $f_2^{(i+1)} < f_2^{(aux)} < f_2^{(i)}$ (if $f_2^{(i+1)} = f_2^{(aux)}$ then (P_{aux}) coincides with (P_{i+1}) and if $f_2^{(i)} = f_2^{(aux)}$ then (P_{aux}) coincides with (P_i); in either case, the result is trivial). We shall prove that $\hat{x}^{(aux)} \in R_\delta^{(i)}$.

- Notice that $f_1(\hat{x}^{(aux)}) \geq f_1(\hat{x}^{(i)})$, since the feasible set of (P_i) contains the feasible set of (P_{aux}) (remember that we are supposing that $f_2^{(aux)} < f_2^{(i)}$).
- Analogously, $f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(i+1)})$.
- On the other hand, from Lemma 3.14, $f_1(\hat{x}^{(i+1)}) \leq f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$.

Thus, $f_1(\hat{x}^{(i)}) \leq f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$, that is, $\hat{x}^{(aux)} \in R_\delta^{(i)}$. The result is derived from the fact that $R_\delta^{(i)} \subseteq OR_\delta^{(i)}$.

Notice that $f_2^{(aux)} > f_2^{(0)}$ cannot hold, since $f_2^{(0)} = +\infty$. And if (P_{last}) is the last problem solved by the algorithm, i.e., (P_{last}) is infeasible, then $f_2^{(aux)} < f_2^{(last)}$ cannot hold either, since then (P_{aux}) will be infeasible too. So, no efficient point is lost in this case by the algorithm.

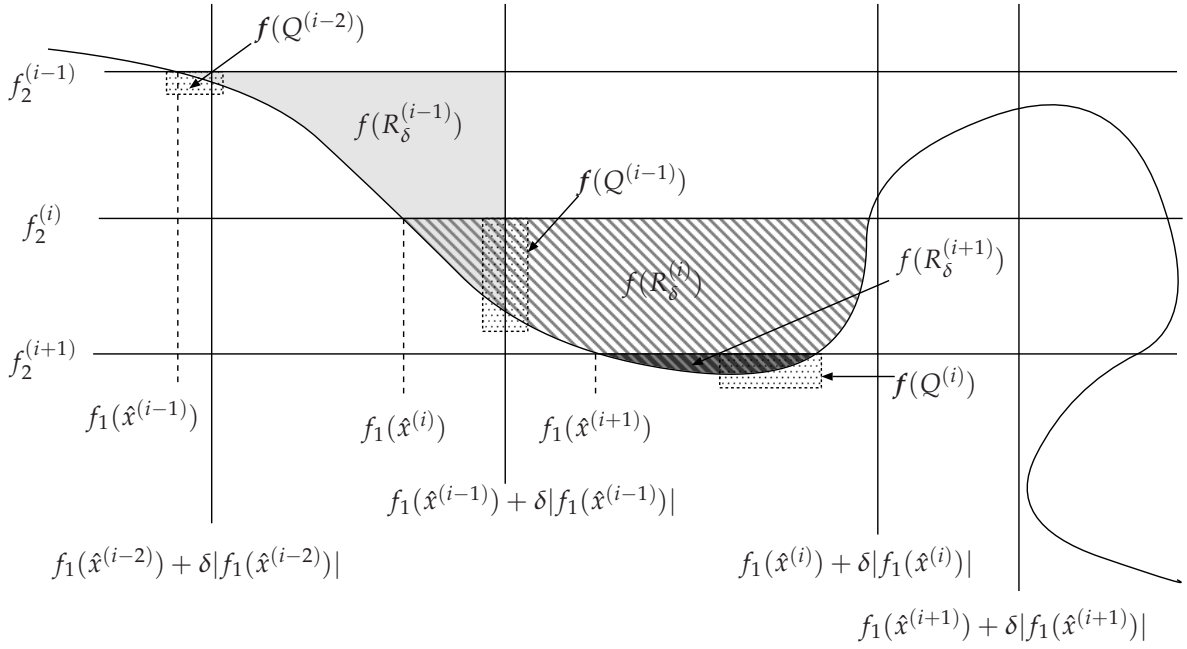


Figure 3.2: Image space of a biobjective problem using CLM. For the sake of clarity, for each problem (P_i) we have only drawn the interval bounded image of one of the sets forming $OR_\delta^{(i)}$, namely, $\boldsymbol{f}(Q^{(i)})$. The light gray region is $f(R_\delta^{(i-1)})$, the striped region is $f(R_\delta^{(i)})$, and the dark gray region (that is totally covered by the striped one) is $f(R_\delta^{(i+1)})$.

B: Let us suppose that $f_2^{(i+1)} \geq f_2^{(i)}$.

B.1: Consider the case in which $f_2^{(i+1)} \geq f_2^{(i)} > f_2^{(aux)}$ holds and the algorithm has to solve k problems of type $(\check{P}_{i+j})$, $k \geq 1$, before the condition $f_2^{(aux)} > f_2^{(i+k+1)}$ holds. In this case $f_1(\hat{x}^{(i)}) \leq f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(i+k+1)})$ (notice that both (P_i) and (P_{i+k+1}) are problems of the same type). Let us suppose that $f_1(\hat{x}^{(i+r)}) < f_1(\hat{x}^{(aux)}) < f_1(\hat{x}^{(i+r+1)})$, $1 \leq r \leq k$ (notice that the optimal objective values of the problems of type $(\check{P}_{i+j})$ are non-decreasing; also, as before, we can assume strict inequalities). We shall prove that $\hat{x}^{(aux)} \in R_\delta^{(i+r)}$. Firstly, notice that $\hat{x}^{(aux)}$ is a feasible point of $(\check{P}_{i+r})$: it is a point in S, $f_2(\hat{x}^{(aux)}) \leq f_2^{(aux)} < f_2^{(i)}$, and $f_1(\hat{x}^{(aux)}) \geq f_1(\hat{x}^{(i+r)}) \geq f_1(x^{(i)}) + r \cdot \delta|f_1(x^{(i)})|$. And secondly, it is in $R_\delta^{(i+r)}$, since $f_1(\hat{x}^{(i+r)}) \leq f_1(\hat{x}^{(aux)})$ and $f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(i+r)}) + \delta|f_1(\hat{x}^{(i+r)})|$: if $f_1(\hat{x}^{(aux)}) > f_1(\hat{x}^{(i+r)}) + \delta|f_1(\hat{x}^{(i+r)})|$, since $f_1(\hat{x}^{(i+r)}) + \delta|f_1(\hat{x}^{(i+r)})| \geq f_1(\hat{x}^{(i)}) + r \cdot \delta|f_1(\hat{x}^{(i)})| + \delta \cdot |f_1(\hat{x}^{(i)}| = f_1(\hat{x}^{(i)}) + (r+1) \cdot \delta|f_1(\hat{x}^{(i)})|$, the point $\hat{x}^{(aux)}$ will be feasible for the problem $(\check{P}_{i+r+1})$, but this is not possible, since $f_1(\hat{x}^{(aux)}) < f_1(\hat{x}^{(i+r+1)})$ and $\hat{x}^{(i+r+1)}$ is an optimal solution for $(\check{P}_{i+r+1})$.

B.2: Suppose now that $f_2^{(i+1)} > f_2^{(aux)} > f_2^{(i)}$ (as before, we can suppose that the inequalities are strict). In this case, $f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(i)})$, since the feasible set of (P_{aux}) contains the feasible set of (P_i) (now we are supposing that $f_2^{(aux)} > f_2^{(i)}$). Then, either there exists an index j, $j \leq i-1$, such that $f_2^{(j+1)} < f_2^{(aux)} < f_2^{(j)}$ (and then, as in case A, $\hat{x}^{(aux)} \in R_\delta^{(j)}$) or there exists an index j, $j < i-1$, such that $f_2^{(j+1)} \geq f_2^{(j)} > f_2^{(aux)}$ holds and the algorithm has to solve k' problems of type $(\check{P}_{i+j})$, $k' \geq 1$, before the condition $f_2^{(aux)} > f_2^{(j+k'+1)}$ holds (and then, subcase B.1 applies). In either case, $x^{(aux)}$ will be in the region of δ-optimality of one of the problems solved by the algorithm. □

Notice that we need problems of type $(\check{P}_{i+j})$ to avoid the algorithm getting stuck when $f_2^{(i+1)} \geq f_2^{(i)}$. This may happen when the nondominated set is not connected and the *jump* (along the abscissas axis) is greater than $\delta|f_1(\hat{x}^{(i)})|$. In Figure 3.2, without problems of type $(\check{P}_{i+j})$ CLM would get stuck after solving problem (P_{i+1}), because the constraint on f_2 does not change anymore, and $R_\delta^{(i+j)} = R_\delta^{(i+1)}, \forall j > 1$. The method would also get stuck when there is a continuum of weakly efficient points with the same f_2-value (i.e. in the image space the weakly efficient points form a segment parallel to the abscissas axis, being the length of that segment greater than $\delta|f_1(\hat{x}^{(i)})|$).

On the other hand, in some pathological cases it may happen, at least theoretically, that the algorithm does not stop. For instance, we may have a biobjective problem in which $f_2^{(i)} = \frac{1}{i}$ but in which the optimal value of the single objective problem

$$\begin{array}{ll} \min & f_2(x) \\ \text{s.t.} & x \in S \end{array} \tag{3.5}$$

is less than or equal to 0. In that case, CLM always uses problems of type (P_i), but the algorithm cannot sweep the nondominated set until the bottom: it reaches at most $\lim_{i\to\infty} f_2^{(i)}$.

3.3.2.2 Using a single type of constraint problems

In CLM, in addition to constraint problems of the form (P_i) we also needed problems of the form $(\check{P}_{i+j})$ to be able to avoid that the algorithm gets stuck. In this section we present a modification of CLM which can always find an enclosure of the whole efficient set in finite steps. It only uses one type of constraint problems, denoted by $(\bar{P}_i)$, which is a simplified version of type $(\check{P}_{i+j})$, namely,

$$(\bar{P}_i) \quad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & f_2(x) \leq f_2^{(i)} \\ & f_1(x) \geq f_1(\hat{x}^{(i-1)}) + \delta|f_1(\hat{x}^{(i-1)})| \\ & x \in S. \end{array}$$

Here, again, $\hat{x}^{(i)}$ denotes an optimal solution of $(\bar{P}_i)$.

Similarly as in the previous algorithm, the first problem that we will consider, $(\bar{P}_0)$, is again the single objective problem

$$\begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & x \in S. \end{array}$$

Once we have solved problem $(\bar{P}_i)$ and have obtained a minimizer point $\hat{x}^{(i)}$ and an outer approximation $OR_\delta^{(i)}$ of its region of δ-optimality $R_\delta^{(i)}$ with the procedure mentioned in Subsection 1.5, we compute

$$f_2^{(i+1)} = \min\{f_2^{(i)}, \min\{\overline{F}_2(Q) : Q \in LIR_\delta^{(i)} \cup LOR_\delta^{(i)}, Q \cap R_\delta^{(i)} \neq \emptyset\}\} \tag{3.6}$$

or

$$f_2^{(i+1)} = \min\{f_2^{(i)}, \min\{\overline{F}_2(Q) : Q \in LIR_\delta^{(i)}, Q \cap Y^{(i)} \neq \emptyset\}\}. \tag{3.7}$$

With these ingredients, the modified method for obtaining an outer approximation of the efficient set of (3.1) is the following (see Figure 3.3):

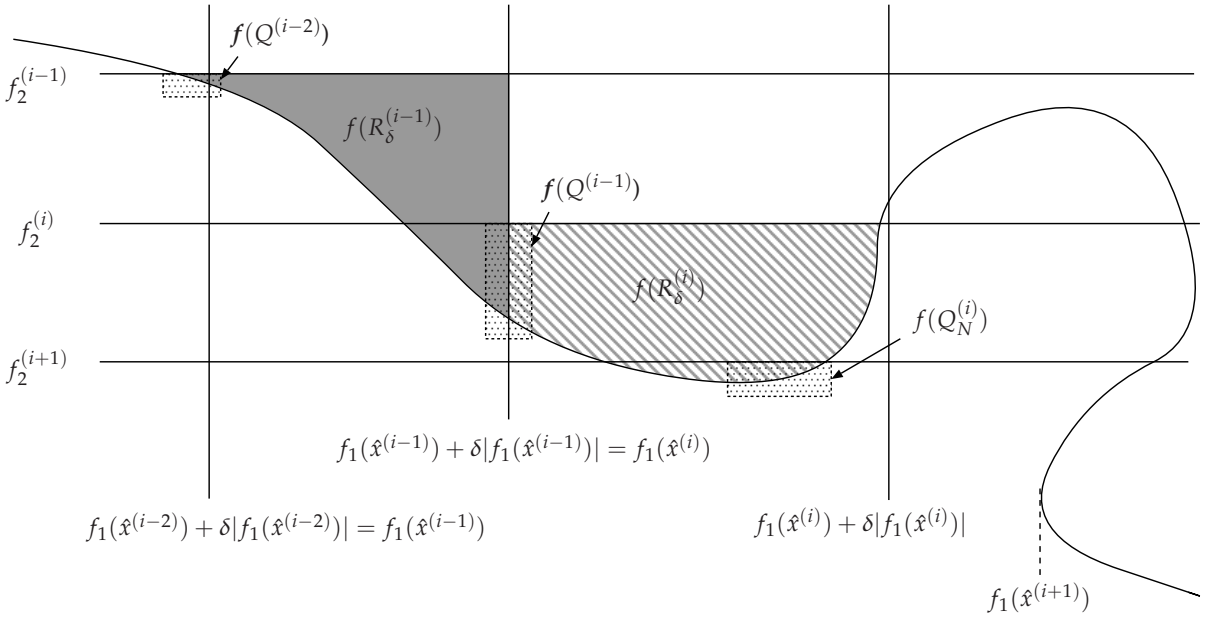


Figure 3.3: Image space of a biobjective problem using MCLM. The gray region is the image of $R_\delta^{(i-1)}$, $f(R_\delta^{(i-1)})$, and the striped region is the image of $R_\delta^{(i)}$, $f(R_\delta^{(i)})$. $\boldsymbol{f}(Q^{(i)})$ denotes the interval bounded image of a subset $Q^{(i)}$ at which the minimum (3.3) or (3.4) is attained, i.e, a subset such that $f_2^{(i+1)} = \overline{F}_2(Q^{(i)})$.

Modified constraint-like method (MCLM)

1. $i \leftarrow 0$.
2. While $(\bar{P}_i)$ is feasible
   (a) Solve $(\bar{P}_i)$ and obtain an outer approximation $OR_\delta^{(i)}$ of its region of δ-optimality $R_\delta^{(i)}$ using the procedure mentioned in Subsection 1.5.
   (b) Calculate $f_2^{(i+1)}$ as given by (3.6) or (3.7).
   (c) $i \leftarrow i + 1$.
3. $\bigcup_{j=0}^{i-1} OR_\delta^{(j)}$ contains the efficient set of (3.1).

Theorem 3.16. *Suppose that both the efficient set and the nondominated set of (3.1) are bounded. Suppose also that $f_1(\hat{x}^{(0)}) > 0$. Then MCLM obtains the complete efficient set of (3.1) in a finite number of steps.*

Proof. From Theorem 3.12, a necessary condition for a feasible vector to be efficient is that it must be a solution for a problem of the form

$$(P_{aux}) \quad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & f_2(x) \leq f_2^{(aux)} \\ & x \in S. \end{array}$$

We shall prove that all the optimal solutions for problems of that type which are efficient lie in a region of δ-optimality of one of the problems of type $(\bar{P}_i)$ solved by MCLM. Let $\hat{x}^{(aux)}$ be an optimal solution of (P_{aux}).

Notice that the sequence $\{f_1(\hat{x}^{(i)})\}$ is non-decreasing. Thus, one of the following three cases must happen:

A: $f_1(\hat{x}^{(0)}) \leq f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(0)}) + \delta|f_1(\hat{x}^{(0)})|$. It means that $\hat{x}^{(aux)}$ is in the region of δ-optimality of $(\bar{P}_0)$.

B: There is an index i, $1 \leq i \leq last - 1$, such that $f_1(\hat{x}^{(i-1)}) + \delta|f_1(\hat{x}^{(i-1)})| \leq f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$ ($last - 1$ is the index of the last feasible problem considered by the algorithm). Two subcases can happen:

B.1: $f_2(\hat{x}^{(aux)}) \leq f_2^{(i)}$. Then the point $\hat{x}^{(aux)}$ is a feasible point of $(\bar{P}_i)$. In particular, $f_1(\hat{x}^{(aux)}) \geq f_1(\hat{x}^{(i)})$. Furthermore, by assumption, $f_1(\hat{x}^{(aux)}) \leq f_1(\hat{x}^{(i)}) + \delta|f_1(\hat{x}^{(i)})|$. Thus, $\hat{x}^{(aux)}$ is in the region of δ-optimality of $(\bar{P}_i)$.

B.2: $f_2(\hat{x}^{(aux)}) > f_2^{(i)}$. Then, there must exist a point $\check{x} \in R_\delta^{(i-1)}$ such that $f_2(\check{x}) \leq f_2^{(i)} < f_2(\hat{x}^{(aux)})$ and $f_1(\check{x}) \leq f_1(\hat{x}^{(i-1)}) + \delta|f_1(\hat{x}^{(i-1)})| \leq f_1(\hat{x}^{(aux)})$. That is, $\check{x}$ dominates $\hat{x}^{(aux)}$, i.e., $\hat{x}^{(aux)}$ is not an efficient point for problem (3.1). Thus, in this case, we do not have to care about if $\hat{x}^{(aux)}$ is in the region of δ-optimality of one of the problems solved by the algorithm.

C: $f_1(\hat{x}^{(aux)}) > f_1(\hat{x}^{(last-1)}) + \delta|f_1(\hat{x}^{(last-1)})|$. Two cases can happen:

C.1: If $f_2(\hat{x}^{(aux)}) > f_2^{(last)}$ then $\hat{x}^{(aux)}$ will not be an efficient point (similarly as in subcase B.2).

C.2: If $f_2(\hat{x}^{(aux)}) \leq f_2^{(last)}$ then $\hat{x}^{(aux)}$ will be a feasible point of $(\bar{P}_{last})$, but this is a contradiction, since $(\bar{P}_{last})$ is infeasible (it was the problem provoking the termination of the algorithm).

Notice that $f_1(\hat{x}^{(aux)}) < f_1(\hat{x}^{(0)})$ cannot happen, since the feasible set of $(\bar{P}_0)$ contains the feasible set of (P_{aux}).

Thus, in any case, if $\hat{x}^{(aux)}$ is an efficient point, then it lies in the region of δ-optimality of one of the problems solved by the algorithm.

To prove that the algorithm stops after a finite number of steps, just notice that with problem $(\bar{P}_i)$ we sweep $\delta|f_1(\hat{x}^{(i)})|$ units length along the f_1 axis on Z_N in the criterion space. Thus, if we denote by $\hat{y}^{(0)}$ an optimal solution for problem (3.5), we need at most

$$\frac{f_1(\hat{y}^{(0)}) - f_1(\hat{x}^{(0)})}{\delta f_1(\hat{x}^{(0)})}$$

problems to sweep the whole nondominated set. □

Remark 3.17. Notice that in addition to the constraint on f_2 employed in the classical constraint method, we have a second constraint on f_1. We need to add this second constraint, because otherwise, the algorithm may get stuck when $f_2^{(i+1)} = f_2^{(i)}$. This may happen when the nondominated set is not connected and the *jump* (along the abscissa) is greater than $\delta|f_1(\hat{x}^{(i)})|$ or when there is a continuum of weakly efficient points with the same f_2-value (i.e., in the image space the weakly efficient points form a segment parallel to the axis, the length of that segment being greater than $\delta|f_1(\hat{x}^{(i)})|$).

Also, with the same δ, the quality of the approximation can vary depending on the shape of the nondominated set. See for instance Figure 3.4. For the sake of simplicity, in that figure, it is supposed that the shape of the nondominated set is linear and that the regions of δ-optimality (triangles in the example) do not overlap. Whereas in Figure 3.4(a) the overestimation provided by the algorithm has an area of 2 units, in Figure 3.4(b) the overestimation is only 1 unit.

But the quality can also vary depending on the objective function selected to be minimized in the constrained problems. Consider for instance again Figure 3.4, but supposing now that Figure 3.4(a) corresponds to the image space of a part of the nondominated set when the f_1 values are plotted in the abscissas axis and the f_2 values in the ordinate axis and Figure 3.4(b) when the f_1 values are plotted

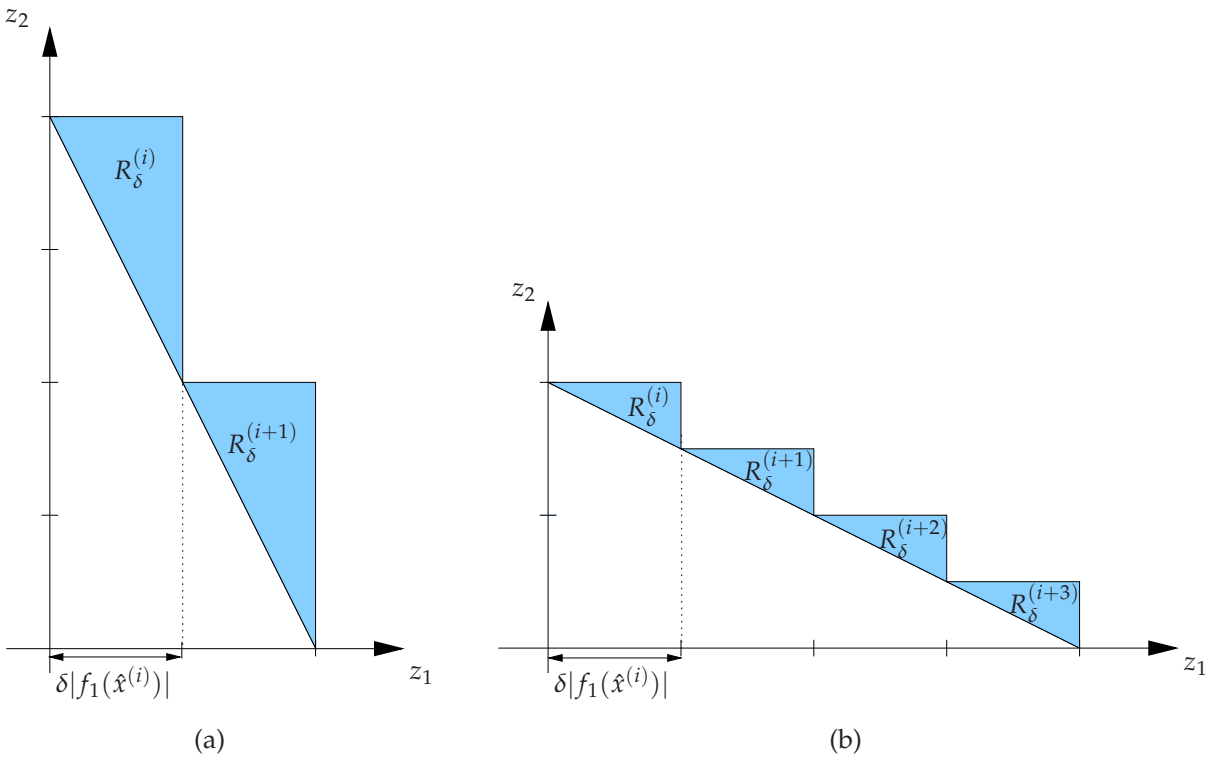


Figure 3.4: In (a) the overestimation is $2 \cdot 1 = 2$ units and in (b) only $4 \cdot 0.25 = 1$.

in the ordinate axis and the f_2 values in the abscissas axis. We can see clearly that in this example we obtain a better approximation of the nondominated set by minimizing f_2 in the constrained problems. As a rule, we could say that in the constraint problems it is better to select, as the objective function to be minimized, the one whose range in the nondominated set is larger.

Another way to improve the quality of the outer approximation of the efficient set is to remove from the regions of δ-optimality $OR_\delta^{(i)}$ all the subsets Q for which we can guarantee that they do not contain any efficient point (see for instance the multiobjective cut-off test described in the next section).

Both CLM and MCLM generate the nondominated set from top-left to bottom-right. If we give the decision maker the regions of δ-optimality as we compute them, it may happen that (s)he becomes satisfied with one of the efficient points already found during the execution of the algorithm. In that case, the process can be stopped prematurely, and we can save the time of computing the rest of the efficient set. This can be seen as an interactive method.

Remark 3.18. The constraint-like method obtains a superset of the weakly efficient set S_{WE} which maps into a superset of the weakly nondominated set Z_{WN}, which may be made as tight as required by reducing the value of δ (and the tolerances used in the procedure obtaining the regions of δ-optimality): the smaller the value of δ is, the better the quality of the approximation is, but also the higher the number of subproblems to be solved.

Remark 3.19. Problems $(\check{P}_{i+j})$ and $(\bar{P}_i)$ can be rewritten as

$$(\tilde{P}_i) \quad \begin{array}{ll} \min & f_1(x) \\ \text{s.t.} & f_2(x) \leq f_2^{(i)} \\ & x \in S \setminus \cup_{j=1}^{i-1} OR_\delta^{(j)} \end{array}$$

where instead of the constraint on f_1, we remove, from S, the outer approximations of the regions of δ-optimality of the problems already solved. In fact, if we use this kind of problem instead of problems (P_i) as well, the two methods become the same.

Remark 3.20. The condition $f_1(\hat{x}^{(0)}) > 0$ in Theorem 3.16 is not restrictive. If a function f_1 does not satisfy it, we can use, for instance, the function $\hat{f}_1(x) = f_1(x) + |f_1(\hat{x}^{(0)})| + 1$ instead.

Remark 3.21. Notice that although the method that we have used to obtain the whole set R_δ when deriving (M)CLM is inspired by the GBSSS method (see Section 1.5), any other method which obtains the *complete* set R_δ (or an outer approximation of it) could be useful. If we denote by NOR_δ the outer approximation (or exact representation) of R_δ obtained by any other method, the only thing that has to be changed when using it in (M)CLM is the computation of the bounds $f_2^{(i)}$. If we denote by q any point in NOR_δ, the bounds should be computed as

$$f_2^{(i+1)} = \min\{f_2^{(i)}, \min\{f_2(q) : q \in NOR_\delta^{(i)} \cap R_\delta^{(i)}\}.$$

In fact, (3.3) and (3.4) are surrogates for the previous formula.

3.3.2.3 Carrying out the method: an interval implementation

Since the purpose of this section is to deal with general nonlinear biobjective problems, the constraint problems that we have to solve may be global optimization ones (they may have many local optima). Thus, we need to use global optimization techniques to cope with them. Furthermore, instead of merely solving the constraint problems, we must also obtain their region of δ-optimality. Among the global optimization techniques only a Branch and Bound scheme seems to be appropriate for our purposes, although the computation of bounds is a difficult task, too. In this section we present such a method, which has some similarities with the two-phase method mentioned in Subsection 1.5 and described in [120], although modified for our purposes and in the more general framework of *Interval Analysis*.

3.3.2.4 The algorithm

The implementation of the algorithm is described in pseudo-code form in Algorithm 3.2. We have considered the constraint problems to be written in the form of $(\tilde{P}_i)$. In this way, we only have to deal with the constraint on f_2, and will never have the one on f_1 (the removal of the regions of δ-optimality of the previous problems is done easily in our implementation, see Step 35 of Phase 2 in Algorithm 3.3).

In Algorithm 3.2, $\boldsymbol{s}$ denotes a box containing the feasible set S and it uses the following lists: $\mathcal{L}_W^{min}$ and $\mathcal{L}_W^{\delta}$ are the working lists of Phase 1 and Phase 2, to find the global minimum and its δ-optimal region, respectively, $\mathcal{L}_S^{cur}$ is the solution list of the current problem (both in Phase 1 and 2) and $\mathcal{L}_{Next}$ is the list where the boxes for the next problems are stored. We denote by $maxf_j$ is the maximum f_j-value that any efficient point can take, $j = 1, 2$, that is, $(maxf_1, maxf_2)$ is the *nadir point* of (3.1). As we can see, to calculate $maxf_1$ we first solve the problem (3.5) to optimality by calling Phase 1 of Algorithm 3.3 (see Step 3 of Algorithm 3.2) with $\delta = 0$ and using f_2 instead of f_1. Then, we compute $maxf_1 = \max\{\overline{f_1}(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{L}_S^{cur}\}$, where $\mathcal{L}_S^{cur}$ contains the global minimizers of (3.5). Something similar

Algorithm 3.2 Constraint-Like Method

Input: $\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{h}_l (l = 1, \ldots, q), \boldsymbol{s}, \delta, \varepsilon, \eta, \mu$
Output: $\mathcal{L}_{\mathcal{S}}$

1: $maxf_1 = maxf_2 = \infty$;
2: Eval $\boldsymbol{f}_2(\boldsymbol{s})$; $\tilde{f} = \overline{\boldsymbol{f}_2}(\boldsymbol{s})$; $\boldsymbol{s} \to \mathcal{L}_W^{min}$;
3: Call *Phase 1* using f_2 as f_1, and $\delta = 0$; $maxf_1 = \max\{\overline{\boldsymbol{f}_1}(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{L}_S^{cur}\}$;
4: Eval $\boldsymbol{f}_1(\boldsymbol{s})$; $\tilde{f} = \overline{\boldsymbol{f}_1}(\boldsymbol{s})$; $\boldsymbol{s} \to \mathcal{L}_W^{min}$;
5: Call *Phase 1*; $maxf_2 = \max\{\overline{\boldsymbol{f}_2}(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{L}_S^{cur}\}$;
6: **repeat**
7: *Phase 2*
8: *Phase 1*
9: **until** *Phase 1* terminates with no solution.

is done to calculate $max f_2$ (Step 5), although this time we do not modify the value of δ, since (P_0) is the first problem for which we have to compute its region of δ-optimality. After that, the algorithm (Steps 6 to 9) keeps calling Algorithm 3.3, the main procedure which obtains the regions of δ-optimality of the constraint problems, until one of the problems is infeasible.

Algorithm 3.3 is the core of Algorithm 3.2. It obtains the outer approximation of the region of δ-optimality of the constraint problems. The algorithm is inspired by the GBSSS method described in Section 1.5.

However, Algorithm 3.3 differs from the method in Section 1.5 in several points. Firstly, the bounds are computed here with the help of interval analysis. Secondly, in addition to the feasibility test described in Section 1.4 (with two implementations, Feasible and Infeasible, in the code) and the δ-Cut-off test (CutOffTest in the code, introduced in Section 3.2.1), we also use a variant of the δ-monotonicity test for feasible and undetermined boxes (MonoTest) (see Section 3.2.1), a pruning test described in 2.2.2 (Prune in the code) applied to f_2 or to both f_1 and f_2, and a multiobjective cut-off test (which discards dominated boxes).

We give a brief description on the discarding tests used here that were not explained before.

δ-cut-off test: Every time a box $\boldsymbol{x}$ is chosen from the list $\mathcal{L}_W^{min}$, and provided that its midpoint $\mathrm{m}(\boldsymbol{x})$ is feasible, we use $\overline{\boldsymbol{f}}_1(\mathrm{m}(\boldsymbol{x}))$ (previously computed to evaluate the centered form) to update (if possible, i.e. when $\overline{\boldsymbol{f}}_1(\mathrm{m}(\boldsymbol{x}) < \tilde{f}$) the best upper bound $\tilde{f}$ of the global minimum of the constraint problem (but see also the multiobjective cut-off test). If updated, then the CutOffTest($\mathrm{m}(\boldsymbol{x})$, $\mathcal{L}_W^{min} \cup \mathcal{L}_S^{cur}, \mathcal{L}_W^{\delta}, \delta$) sends $\mathcal{L}_W^{\delta}$ all the boxes $\boldsymbol{x}$ in $\mathcal{L}_W^{min}$ and $\mathcal{L}_S^{cur}$ such that $\underline{\boldsymbol{f}_1}(\boldsymbol{x}) > \tilde{f}$ (since they cannot contain the optimal value of the constraint problem), and then sends $\mathcal{L}_{Next}$ all the boxes $\boldsymbol{x}$ in $\mathcal{L}_W^{\delta}$ such that $\underline{\boldsymbol{f}_1}(\boldsymbol{x}) > \tilde{f}(1+\delta)$ (since they cannot be in $R_{\delta}^{(i)}$).

Pruning test applied to f_2: A modification of the pruning method described in Section 2.2.4 is used here. Due to the biobjective nature of the problem, some modifications are necessary. In particular, when applied to f_2, the upper bounding value $f_2^{(i)}$ is used, therefore the regions which can be pruned are infeasible for the current constraint problem as well as for the rest of constraint problems (since in the following constraint problems the upper bound on f_2 will be smaller than or equal to $f_2^{(i)}$).

Pruning test applied to f_1 and f_2: A similar process to the one described in the above pruning test can be done applying it to function f_1 considering $\tilde{f}(1+\delta)$ the upper bounding value. However, now, the parts of the actual box which are not of interest (because their f_1 value is greater than $\tilde{f}(1+\delta)$) have to be sent to $\mathcal{L}_{Next}$, because they may be of interest to the next constraint problems.

In order to apply the pruning technique to both f_1 and f_2 within the same algorithm without generating too many boxes, we have used the strategy which we explain with the following example (see Figure 3.5): Let us suppose that the gray region A can be deleted by pruning with f_2, while the striped area B can be cut by pruning with f_1. It is easy to see that cutting region A is always a good decision, because we generate, at most, two new subboxes, and the deleted regions do not have to be taken into consideration anymore (they are not of interest to any of the remaining problems). However, region $B \setminus A$ has to be sent to $\mathcal{L}_{Next}$, if we decide to cut it. That is why we only cut it if its area is greater than 10% of the area of the original box. Furthermore, when $B \setminus A$ is not connected ($A \subset B$), only its greater part is sent to $\mathcal{L}_{Next}$ (provided that the previous condition holds, i.e. its area is greater than 10% of the area of the original box). The selection of the coordinate direction to apply the test is done as usual (see Section 2.2.4).

δ-monotonicity test (for feasible and undetermined boxes): Since we are now computing $R_{\delta}^{(i)}$ and not solving the constraint problem till optimality, we cannot discard monotonous boxes. Instead, when MonoTest($\boldsymbol{x}, \boldsymbol{z}$) is true, i.e., when the objective function f_1 is monotonous at $\boldsymbol{x}$, the δ-monotonicity test sends box $\boldsymbol{x}$ to $\mathcal{L}_W^{\delta}$ (since it cannot contain the optimal value of the constraint problem), and follows the process with facet $\boldsymbol{z}$ of $\boldsymbol{x}$ containing its best points: $\boldsymbol{z}$ cannot be discarded since it can lie on the border of the feasible set of the constraint problem). Note that this test is only useful in Phase 1.

Algorithm 3.3 Main procedure

Phase 1

Input: $f_1, f_2, h_l, \mathcal{L}_W^{min}, \mathcal{L}_{PNS}, \delta, \varepsilon, maxf_1, maxf_2$
Output: $\mathcal{L}_S^{cur}, \mathcal{L}_W^{\delta}, \mathcal{L}_{Next}, \mathcal{L}_{PNS}, \tilde{f}$

while ($\mathcal{L}_W^{min} \neq \emptyset$) **do**
  $\mathcal{L}_W^{min} \rightarrow \boldsymbol{x}$; $c \rightarrow \mathrm{m}(\boldsymbol{x})$
  Eval CenteredForm($f_1, \boldsymbol{x}, c$)
  Eval CenteredForm($f_2, \boldsymbol{x}, c$)
  if ($\underline{f_1}(\boldsymbol{x}) > maxf_1$ || $\underline{f_2}(\boldsymbol{x}) > maxf_2$)
    continue
  if Infeasible($\boldsymbol{x}$)
    continue
  if MultiCutOffTest($\boldsymbol{x}$))
    continue
  CutOffTest($c, \mathcal{L}_W^{min} \cup \mathcal{L}_S^{cur}, \mathcal{L}_W^{\delta}, \tilde{f}, \delta$);
  if ($\underline{f_1}(\boldsymbol{x}) > \tilde{f}$)
    if ($\underline{f_1}(\boldsymbol{x}) \leq \tilde{f} * (1+\delta)$)
      $\boldsymbol{x} \rightarrow \mathcal{L}_W^{\delta}$; **continue**
    else
      $\boldsymbol{x} \rightarrow \mathcal{L}_{Next}$; **continue**
  if MonoTest($\boldsymbol{x}$,$\boldsymbol{z}$)
    $\boldsymbol{x} \rightarrow \mathcal{L}_W^{\delta}$
    $\boldsymbol{z} \rightarrow \boldsymbol{x}$
  if ($\underline{f_1}(\boldsymbol{x}) + \varepsilon|\underline{f_1}(\boldsymbol{x})| \geq \tilde{f}$ || $\mathrm{w}(\boldsymbol{x}) < \varepsilon$)
    $\boldsymbol{x} \rightarrow \mathcal{L}_S^{cur}$
  else
    Prune($\boldsymbol{x}, \tilde{f}, \varepsilon, \delta$) $\rightarrow \boldsymbol{x}_1, \boldsymbol{x}_2$
  for $i = 1, 2$ **do**
    $f_2(\boldsymbol{x}_i) \cap = f_2(c) + (\boldsymbol{x}_i - c)^T \nabla f_2(\boldsymbol{x})$
    if Infeasible($\boldsymbol{x}_i$)
      continue
    Eval $f_1(\boldsymbol{x}_i)$
    $f_1(\boldsymbol{x}_i) \cap = f_1(c) + (\boldsymbol{x}_i - c)^T \nabla f_1(\boldsymbol{x})$
    if MultiCutOffTest($\boldsymbol{y}_i$)
      continue
    if ($\underline{f_1}(\boldsymbol{x}_i) > \tilde{f}$)
      if ($\underline{f_1}(\boldsymbol{x}_i) \leq \tilde{f} * (1+\delta)$)
        $\boldsymbol{x}_i \rightarrow \mathcal{L}_W^{\delta}$; **continue**
      else
        $\boldsymbol{x}_i \rightarrow \mathcal{L}_{Next}$; **continue**
    if ($\underline{f_1}(\boldsymbol{x}_i) > maxf_1$ || $\underline{f_2}(\boldsymbol{x}_i) > maxf_2$)
      continue
    $\boldsymbol{x}_i \rightarrow \mathcal{L}_W^{min}$
  endfor
endwhile

Phase 2

Input: $f_1, f_2, h_l, \mathcal{L}_S^{cur}, \mathcal{L}_W^{\delta}, \mathcal{L}_{Next}, \mathcal{L}_{PNS}, \tilde{f}, \delta, \varepsilon, \eta, \mu, maxf_1, maxf_2$
Output: $\mathcal{L}_W^{min}, \mathcal{L}_{\mathcal{S}}, \mathcal{L}_{PNS}$

while ($\mathcal{L}_W^{\delta} \neq \emptyset$) **do**
  $\mathcal{L}_W^{\delta} \rightarrow \boldsymbol{x}$; $c = \mathrm{m}(\boldsymbol{x})$
  Eval CenteredForm($f_1, \boldsymbol{x}, c$)
  Eval CenteredForm($f_2, \boldsymbol{x}, c$)
  if ($\underline{f_1}(\boldsymbol{x}) > maxf_1$ || $\underline{f_2}(\boldsymbol{x}) > maxf_2$)
    continue
  if Infeasible($\boldsymbol{x}$)
    continue
  if MultiCutOffTest($\boldsymbol{x}$)
    continue
  if ($\underline{f_1}(\boldsymbol{x}) > \tilde{f} * (1+\delta)$)
    $\boldsymbol{x} \rightarrow \mathcal{L}_{Next}$; **continue**
  if (Feasible(c) & $\overline{f_2}(c) < f_2^{(i)}$ & $\overline{f_1}(c) < \tilde{f} * (1+\delta)$)
    $f_2^{(i)} = \overline{f_2}(c)$
  if ($\overline{f}_1(\boldsymbol{x}) < \tilde{f} * (1+\delta)(1+\eta)$ & (Feasible($\boldsymbol{x}$) || $\mathrm{w}(f_2(\boldsymbol{x})) < \mu$ || $\mathrm{w}(\boldsymbol{x}) < \varepsilon$))
    $\boldsymbol{x}_i \rightarrow \mathcal{L}_S^{cur}$; **continue**
  Prune($\boldsymbol{x}, \tilde{f}, \varepsilon, \delta$) $\rightarrow \boldsymbol{x}_1, \boldsymbol{x}_2$
  for $i = 1, 2$ **do**
    $f_2(\boldsymbol{x}_i) \cap = f_2(c) + (\boldsymbol{x}_i - c)^T \nabla f_2(\boldsymbol{x})$
    if Infeasible($\boldsymbol{x}_i$)
      continue
    Eval $f_1(\boldsymbol{x}_i)$
    $f_1(\boldsymbol{x}_i) \cap = f_1(c) + (\boldsymbol{x}_i - c)^T \nabla f_1(\boldsymbol{x})$
    if MultiCutOffTest($\boldsymbol{x}_i$)
      continue
    if ($\underline{f_1}(\boldsymbol{x}_i) \leq \tilde{f} * (1+\delta)$)
      $\boldsymbol{x}_i \rightarrow \mathcal{L}_W^{\delta}$; **continue**
    else
      $\boldsymbol{x}_i \rightarrow \mathcal{L}_{Next}$; **continue**
    if ($\underline{f_1}(\boldsymbol{x}_i) > maxf_1$ || $\underline{f_2}(\boldsymbol{x}_i) > maxf_2$)
      continue
    $\boldsymbol{x}_i \rightarrow \mathcal{L}_W^{\delta}$
  endfor
endwhile
$\mathcal{L}_S^{cur} \rightarrow \mathcal{L}_{\mathcal{S}}$; $\mathcal{L}_{Next} \rightarrow \mathcal{L}_W^{min}$
for $z^* \in \mathcal{L}_{PNS}$ **do**
  if $z_2^* > f_2^{(i)}$
    Remove z^* from $\mathcal{L}_{PNS}$

Multi-objective cut-off test: Every time MultiCutOffTest($\boldsymbol{x}$) is applied to a box $\boldsymbol{x}$, and provided that its midpoint c (as a point interval) is feasible, we compute $\overline{f}(c) = (\overline{f}_1(c), \overline{f}_2(c))$ and update the list $\mathcal{L}_{PNS}$ of *provisional nondominated solutions* available so far (if possible, i.e., if $\overline{f}(c)$ is not dominated

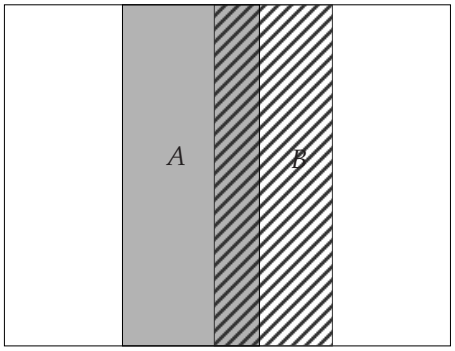


Figure 3.5: Pruning applied to f_1 and f_2.

by any point in $\mathcal{L}_{PNS}$). The test discards boxes whose points are not efficient, i.e., a box $\boldsymbol{x}$ is removed (and then the test is true) if $\underline{\boldsymbol{f}}(\boldsymbol{x}) > z^*$ for some $z^* \in \mathcal{L}_{PNS}$. When this test and the δ-cut-off test are used together in the same algorithm, then $\tilde{f}$ is updated in the δ-cut-off test only if $\overline{\boldsymbol{f}}(c)$ is not dominated by any point in $\mathcal{L}_{PNS}$. This is to avoid the possibility that the list $\mathcal{L}_W^{min}$ of a constraint problem becomes empty, which may happen in some cases in which all its boxes are deleted by the multiobjective cut-off test or the δ-cut-off test.

A few comments on Algorithm 3.3 are in order. The box $\boldsymbol{x}$ to be chosen from $\mathcal{L}_W^{min}$ (Step 2 of Phase 1) and $\mathcal{L}_W^{\delta}$ (Step 2 of Phase 2) is the one with the lowest $\underline{\boldsymbol{f}}_1(\boldsymbol{x})$ value. Notice that in some steps we compute the centered form as described in Subsection 1.3.1.2, whereas in Steps 25 and 29 of Phase 1 and in Steps 19 and 23 of Phase 2, we use the inclusion of the gradient of the parent box (to save time). In steps 37 to 39 of Phase 2, we remove from $\mathcal{L}_{PNS}$ points $z^* = (z_1^*, z_2^*)$ for which $z_2^* > f_2^{(i)}$ holds, since those points will not discard any box in the next constraint problems.

In Step 20 of Phase 1 (see also Step 15 of Phase 2), we send a box to $\mathcal{L}_S^{cur}$ if its width is less than a given tolerance. We have added this stopping rule because if the tolerances are chosen too small then, due to the overestimation of the inclusion functions, it may happen that the other stopping criteria are never fulfilled. On the other hand, in Step 15 of Phase 2, for sending a box $\boldsymbol{x}$ to $\mathcal{L}_S^{cur}$, in addition to the condition $\overline{\boldsymbol{f}}_1(\boldsymbol{x}) < \tilde{f}(1+\delta)(1+\eta)$ we also require the box $\boldsymbol{x}$ either to be feasible or to satisfy either $\mathrm{w}(\boldsymbol{f}_2(\boldsymbol{x})) < \mu$ (μ is a given tolerance) or $\mathrm{w}(\boldsymbol{x}) < \varepsilon$. This is to avoid sending $\mathcal{L}_S^{cur}$ big undetermined boxes.

We also want to point out that if we remove the pruning tests (and we perform a simple bisection perpendicular to the direction of maximum width instead), the δ-monotonicity test and the multiobjective cut-off test, we have an interval version of the GBSSS method [120].

The resulting solution list $\mathcal{L}_{\mathcal{S}}$ of Algorithm 3.2 is the desired list of boxes containing the complete efficient set of (3.1).

Computational experiments of the Constraint-like method and the efficiency of its discarding tests can be seen on some facility location problems in Section 8.2.3.

3.4 Biobjective Interval Branch and Bound method

In this section we introduce a new method, a biobjective interval Branch and Bound algorithm, which is also able to obtain a superset of S_{WE}. This superset of S_{WE} maps into a superset of the weakly nondominated set Z_{WN}, which may be made as tight as required (up to the precision provided by the inclusion functions used) by reducing the tolerances employed in the termination rules (see Section 3.4.4).

The biobjective interval B&B method deals with the multiple objectives directly (that is, it does not convert the problem into a single objective optimization problem or a family of such kind of problems,

as most of the multiobjective optimization methods do). To our knowledge, in all the publications on interval methods, there is only one paper dealing with multiobjective problems apart from the methods presented in this thesis. In [85] it is simply proposed to use the classical interval B&B global optimization methods for solving the single objective problems to which the multiobjective problem is reduced when using the weighting method or the minimax method.

Furthermore, we also give theoretical results which show the strength of the approach (see Sections 3.4.5 and 3.4.6). The limit properties (when the tolerances are set equal to zero and the algorithm does not stop) prove that under mild conditions, the algorithm converges to the set of weakly efficient points. In the more practical case, when the tolerances used are strictly positive (and then the algorithm stops after a finite number of iterations), it is proved that the points in the boxes of the solution list are very close to being efficient.

The main structure of the method is the same as that of the interval Branch and Bound method in Section 1.4, written in Algorithm 1.1. Of course all the rules have to be changed in order to fit the biobjective nature of the problem. In the next sections these rules are specified.

3.4.1 Selection rules

Several rules for the selection of the next box to be processed can be used. In particular, we have worked with the following ones:

SR1: Select the box $\boldsymbol{x} \in \mathcal{L}_{\mathcal{W}}$ with minimum lower bound $\underline{\boldsymbol{f}_1}(\boldsymbol{x})$. In the criterion space, this implies that Z_{WN} will be generated from left-top to right-bottom.

SR2: Select the box $\boldsymbol{x} \in \mathcal{L}_{\mathcal{W}}$ with minimum lower bound $\underline{\boldsymbol{f}_2}(\boldsymbol{x})$. In the criterion space, this implies that Z_{WN} will be generated from right-bottom to left-top.

SR3: We first rescale the objective functions so that their objective values are of approximately the same magnitude. Ideally, the normalization process is done with the help of the ideal (z^*) and nadir (z^{nad}) objective vectors.

Once the ideal and nadir objective vectors are known, we replace $f_i(x)$ by

$$\frac{f_i(x) - z_i^*}{z_i^{nad} - z_i^*},$$

whose range is within $[0, 1]$. However, since in practice, those vectors are not known in advance, we propose the following normalization:

$$n_i(x) = \frac{f_i(x) - \tilde{z}_i^*}{\tilde{z}_i^{nad} - \tilde{z}_i^*},$$

where

$$\tilde{z}_i^* = \min\{\underline{\boldsymbol{f}_i}(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{L}_{\mathcal{W}} \cup \mathcal{L}_{\mathcal{S}}\}$$

and

$$\tilde{z}_i^{nad} = \max\{\underline{\boldsymbol{f}_i}(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{L}_{\mathcal{W}} \cup \mathcal{L}_{\mathcal{S}}\}.$$

The new selection rule that we propose is to select the box $\boldsymbol{x} \in \mathcal{L}_{\mathcal{W}}$ with minimum lower bound $\lambda \underline{\boldsymbol{n}_1}(\boldsymbol{x}) + (1-\lambda)\underline{\boldsymbol{n}_2}(\boldsymbol{x})$, with $\lambda \in [0, 1]$ a given value. Depending on the actual value of λ we first explore a given area of Z_N. Notice that this rule generalizes the previous ones (we can obtain the first rule by setting $\lambda = 1$ and the second one when $\lambda = 0$, see Figure 3.6).

SR4: We may alternate between different λ-values in the previous rule from iteration to iteration. In this way, all the parts of the criterion space are generated in a more or less uniform way. In particular, to consider p different values, at iteration k we may set

$$\lambda = \frac{(k-1) \bmod p}{p-1}.$$

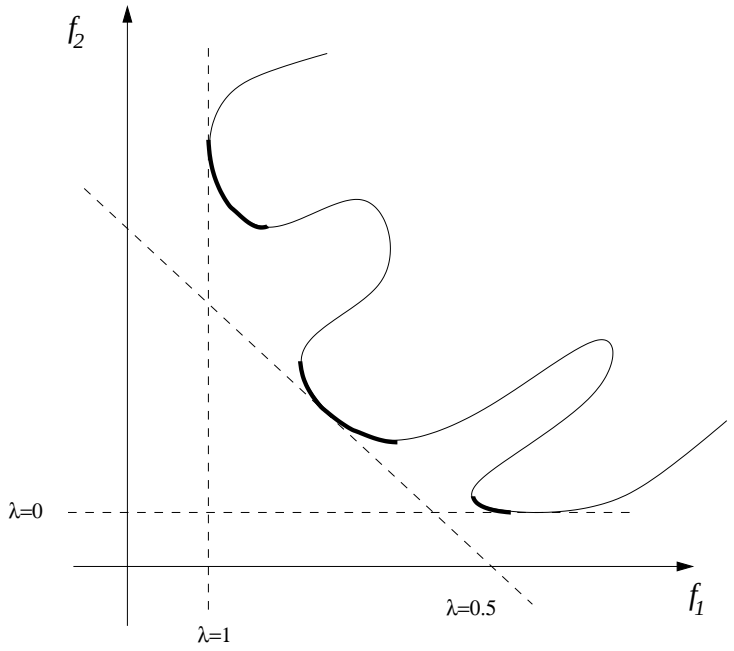


Figure 3.6: Selection rules.

Numerical comparison of the usefulness of the different selection rules can be found in Section 8.2.4.

3.4.2 Division rule

We have applied the most widely used division rule in interval B&B methods, which is the bisection of the box perpendicular to a direction of maximum width.

3.4.3 Discarding tests

The tests for verifying that a box contains no efficient point form the most important part of the biobjective interval Branch and Bound method. Here we have used the common feasibility test (see Section 1.4), the multiobjective cut-off test described in Section 3.3.2.3, two new multiobjective monotonicity tests and a new multiobjective pruning test. We next describe our new tests.

3.4.3.1 Multiobjective monotonicity test

Let $\boldsymbol{g}^l = (\boldsymbol{g}^l_1, \ldots, \boldsymbol{g}^l_n)$ be an inclusion function for the gradient f'_l of the objective function f_l, $l = 1, 2$, and $\boldsymbol{x} \subseteq \boldsymbol{s}$ a box, and suppose that the following conditions hold:

1. $\boldsymbol{g}^1_j(\boldsymbol{x}) \cdot \boldsymbol{g}^2_j(\boldsymbol{x}) > 0$ for some j (i.e., both objectives are either increasing or decreasing along the j-th variable).

2. The facet $\boldsymbol{x}'$ of box $\boldsymbol{x}$ containing the best points of $\boldsymbol{x}$, i.e., those points not weakly dominated by any other point in $\boldsymbol{x}$ (for instance, if $\boldsymbol{g}^l_j(\boldsymbol{x}) < 0, l = 1, 2$ then $\boldsymbol{x}' = (\boldsymbol{x}_1, \ldots, \overline{\boldsymbol{x}}_j, \ldots, \boldsymbol{x}_n)$), is feasible and is not on the border of the initial box $\boldsymbol{s}$.

Then, box $\boldsymbol{x}$ can be discarded from further consideration, since it does not contain any weakly efficient point (by moving along the j-th coordinate direction, we can find points which dominate all the points in $\boldsymbol{x}$). In particular, if the whole box $\boldsymbol{x}$ is feasible, the second condition holds, and it is only necessary to check the first condition. For an example, see Figure 3.7. Notice that we do not require the whole box

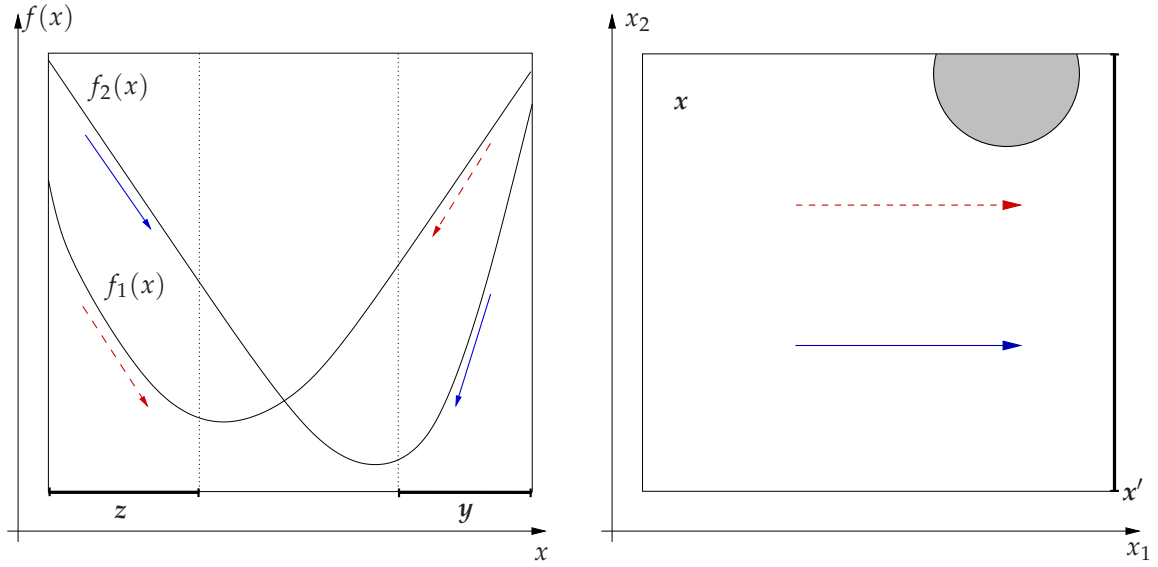


Figure 3.7: Multiobjective monotonicity test. In the left picture we can see the plot of two objective functions (one-dimensional); the feasible intervals $\boldsymbol{z}$ and $\boldsymbol{y}$ can be discarded. In the right picture the arrows show the decreasing directions of other two objective functions (two-dimensional); the gray area is not feasible; however, the best facet $\boldsymbol{x}'$ is feasible, thus, the box $\boldsymbol{x}$ can be discarded.

to be feasible, just the facet $\boldsymbol{x}'$ containing the best points of $\boldsymbol{x}$. Moreover, we only need to check that $\boldsymbol{x}'$ satisfies the constraints in which the j-th variable appears.

On the other hand, if the box $\boldsymbol{x}$ satisfies the first condition, and the facet $\boldsymbol{x}'$ is feasible but it is on the boundary of the initial box, then $\boldsymbol{x}$ can be narrowed to the intersection of the boundary and $\boldsymbol{x}'$.

3.4.3.2 Generalized multiobjective monotonicity test

Let $\boldsymbol{x}$ be a feasible or undetermined box, and let us suppose that the multiobjective monotonicity test has not discarded it, i.e., there is no coordinate direction along which both objective functions are either increasing or decreasing, or for those directions, the corresponding facet is not feasible or it is on the border of $\boldsymbol{s}$. But what happens if one of the objective functions is monotonous along a given coordinate direction, say i, and the other objective function is monotonous along a different coordinate direction, say j, $j \neq i$? In this case, we can still try to find out whether both objective functions are either increasing or decreasing along a given direction v different from the coordinate directions. In particular, the generalized multiobjective monotonicity test introduced here searches for directions which are linear combinations of the i-th and j-th coordinate directions, i.e., for directions v of the form $v = (v_1, \ldots, v_n)$, with $v_k = 0 \ \forall k \neq i, j$ (see Figure 3.8). Remember that the monotonicity of the objective function $f_l, l = 1, 2$, along a direction v is given by the directional derivative of f_l along the vector v, $\frac{\partial f_l(x)}{\partial v}$, and if we denote with $f_l'(x)$ the gradient of f_l at x, then

$$\frac{\partial f_l(x)}{\partial v} = \lim_{h \searrow 0} \frac{f_l(x + hv) - f_l(x)}{h} = f_l'(x) \cdot v, \qquad l = 1, 2.$$

In particular, if v is as described above, then

$$\frac{\partial f_l(x)}{\partial v} = v_i \boldsymbol{g}_i^l(x) + v_j \boldsymbol{g}_j^l(x), \qquad l = 1, 2.$$

Notice that it is enough to search for a direction v such that

$$\frac{\partial f_l(x)}{\partial v} > 0, \quad l = 1, 2 \tag{3.8}$$

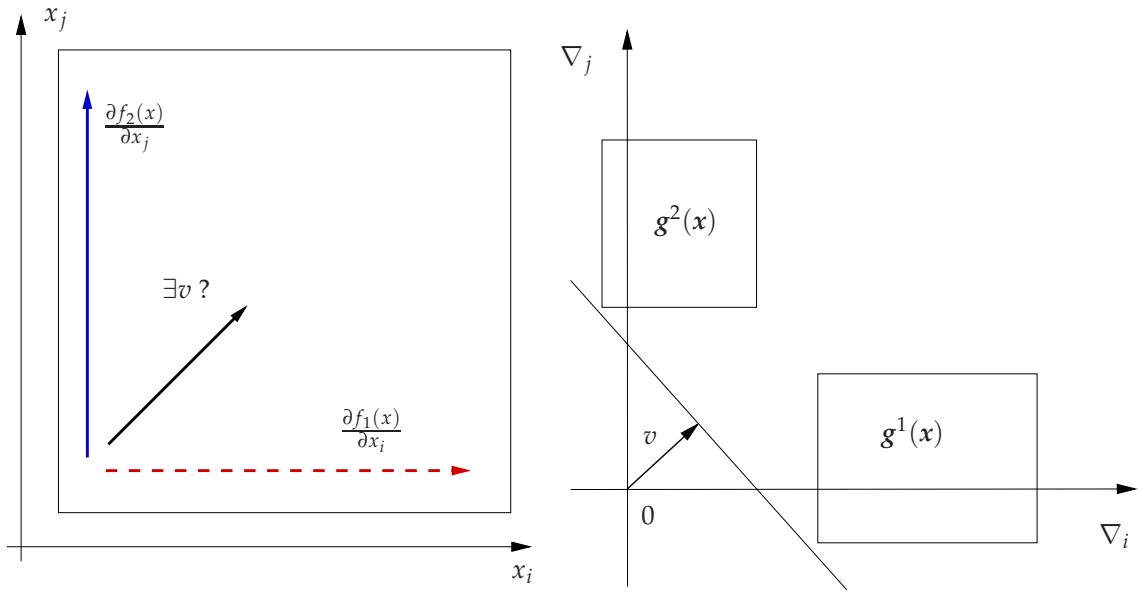


Figure 3.8: Generalized multiobjective monotonicity test. f_1 is monotonous along coordinate direction i, whereas f_2 is monotonous along coordinate direction j.

since if both objective functions are decreasing along a direction v, then they are increasing along the opposite direction $-v$.

Let us suppose, without loss of generality, that f_1 is monotonous in direction i, while f_2 is monotonous along direction j. Four cases can be distinguished depending on the directions of the monotonicity. Each case corresponds to one quadrant of the coordinate system where vector v has to be found, although they lead to similar conditions. Let us introduce the notation, $\boldsymbol{G}$, to transform every case to the first quadrant of the coordinate system (see Figure 3.8):

$$\boldsymbol{G}_{il} = \boldsymbol{g}_i^l(\boldsymbol{x}) \operatorname{sign}(\boldsymbol{g}_i^1(\boldsymbol{x})), \quad \boldsymbol{G}_{jl} = \boldsymbol{g}_j^l(\boldsymbol{x}) \operatorname{sign}(\boldsymbol{g}_j^2(\boldsymbol{x})), \quad l = 1, 2, \tag{3.9}$$

i.e. if $\boldsymbol{g}_i^1(\boldsymbol{x})$ (the monotonous direction for f_1) is negative, both $\boldsymbol{g}_i^1(\boldsymbol{x})$ and $\boldsymbol{g}_i^2(\boldsymbol{x})$ must be multiplied by -1, and similarly, if $\boldsymbol{g}_j^2(\boldsymbol{x})$ is negative, both $\boldsymbol{g}_j^1(\boldsymbol{x})$ and $\boldsymbol{g}_j^2(\boldsymbol{x})$ must be multiplied by -1.

Theorem 3.22. *Let $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ be two real functions, $\boldsymbol{x} \in \mathbb{I}^n$ a box within the domains of both functions, and $\boldsymbol{g}_k^l(\boldsymbol{x})$ an inclusion for the partial derivative $\frac{\partial f_l}{\partial x_k}$, $k = i, j$ and $l = 1, 2$, at $\boldsymbol{x}$, respectively. Suppose that there is no coordinate direction along which both functions are either increasing or decreasing. If f_1, f_2 are monotonous along the coordinate directions i, j, respectively, and the condition*

$$\frac{\underline{\boldsymbol{G}_{i2}}}{\underline{\boldsymbol{G}_{j2}}} - \frac{\underline{\boldsymbol{G}_{i1}}}{\underline{\boldsymbol{G}_{j1}}} > 0,$$

holds, then there exists a direction v along which both objective functions are increasing.

Proof. We are searching for a direction $v > 0$, where both objectives are increasing. We can suppose, without loss of generality, that $v_i = 1$, and $v_k = 0, k \in \{1, \ldots, n\} \backslash \{i, j\}$. Therefore we are searching for $v_j > 0$ such that

$$\boldsymbol{G}_{il} + v_j \cdot \boldsymbol{G}_{jl} > 0, \quad l = 1, 2.$$

Since $v_j > 0$, the previous condition holds iff

$$\underline{\boldsymbol{G}_{il}} + v_j \cdot \underline{\boldsymbol{G}_{jl}} > 0, \quad l = 1, 2,$$

that is, iff

$$v_j \cdot \underline{\boldsymbol{G}_{jl}} > -\underline{\boldsymbol{G}_{il}}, \quad l = 1, 2.$$

We know that $\underline{\boldsymbol{G}_{j1}} < 0$ (i.e. $0 \in \boldsymbol{g}_j^1$ or $\boldsymbol{g}_j^1(\boldsymbol{x}) \cdot \boldsymbol{g}_j^2(\boldsymbol{x}) < 0$), because otherwise the multiobjective monotonicity test would already discard the box, and $\underline{\boldsymbol{G}_{j2}} > 0$, because f_2 is monotonous along direction j and $\boldsymbol{G}_{j2} = \boldsymbol{g}_j^2(\boldsymbol{x})\ \mathrm{sign}(\boldsymbol{g}_j^2(\boldsymbol{x}))$. Thus

$$\frac{\overline{\boldsymbol{G}_{i2}}}{\underline{\boldsymbol{G}_{j2}}} > -v_j > \frac{\overline{\boldsymbol{G}_{i1}}}{\underline{\boldsymbol{G}_{j1}}}.$$

Both fractions are negative, thus $v_j > 0$ is ensured. It means that the direction v exists as far as

$$\frac{\overline{\boldsymbol{G}_{i2}}}{\underline{\boldsymbol{G}_{j2}}} - \frac{\overline{\boldsymbol{G}_{i1}}}{\underline{\boldsymbol{G}_{j1}}} > 0.$$

□

Corollary 3.23. *Let $\boldsymbol{x}$ be a feasible or undetermined box of problem (3.1) which has not been discarded by the multiobjective monotonicity test, but for which the conditions of the previous theorem hold. If the facets $\boldsymbol{x}'$ and $\boldsymbol{x}''$ corresponding to the degenerated intervals in direction i and j which contain the minimum of f_1 and f_2 over the box, respectively, are feasible, then the box does not contain any weakly efficient point in its interior. In particular, if those facets are feasible and are not on the border of the initial box $\mathbf{s}$, then the box can be discarded, and if those facets are feasible but are on the border of $\mathbf{s}$ then the box can be narrowed to $(\boldsymbol{x}' \cup \boldsymbol{x}'') \cap \partial\boldsymbol{s}$, where $\partial\mathbf{s}$ denotes the boundary of S.*

In both the multiobjective monotonicity test and the generalized multiobjective monotonicity test the feasibility of the corresponding facets has to be checked. In the first test we store information about the facets which are not feasible, thus when we apply the generalized test those directions are not taken into account. This allows us to save time when carrying out the second test and to assure the condition of Theorem 3.22 that there is no coordinate direction along which both functions are either increasing or decreasing.

3.4.3.3 Multiobjective pruning test

Next, we extend the pruning test described in Section 2.2.4 so that it can handle biobjective problems. Not to complicate matters, we briefly explain the multiobjective pruning test using a one-dimensional biobjective minimization problem (see Figure 3.9). Consider a (feasible or undetermined) box $\boldsymbol{x}$, and suppose that we know a lower bound for the value of the objective functions at $\mathrm{m}(\boldsymbol{x})$ (for instance, $\underline{f_l}(\mathrm{m}(\boldsymbol{x})), l = 1, 2$), and also an inclusion for the gradient of the objective functions at $\boldsymbol{x}$, $\boldsymbol{g}^l(\boldsymbol{x}), l = 1, 2$. With this information, linear lower bounding functions of f_1 and f_2 can be constructed as it is shown in Figure 3.9 (similar to what is commonly done in Lipschitz optimization [74]). Then, if $(\tilde{f}_1, \tilde{f}_2) = (f_1(c), f_2(c))$ is the objective value at a point $c \in S$, and $\tilde{f}_l < \underline{f_l}(\mathrm{m}(\boldsymbol{x})), l = 1, 2$, then the points in $\boldsymbol{x}_a \cap \boldsymbol{x}_b$ (see Figure 3.9) cannot be weakly efficient (they are dominated by c), and that part of the box can be discarded.

This pruning test can be performed for any coordinate direction of the box, generating one or two new subboxes. In this sense, it can be seen as a bisection method along the chosen coordinate. However, so as to avoid unnecessary divisions, a coordinate j is selected only if $\mathrm{w}(\boldsymbol{x}_j) > \varepsilon$, for a given tolerance ε.

Two important steps in this test are the selection of the point $(\tilde{f}_1, \tilde{f}_2)$ used to do the pruning and the selection of the direction j along which to do it. In order to be able to discard the largest part of $\boldsymbol{x}$ we have worked as follows. The most suitable points to use are those in the list $\mathcal{L}_{PNS}$ of provisional nondominated solutions. Thus, for each point in $\mathcal{L}_{PNS}$ we have computed the size of the box that can be discarded with it along each coordinate direction, and among all the points and all the directions we have chosen the pair such that the corresponding part removed from $\boldsymbol{x}$ is the largest one.

The practical efficiency of the above discarding tests are examined in Section 8.2.5.

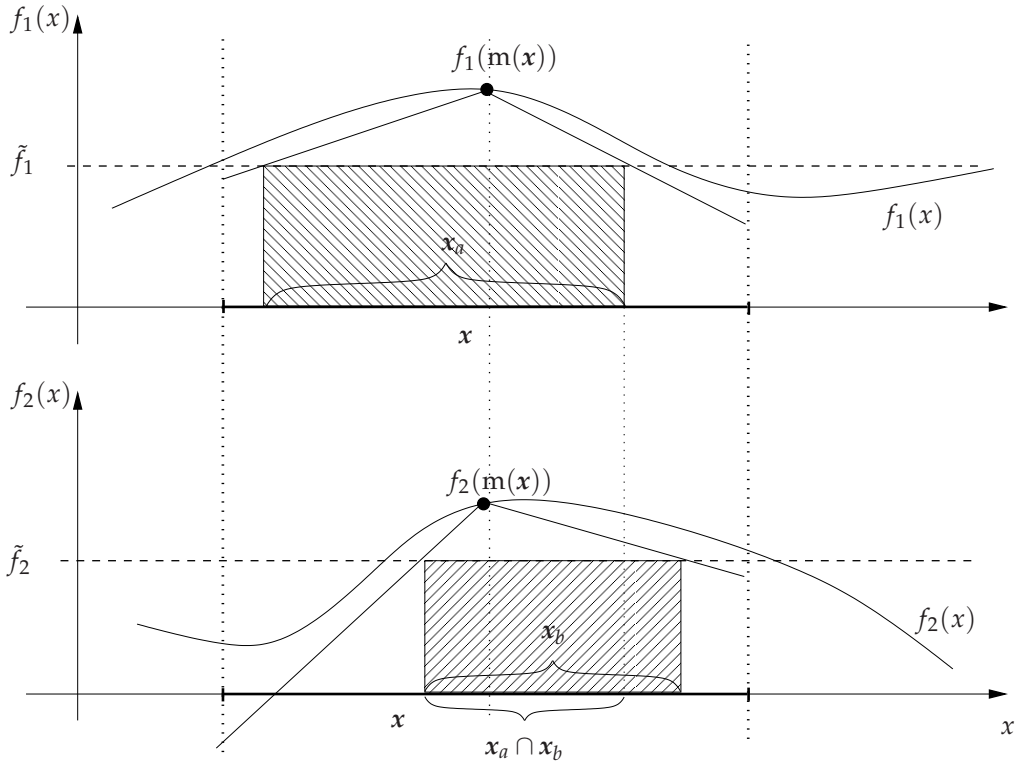


Figure 3.9: Multiobjective pruning test in the one-dimensional case. The points in $\boldsymbol{x}_a \cap \boldsymbol{x}_b$ are dominated by the point c for which $(\tilde{f}_1, \tilde{f}_2) = (f_1(c), f_2(c))$.

3.4.4 Termination rule

As termination rule, we have used the following.

TR1: Send a box $\boldsymbol{x}$ to $\mathcal{L}_{\mathcal{S}}$ whenever

$$\mathrm{w}_{rel}(\boldsymbol{f}_1(\boldsymbol{x})) < \varepsilon_1 \text{ and } \mathrm{w}_{rel}(\boldsymbol{f}_2(\boldsymbol{x})) < \varepsilon_2$$

or

$$\mathrm{w}(\boldsymbol{x}) < \varepsilon_3.$$

3.4.5 Convergence properties

To study the convergence properties of the algorithm, we will suppose that $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 0$, so that the stopping criterion TR1 never holds and the algorithm generates infinitely many boxes. Furthermore, we have assumed that SR1 is the selection rule used, although similar proofs can be derived for the other selection rules as well.

Definition 3.24. *Let $(\boldsymbol{z}^{(k)})_{k=1}^{\infty}$ be a sequence of boxes, $\boldsymbol{z}^{(k)} \in \mathbb{I}^n \ \forall k$. A point $z \in \mathbb{R}^n$ is said to be* an accumulation point of the sequence $(\boldsymbol{z}^{(k)})_{k=1}^{\infty}$ *if there exist points $z^{(k)} \in \boldsymbol{z}^{(k)}$, $k = 1, 2, \ldots$ such that z is an accumulation point of the sequence $(z^{(k)})_{k=1}^{\infty}$.*

Definition 3.25. *We say that the sequence of boxes $(\boldsymbol{z}^{(k)})_{k=1}^{\infty}$ is an* infinite subdivision sequence of the box $\boldsymbol{z}$, *if $\boldsymbol{z}^{(1)} = \boldsymbol{z}$, and for all $k \geq 1$, the box $\boldsymbol{z}^{(k+1)}$ has been generated by bisection from box $\boldsymbol{z}^{(k)}$.*

Definition 3.26. *A subdivision direction selection rule is said to be* completely balanced *if, for any box $\boldsymbol{x}$, and for each infinite subdivision sequence of $\boldsymbol{x}$ generated by the rule, the sequence of the directions generated by the rule contains each i of the possible directions for which $\mathrm{w}(\boldsymbol{x}_i) > 0$ an infinite number of times.*

As we can see from the definition, if the direction selection rule is completely balanced then for each direction the number of iteration steps between two appearances is finite.

Theorem 3.27. *The biobjective interval Branch and Bound algorithm converges in the sense that for any infinite subdivision sequence $(\boldsymbol{z}^{(k)})_{k=1}^{\infty}$ of $\boldsymbol{s}$ generated by the algorithm $\lim_{k\to\infty} \mathrm{w}(\boldsymbol{z}^{(k)}) = 0$, if and only if the subdivision direction selection rule is completely balanced.*

Proof.

("$\Rightarrow$") Assume that there exists an infinite subdivision sequence of $\boldsymbol{s}$, $(\boldsymbol{z}^{(k)})_{k=1}^{\infty}$, with $\lim_{k\to\infty} \mathrm{w}(\boldsymbol{z}^{(k)}) = 0$, such that an index $i \in \{1,\dots,n\}$ with $\mathrm{w}(\boldsymbol{s}_i) > 0$ appears only a finite number of times. Then, for such an i, $\lim_{k\to\infty} \mathrm{w}(\boldsymbol{z}_i^{(k)}) > 0$, and thus $\lim_{k\to\infty} \mathrm{w}(\boldsymbol{z}^{(k)}) > 0$. This is a contradiction.

("$\Leftarrow$") Assume now that the direction selection rule is completely balanced, and for a given problem the algorithm produces an infinite subdivision sequence $(\boldsymbol{z}^{(k)})_{k=1}^{\infty}$ of $\boldsymbol{s}$. The balancedness property implies that $\lim_{k\to\infty} \mathrm{w}(\boldsymbol{z}^{(k)}) = 0$. □

Let us denote by A the set of accumulation points of the infinite subdivision sequences $(\boldsymbol{z}^{(k)})_{k=1}^{\infty}$ of $\boldsymbol{s}$ generated by the algorithm, by $\mathcal{L}_{\mathcal{W}}{}^{(k)}$ the working list $\mathcal{L}_{\mathcal{W}}$ in the k-th iteration and by $U^{(k)}$ the union of all the boxes in $\mathcal{L}_{\mathcal{W}}{}^{(k)}$.

Theorem 3.28. *Assume that for the inclusion functions $\boldsymbol{f} = (\boldsymbol{f}_1, \boldsymbol{f}_2)$ and $\boldsymbol{h}_j$, $j = 1,\dots,q$, the contraction property holds, and the subdivision direction selection rule is completely balanced. Then the sequence $(U^{(k)})_{k=1}^{\infty}$ is nested and $S_{WE} \subseteq U^{(k)}\ \forall k$, where S_{WE} is the weakly efficient set. Furthermore, $A \subseteq S_{WE}$.*

Proof. It is obvious that $(U^{(k)})_{k=1}^{\infty}$ is nested. On the other hand, none of the discarding tests removes weakly efficient points, thus at any iteration k, any weakly efficient point must lie in one of the boxes of the working list, $\mathcal{L}_{\mathcal{W}}{}^{(k)}$. So $S_{WE} \subseteq U^{(k)} = \cup_{\boldsymbol{x}\in\mathcal{L}_{\mathcal{W}}{}^{(k)}} \boldsymbol{x},\ \forall k$.

Let us prove now that any point of A is feasible. Suppose that there exists an infeasible point $x \in A$, and let $(\boldsymbol{x}^{(k)})_{k=1}^{\infty}$ be an infinite subdivision sequence containing a sequence of points $(x^{(k)})_{k=1}^{\infty}$, with $x^{(k)} \in \boldsymbol{x}^{(k)}\ \forall\ k$, and $\lim_{k\to\infty} x^{(k)} = x$. Since the subdivision direction selection rule is completely balanced, $\lim_{k\to\infty} \mathrm{w}(\boldsymbol{x}^{(k)}) = 0$, thus $(\boldsymbol{x}^{k})_{k=1}^{\infty}$ must converge to x. Since x is infeasible, $h_j(x) > 0$ for some $j \in \{1,\dots,q\}$, and the contraction property of $\boldsymbol{h}_j$ implies that $\lim_{k\to\infty} \boldsymbol{h}_j(\boldsymbol{x}^{(k)}) = h_j(x)$. Thus, there must exist a value k_0 of k large enough so that $\underline{\boldsymbol{h}_j}(\boldsymbol{x}^{(k_0)}) > 0$, and the algorithm detects that the box $\boldsymbol{x}^{(k_0)}$ is infeasible, and discards it. Hence, $\boldsymbol{x}^{(k_0)}$ cannot be one of the elements of the sequence $(\boldsymbol{x}^{(k)})_{k=1}^{\infty}$. This is a contradiction.

To prove that $A \subseteq S_{WE}$, let us consider a point $z \in A$, with $z = \lim_{k\to\infty} z^{(k)}$, with $z^{(k)} \in \boldsymbol{z}^{(k)}$ and $(\boldsymbol{z}^{(k)})_{k=1}^{\infty}$ an infinite subdivision sequence of $\boldsymbol{s}$. We have seen above that z is feasible. Now, we shall prove that there is no point x for which $f_l(x) < f_l(z), l = 1,2$ holds. In particular, we shall prove that $f_1(x) < f_1(z)$ cannot hold. To see this, suppose that $f_1(x) < f_1(z)$ holds. Since the subdivision direction selection rule is completely balanced, $\lim_{k\to\infty} \mathrm{w}(\boldsymbol{z}^{(k)}) = 0$; on the other hand, $z^{(k)} \in \boldsymbol{z}^{(k)}$, thus, $\lim_{k\to\infty} \boldsymbol{z}^{(k)} = z$. Then, due to the contraction property of $\boldsymbol{f}_1$, $\lim_{k\to\infty} \mathrm{w}(\boldsymbol{f}_1(\boldsymbol{z}^{(k)})) = 0$, so $\lim_{k\to\infty} \boldsymbol{f}_1(\boldsymbol{z}^{(k)}) = f_1(z)$. Then there must exist an index k_0 such that $\underline{\boldsymbol{f}_1}(\boldsymbol{z}^{(k_0)}) > f_1(x)$ holds. If we denote by $\boldsymbol{x}^{(k)}$ the box in $\mathcal{L}_{\mathcal{W}}{}^{(k)}$ containing x, then for all $k \geq k_0$ the condition $\underline{\boldsymbol{f}_1}(\boldsymbol{x}^{(k_0)}) \leq f_1(x) < \underline{\boldsymbol{f}_1}(\boldsymbol{z}^{(k_0)})$ holds. This means that for all $k, k \geq k_0$, the selection rule SR1 should not have chosen $\boldsymbol{z}^{(k)}$, but this contradicts the fact that $(\boldsymbol{z}^{(k)})_{k=1}^{\infty}$ is an infinite subdivision sequence. Thus, $f_1(x) \geq f_1(z)$. This means that z is either efficient or, in the worst case, weakly efficient (in case $f_1(x) = f_1(z)$ and $f_2(x) < f_2(z)$). □

To be able to guarantee that the algorithm finds all the weakly efficient points, we need to slightly modify SR1, as follows. Let $N_0 \in \mathbb{Z}^+$ be a fixed positive integer:

SR1': Every N_0 iteration selects the box $\boldsymbol{x} \in \mathcal{L}_{\mathcal{W}}$ with maximum width, and in the rest of iterations select the box $\boldsymbol{x} \in \mathcal{L}_{\mathcal{W}}$ with the minimum lower bound $\underline{\boldsymbol{f}_1}(\boldsymbol{x})$.

Theorem 3.29. *Using selection rule SR1', and under the assumptions of Theorem 3.28, $A = S_{WE}$.*

Proof. To prove that $A \subseteq S_{WE}$, let us consider a point $z \in A$, with $z = \lim_{k\to\infty} z^{(k)}$, $z^{(k)} \in \boldsymbol{z}^{(k)}$ and $(\boldsymbol{z}^{(k)})_{k=1}^{\infty}$ an infinite subdivision sequence of $\boldsymbol{s}$. Point z is feasible (the proof of the previous theorem can be applied here). Now, we shall prove that there is no point x for which $f_l(x) < f_l(z), l = 1, 2$ holds. Suppose that such a point exists. Let us denote by $\boldsymbol{x}^{(k)}$ the interval containing the point x in $\mathcal{L}_{\mathcal{W}}{}^{(k)}$. The use of selection rule SR1' guarantees that an infinite subdivision sequence $(\boldsymbol{x}^{(k)})_{k=1}^{\infty}$ of $\boldsymbol{s}$ exists such that $x = \lim_{k\to\infty} x^{(k)}$ with $x^{(k)} \in \boldsymbol{x}^{(k)}$. Since the subdivision direction selection rule is completely balanced, $\lim_{k\to\infty} \mathrm{w}(\boldsymbol{x}^{(k)}) = 0$; on the other hand, $x^{(k)} \in \boldsymbol{x}^{(k)}$, thus, $\lim_{k\to\infty} \boldsymbol{x}^{(k)} = x$. Then, due to the contraction property of $\boldsymbol{f}_l$, $\lim_{k\to\infty} \mathrm{w}(\boldsymbol{f}_l(\boldsymbol{x}^{(k)})) = 0$, so $\lim_{k\to\infty} \boldsymbol{f}_l(\boldsymbol{x}^{(k)}) = f_l(x), l = 1, 2$. The same reasoning holds for the sequence $(\boldsymbol{z}^{(k)})_{k=1}^{\infty}$. Then, there must exist an index k_0 such that $\overline{\boldsymbol{f}_l}(\boldsymbol{x}^{(k_0)}) < \underline{\boldsymbol{f}_l}(\boldsymbol{z}^{(k_0)}), l = 1, 2$ holds. On the other hand, due to inclusion isotonicity, $\overline{\boldsymbol{f}_l}(\mathrm{m}(\boldsymbol{x}^{(k_0)})) \leq \overline{\boldsymbol{f}_l}(\boldsymbol{x}^{(k_0)}), l = 1, 2$, therefore at iteration k_0 the box $\boldsymbol{z}^{(k_0)}$ is eliminated by the multiobjective cut-off test. But this contradicts the fact that z is an accumulation point. Thus, $f_l(x) < f_l(z), l = 1, 2$ cannot hold. This means that z is either efficient or, in the worst case, weakly efficient.

Let us prove now that $S_{WE} \subseteq A$. Consider a point $z \in S_{WE}$. None of the discarding tests removes weakly efficient points. Thus, at any iteration k, z is contained in a box $\boldsymbol{z}^{(k)} \in \mathcal{L}_{\mathcal{W}}{}^{(k)}$. As the algorithm goes on, from time to time the box containing z will be selected by SR1' as the next box to be processed by the algorithm, since if it is not selected as the box, $\boldsymbol{x}$, having the minimum lower bound $\underline{\boldsymbol{f}_1}(\boldsymbol{x})$ it will become, sooner or later, the box of maximum width. Thus, there will be an infinite sequence of indices $(k_l)_{l=1}^{\infty}$ such that $z \in \boldsymbol{z}^{(k_l)}\ \forall l$, and $(\boldsymbol{z}^{(k_l)})_{l=1}^{\infty}$ will be an infinite subdivision sequence of $\boldsymbol{s}$. Now, since the subdivision direction selection rule is completely balanced, from Theorem 3.27 we have that $\lim_{l\to\infty} \mathrm{w}(\boldsymbol{z}^{(k_l)}) = 0$, so $\lim_{l\to\infty} \boldsymbol{z}^{(k_l)} = z$. Thus, $z \in A$. □

As we can see from the previous theorems, a key point for the algorithm to converge is to have a completely balanced subdivision direction selection rule. The next property proves that the classical division rule is in fact completely balanced.

Property 3.30. *The division rule which bisects the box perpendicularly to the direction of maximum width is completely balanced.*

Proof. Let $(\boldsymbol{z}^{(k)})_{k=1}^{\infty}$ be an infinite subdivision sequence of $\boldsymbol{s}$ generated by the algorithm when the division rule used is the bisection perpendicular to the direction of maximum width. If j is a coordinate direction that it is selected an infinite number of times, then $\lim_{k\to\infty} \mathrm{w}(\boldsymbol{z}_j^{(k)}) = 0$. Since the coordinate direction i selected for the subdivision is that for which $\mathrm{w}(\boldsymbol{z}_i)$ is the largest one, it follows that for all the coordinate directions $i \in \{1, \ldots, n\}$ for which $\mathrm{w}(\boldsymbol{s}_i) > 0$ holds it must happen that $\lim_{k\to\infty} \mathrm{w}(\boldsymbol{z}_i) = 0$. Thus, coordinate direction i must be selected an infinite number of times. □

Remark 3.31. Notice that in Theorem 3.29 concerning the selection rule, we have only used the fact that at every N_0 iteration the box with maximal width is chosen. Thus, the theorem also holds for any other selection rule with this modification.

3.4.6 Superdominance and underdominance

3.4.6.1 On the quality of the solutions

In the previous section we dealt with limit properties, i.e., when the stopping criterion never holds and the algorithm does not stop. But in practice, when run on a computer, the tolerances of the termination

rules are strictly positive, and the algorithm stops after a finite number of steps. In this section we investigate some properties of the algorithm in the practical case that the algorithm stops after a finite number of iterations.

In this case, the use of termination rule TR1 does not guarantee that all the boxes in $\mathcal{L}_{\mathcal{S}}$ contain feasible points, since an infeasible box $\boldsymbol{x}$ may be considered undetermined by the algorithm, and if $\mathrm{w}(\boldsymbol{x}) < \varepsilon_3$ then the box will be sent to $\mathcal{L}_{\mathcal{S}}$. That is why we propose using the following stopping criterion:

TR2: we send a box $\boldsymbol{x}$ to $\mathcal{L}_{\mathcal{S}}$ provided that:

1. $\mathrm{w}(\boldsymbol{f}_1(\boldsymbol{x})) < \varepsilon_1$ and $\mathrm{w}(\boldsymbol{f}_2(\boldsymbol{x})) < \varepsilon_2$, and
2. $\boldsymbol{x}$ contains at least one feasible point.

where $\varepsilon_1, \varepsilon_2 > 0$ are given tolerances.

If the midpoint of the box is feasible, then the second condition holds, and in fact, the objective value at that midpoint would have been used to update $\mathcal{L}_{PNS}$ in the multiobjective cut-off test. Otherwise, we should try to find a feasible point of $\boldsymbol{x}$ (usually, one of the corners of the box will be useful, which has always been the case in our computational studies) and update $\mathcal{L}_{PNS}$ accordingly, or at least to prove that $\boldsymbol{x}$ does contain feasible points and update $\mathcal{L}_{PNS}$ with $(\overline{\boldsymbol{f}}_1(\boldsymbol{x}), \overline{\boldsymbol{f}}_2(\boldsymbol{x}))$ (using, for instance, the techniques described in [89, 90, 154]).

Provided that $\varepsilon_1, \varepsilon_2$ are not chosen too small (smaller than the excess width of the inclusions provided by $\boldsymbol{f}_l, l = 1, 2$, at small boxes) the method behaves the same as with selection rule SR1', in the sense that after a given number of iterations the box of maximum width is selected as the next box to be processed by the algorithm. Thus, when the algorithm finishes, $\mathcal{L}_{\mathcal{S}}$ tightly encloses S_{WE}. And although some of the points in the boxes of $\mathcal{L}_{\mathcal{S}}$ will not be weakly efficient, they will be close to being, as the next result shows.

Definition 3.32. *Let $x, y \in S$ be two feasible points, and $\beta_1, \beta_2 \geq 0$ two non-negative numbers. We say that x (β_1, β_2)-superdominates y, i.e. $x \gg y$, if*

$$f_1(x) + \beta_1 < f_1(y) \quad \textit{and} \quad f_2(x) + \beta_2 < f_2(y).$$

$(0, 0)$-superdominance corresponds to the usual strong dominance. As we can see, if $x \gg y$ then x (strongly) dominates y, i.e., superdominance implies (strong) dominance.

Theorem 3.33. *If the biobjective interval Branch and Bound method is run with termination rule TR2 with positive tolerances $\varepsilon_1, \varepsilon_2 > 0$, then the points in the boxes of $\mathcal{L}_{\mathcal{S}}$ are not $(2\varepsilon_1, 2\varepsilon_2)$-superdominated.*

Proof. Let us suppose that there exists a point y in a box $\boldsymbol{y} \in \mathcal{L}_{\mathcal{S}}$ such that it is $(2\varepsilon_1, 2\varepsilon_2)$-superdominated by a point $\hat{x} \in S$. Then,

$$f_1(\hat{x}) + 2\varepsilon_1 < f_1(y) \quad \text{and} \quad f_2(\hat{x}) + 2\varepsilon_2 < f_2(y).$$

On the other hand, we can assume that $\hat{x}$ is included in a box $\hat{\boldsymbol{x}} \in \mathcal{L}_{\mathcal{S}}$. From the stopping criterion we have

$$\overline{\boldsymbol{f}}_l(\hat{\boldsymbol{x}}) \leq f_l(\hat{x}) + \varepsilon_l, l = 1, 2$$

and

$$f_l(y) \leq \underline{\boldsymbol{f}}_l(\boldsymbol{y}) + \varepsilon_l, l = 1, 2.$$

Thus,

$$\overline{\boldsymbol{f}}_l(\hat{\boldsymbol{x}}) + \varepsilon_l \leq f_l(\hat{x}) + 2\varepsilon_l < f_l(y) \leq \underline{\boldsymbol{f}}_l(\boldsymbol{y}) + \varepsilon_l, l = 1, 2,$$

that is,

$$\overline{\boldsymbol{f}}_l(\hat{\boldsymbol{x}}) < \underline{\boldsymbol{f}}_l(\boldsymbol{y}), l = 1, 2.$$

This means that any point in $\hat{\boldsymbol{x}}$ dominates $\boldsymbol{y}$. Thus, $\boldsymbol{y}$ would have been removed by the multiobjective cut-off test, which is a contradiction. □

From the previous theorem we can see that although the boxes in $\mathcal{L}_{\mathcal{S}}$ may contain weakly dominated points, they do not contain $(2\varepsilon_1, 2\varepsilon_2)$-superdominated points.

3.4.6.2 A finite representation of the weakly efficient set

It is also possible to prove that if we take one feasible point from each box in $\mathcal{L}_{\mathcal{S}}$, then that set of points forms a finite underdominator of (3.1). We need some definitions.

Definition 3.34. *Let $x, y \in S$ be two feasible points, and $\beta_1, \beta_2 > 0$ two positive numbers. We say that x (β_1, β_2)-*underdominates *y, i.e. $x \succeq y$, if*

$$f_1(x) - \beta_1 < f_1(y) \quad \text{and} \quad f_2(x) - \beta_2 < f_2(y).$$

As we can see, (weak) dominance implies (β_1, β_2)-underdominance.

Definition 3.35. *We say that a point x (β_1, β_2)-underdominates a set F if $x \succeq y$ for each point $y \in F$. We say that a set B is a (β_1, β_2)-underdominator for problem (3.1) if for each $y \in S$ there exists an $x \in B$ such that $x \succeq y$.*

Notice that if B is an underdominator for problem (3.1), then it can be considered as a (finite) representation of its complete weakly efficient set (see [9]).

Definition 3.36. *A function $h : \mathbb{R}^n \to \mathbb{R}$ is said to be Lipschitzian, with a Lipschitz constant L, if*

$$|h(x) - h(y)| \leq L\|x - y\|, \ \forall\, x, y.$$

Next, we show that if a box is small enough, then it is (β_1, β_2)-underdominated by any feasible point in it. The diameter of a set F will be denoted by $\delta(F)$.

Theorem 3.37. *Suppose that $f_l, l = 1, 2$ are Lipschitzian, with a Lipschitz constant $L_l, l = 1, 2$, and that $\boldsymbol{f}_l$ is an α_l-convergent inclusion function for $f_l, l = 1, 2$. Let $\boldsymbol{x}$ be any box and $\hat{x} \in \boldsymbol{x} \cap S$ a feasible point in $\boldsymbol{x}$. If $\delta(\boldsymbol{x} \cap S)$ is small enough then $\hat{x}$ (β_1, β_2)-underdominates $\boldsymbol{x} \cap S$.*

Proof. For each $l = 1, 2$, there is a constant c_l such that

$$\mathrm{w}(\boldsymbol{f}_l(\boldsymbol{x})) \leq c_l(\mathrm{w}(\boldsymbol{x}))^{\alpha_l} + \mathrm{w}(f_l(\boldsymbol{x})) \leq c_l(\mathrm{w}(\boldsymbol{x}))^{\alpha_l} + L_l\mathrm{w}(\boldsymbol{x})$$

In particular, for each $x \in \boldsymbol{x}$ we have that

$$f_l(\hat{x}) - f_l(x) \leq c_l(\mathrm{w}(\boldsymbol{x}))^{\alpha_l} + L_l\mathrm{w}(\boldsymbol{x}), l = 1, 2.$$

Then, if the width of $\boldsymbol{x}$ is such that

$$c_l(\mathrm{w}(\boldsymbol{x}))^{\alpha_l} + L_l\mathrm{w}(\boldsymbol{x}) < \beta_l, l = 1, 2 \tag{3.10}$$

we will have that

$$f_l(\hat{x}) < f_l(x) + \beta_l, l = 1, 2,$$

that is, $\hat{x}$ (β_1, β_2)-underdominates x. The width of $\boldsymbol{x}$ has to be small enough so that the two conditions in (3.10) hold simultaneously. □

As a consequence, the following result is true. Consider the following stopping rule:

TR3: we send a box $\boldsymbol{x}$ to $\mathcal{L}_{\mathcal{S}}$ provided that:

1. $\mathrm{w}(\boldsymbol{x})$ satisfies (3.10), and
2. $\boldsymbol{x}$ contains at least one feasible point.

Corollary 3.38. *Suppose that the interval algorithm is run with termination rule TR3, and let B be the set of points which contains one feasible point from each box in $\mathcal{L}_{\mathcal{S}}$. Then B is a (β_1, β_2)-underdominator for problem (3.1).*

In particular, if the convergence order of $\boldsymbol{f}_l, l = 1, 2$, is one (for instance, if they are the corresponding natural interval extensions of $f_l, i = 1, 2$, see [125]), then (3.10) holds provided that

$$\mathrm{w}(\boldsymbol{x}) \leq \min \left\{ \frac{\beta_l}{c_l + L_l} : l = 1, 2 \right\}.$$

If $\boldsymbol{f}_l, l = 1, 2$, are quadratically convergent (for instance, if they are centered forms, see [125]), then (3.10) holds provided that

$$\mathrm{w}(\boldsymbol{x}) \leq \min \left\{ \frac{-L_l + \sqrt{L_l^2 + 4\beta_l c_l}}{2c_l} : l = 1, 2 \right\}.$$

Notice also that if in termination rule TR2 we set $\varepsilon_1 = \beta_1$ and $\varepsilon_2 = \beta_2$, then for any box $\boldsymbol{x}$ in $\mathcal{L}_{\mathcal{S}}$ it happens that any feasible point in it, $\hat{x} \in \boldsymbol{x} \cap S$, (β_1, β_2)-underdominates $\boldsymbol{x}$. Thus, if B is the set of points which contains one feasible point from each box in $\mathcal{L}_{\mathcal{S}}$, then B is a (β_1, β_2)-underdominator for problem (3.1).

3.5 Conclusions

In this chapter three methods were introduced for handling biobjective optimization problems. The first method, the Lexicographical-like method is devoted to problems where one of the objective functions is more important than the other, so we are not interested in the whole efficient set, but only in some parts of it. Using the parameter δ, the decision maker can choose the required fraction of the optimal value of the more important objective which can be traded off with a possibly important decrement of the other objective. The algorithm finds the corresponding efficient points.

The second method we developed, the Constraint-Like method, is able to find all the efficient points. In fact, it gives an outer estimation of the weakly efficient set, which is very useful for certain problems. Also, as this method finds parts of the efficient set in the image space from left-top to right-bottom, it can be viewed as an interactive method, i.e. the decision maker can stop the process if the obtained parts of the efficient set give enough information. This method is very useful when one of the objectives is more important than the other, but we are interested in a wide range of efficient solutions.

When none of the objectives is more important then the other, and so we are interested in the whole efficient set without interaction, the biobjective interval Branch and Bound method can be used to obtain a tight enclosure of it. Similarly to the other methods, several biobjective discarding tests were developed, which speed up the optimization process. Moreover, the biobjective interval B&B method has very good properties: we know that the solution set converges to the weakly efficient set, and in finite steps the obtained solution list cannot contain $(2\varepsilon_1, 2\varepsilon_2)$-superdominated points (if in the termination rule the $\varepsilon_1, \varepsilon_2$ parameters were used).

We can say that from this set of methods anybody can find the one which suits his/her requirements best.

Symbol Index

Sets, regions

$\mathbb{R}$	set of reals
$\mathbb{I}$	set of closed intervals, $\mathbb{I} = \{[a,b] \mid a,b \in \mathbb{R}, a \leq b\}$
S	feasible set (it can be given by a set of analytical constraints)
R_δ	region of δ-optimality
OR_δ	outer approximation of R_δ
IR_δ	inner approximation of R_δ
PR	pruneable region
X^*	set of global minimizer points, i.e. $X^* = \{x^* \in S \mid x^* = \text{argmin}_{x \in S} f(x)\}$
S_E	efficient set (in decision space)
Z_N	nondominated set (in criterion space)
S_{WE}	weakly efficient set (in decision space)
Z_{WN}	weakly nondominated set (in criterion space)

Notation of Real Analysis

$x, y, \ldots$	real numbers or vectors; in case of vectors $x = (x_1, \ldots, x_n)^T, x_i \in \mathbb{R}, i = 1, \ldots, n$
$f(x)$	a function, usually the objective function
$f'(x)$	gradient of function f
$H(x)$	Hessian matrix of function f
$h_j(x)$	function of the inequality constraints defining the feasible set S, $j = 1, \ldots, q$

Notation of Interval Analysis

$\boldsymbol{x}, \boldsymbol{y}, \ldots$	intervals or boxes, i.e. $\boldsymbol{x} = [\underline{x}, \overline{x}] \in \mathbb{I}$ or $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T \in \mathbb{I}^n, \boldsymbol{x}_i = [\underline{x}_i, \overline{x}_i] \in \mathbb{I}, i = 1, \ldots, n$
$\underline{x}, \overline{x}$	lower and upper bounds of $\boldsymbol{x}$, i.e. $\underline{x} = \min\{x \in \boldsymbol{x}\}$ and $\overline{x} = \max\{x \in \boldsymbol{x}\}$
$\lfloor x \rceil$	either the lower or the upper bound of box $\boldsymbol{x}$, i.e. $\lfloor x \rceil = \underline{x}, \overline{x}$
$\text{m}(\boldsymbol{x})$	midpoint of $\boldsymbol{x} \in \mathbb{I}$, i.e. $\text{m}(\boldsymbol{x}) = \frac{\underline{x}+\overline{x}}{2}$. If $\boldsymbol{x} \in \mathbb{I}^n$, then $\text{m}(\boldsymbol{x}) = (\text{m}(\boldsymbol{x}_1), \ldots, \text{m}(\boldsymbol{x}_n))^T$
$\text{w}(\boldsymbol{x})$	width of $\boldsymbol{x}$, i.e. $\text{w}(\boldsymbol{x}) = \overline{x} - \underline{x}$ if $\boldsymbol{x} \in \mathbb{I}$, and $\text{w}(\boldsymbol{x}) = \max_{i=1,\ldots,n} \text{w}(\boldsymbol{x}_i)$ if $\boldsymbol{x} \in \mathbb{I}^n$
$\text{w}_{rel}(\boldsymbol{x})$	relative width of $\boldsymbol{x}$, i.e., $\text{w}_{rel}(\boldsymbol{x}) = \text{w}(\boldsymbol{x}) / \max\{1, \min_{x \in \boldsymbol{x}} \lvert x \rvert\}$ if $\boldsymbol{x} \in \mathbb{I}$, and $\text{w}_{rel}(\boldsymbol{x}) = \max_{i=1,\ldots,n} \text{w}_{rel}(\boldsymbol{x}_i)$ if $\boldsymbol{x} \in \mathbb{I}^n$
$\boldsymbol{g}(\boldsymbol{x})$	inclusion of the gradient of function f over interval $\boldsymbol{x}$
$\boldsymbol{g}^l(\boldsymbol{x})$	inclusion of the gradient f_l' of the function f_l over the box $\boldsymbol{x}$, $l = 1, 2$
$\boldsymbol{H}(\boldsymbol{x})$	inclusion of the Hessian matrix H of the function f over the box $\boldsymbol{x}$
$\boldsymbol{h}_j(\boldsymbol{x})$	inclusion function for $h_j(x)$, $j = 1, \ldots, q$

Lists

$\mathcal{L}_{\mathcal{W}}$	working list
$\mathcal{L}_{\mathcal{S}}$	solution list
$\mathcal{L}_{PNS}$	list of provisional nondominated points

Special Intervals

$\boldsymbol{x}(i:p)$	a degenerated n-dimensional interval, $\boldsymbol{x}(i:p) = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, [p,p], \boldsymbol{x}_{x+1}, \ldots, \boldsymbol{x}_n)$
$\boldsymbol{x}_i^l, \boldsymbol{x}_i^r, \boldsymbol{x}_i^m$	special cases of $\boldsymbol{x}(i:p)$, $\boldsymbol{x}_i^l = \boldsymbol{x}(i:\underline{x})$, $\boldsymbol{x}_i^r = \boldsymbol{x}(i:\overline{x})$ and $\boldsymbol{x}_i^m = \boldsymbol{x}(i:\mathrm{m}(x))$
$\boldsymbol{s}$	box containing the search region
$\boldsymbol{bpr}$	box containing the pruneable region PR
$\boldsymbol{ob}$	original box, $\boldsymbol{ob} = \boldsymbol{x} - c$
$\boldsymbol{pb}$	pruneable box
$\boldsymbol{cpb}$	centered pruneable box
$\boldsymbol{spb}$	shifted pruneable box
$\boldsymbol{spb}^{\llcorner ob \urcorner_i}$	pruneable box shifted to $\llcorner ob \urcorner_i$

Other functions

$lbf(\boldsymbol{x})$	lower bound for $f(\boldsymbol{x})$
$sp(\boldsymbol{x})$	support function of $f(\boldsymbol{x})$ at the facets of the interval $\boldsymbol{x}$, i.e. $sp(\boldsymbol{x}) = (sp(\boldsymbol{x}_1), \ldots, sp(\boldsymbol{x}_n))$, where $sp(\boldsymbol{x}_i) = \{sp(\boldsymbol{x}_i^l), sp(\boldsymbol{x}_i^r)\} = \{lbf(\boldsymbol{x}_i^l), lbf(\boldsymbol{x}_i^r)\}, i = 1, \ldots, n$
$A(\boldsymbol{x})$	area of $\boldsymbol{x}$
$V(\boldsymbol{x})$	volume of $\boldsymbol{x}$

Other symbols and coefficients

x^*	global minimizer point, i.e. $x^* = \mathrm{argmin}_{x \in S} f(x)$
f^*	global minimum value, i.e. $f^* = \min_{x \in S} f(x)$
$\tilde{f}$	upper bound of the minimum f^*
α, c	coefficients of the empirical convergence speed
b	Baumann point, $b_i = \frac{\overline{g}_i \underline{x}_i - \underline{g}_i \overline{x}_i}{\overline{g}_i - \underline{g}_i}, i = 1, \ldots, n$
sf_i	shifting factor, $sf_i = \frac{\overline{ob}_i}{\overline{pr}_i} = \frac{\underline{ob}_i}{\underline{pr}_i}$
osf_i	opposite shifting factor, $osf_i = 1 - sf_i$
$osf_{j_1,\ldots,j_k}$	opposite shifting factor for shifting in the components $j_1, \ldots, j_k$, i.e. $osf_i = 1 - \sum_{i=j_1,\ldots,j_k} sf_i$
PI	pruning index
δ	parameter for the region of δ-optimality
η	accuracy of the region of δ-optimality, i.e. the outer approximation of the R_δ may contain $\delta + \eta(1+\delta)$-optimal points, but not worse
θ	parameter for the region of δ-optimality of the second objective in the (δ, θ)-lexicographical method, i.e. $\delta = \theta$
z^*	ideal vector, $z_i^* = \min_{x \in S} f_i(x)$
z^{nad}	nadir vector, $z_i^{nad} = \max\{f_i(x) \mid x \in S_E\}$
$(\tilde{f}_1, \tilde{f}_2)$	provisional nondominated point

Part II

Competitive Location Problems

Chapter 4

Introduction to Competitive Location

Location Science deals with the location of one or more facilities in a way that optimizes a certain objective such as minimizing transportation cost, minimizing the undesired effects produced by the facility, capturing the largest market share, etc. Thus, location analysis encloses a wide range of problems, including location of emergency services, nuclear plants or supermarkets, to name a few. The field is very active and many problems are still being investigated, both from a problem formulation standpoint and from an algorithmic point of view.

In this chapter we give an introduction to competitive location problems. In the following section we describe the basic components of any location problem and give a brief review of solution procedures for continuous location problems. Then, in Section 4.2 we present a short revision of the literature on multiobjective location problems. Finally, we concentrate on competitive location problems, where the main ingredients of such problems are considered and a review of Huff-like models is given. This type of location model is the subject of research in the coming chapters of this thesis, where single objective and biobjective problems are studied. The interested reader is referred to [32, 33, 61, 96, 104] for a more exhaustive introduction to locational analysis.

4.1 Basic location models

In any location problem we can always find the following four components: customers, facilities to be located, a space in which customers and facilities are located, and a metric that indicates distances (or travel times) between customers and facilities. The variation of the characteristics of each of those components leads to different models. Several classifications of the location problems can be obtained depending on the main ingredients which define the model (see[68]). Here we consider the followings:

4.1.1 Location space

It is customary to distinguish between *continuous* location models, i.e., models in which the facilities can be located anywhere in a subregion of the plane, *network* location models, in which the facilities can be located on any vertex or edge of a network, and *discrete* location problems, in which facilities may only be located at some prespecified points. Furthermore, the space available to locate the facilities may be restricted by the introduction of *forbidden zones*, i.e. areas in which facilities cannot be located [55].

Depending on the space, the metric used to measure distances varies. In continuous locations problems, the estimation of distances between points is based purely on the coordinates of the points. The most often used distance measures are the Manhattan and the Euclidean distances. These can be generalized with the weighted ℓ_p-norm, given by

$$l_{k,p}(z) = k \sqrt[p]{|z_1|^p + |z_2|^p}, \quad k > 0, p \geq 1,$$

where $z = (z_1, z_2)$ is a point on the plane and k and p are parameters to be fine-tuned for the specific region considered. The Manhattan and Euclidean distances can be obtained by setting p equal to 1 and 2, respectively ($k = 1$ in both cases). For the general case, the weighted ℓ_p-norm is a good distance predicting function whenever an angle of rotation, θ, is taken into account (the parameter θ is the angle measurement between the originally defined coordinate system and the counter-clockwise rotated coordinate system), see [6]. Nevertheless, another type of norm, namely, the ℓ_{2b}-norm, given by

$$l_{2b}(z) = \sqrt{b_1(z_1)^2 + b_2(z_2)^2}, \quad b_1, b_2 > 0,$$

which has the same number of parameters as the weighted ℓ_p-norm, has also proved to be a good distance predicting function (see [47]). Neither distance function dominates the other. On the contrary, depending on the region considered either norm may be significantly better than the other. Thus, in real applications, both functions should be tailored and the one producing the best results selected.

In contrast, distances in network location problems are typically measured as the shortest route on the network (this usually requires a preprocessing phase to find all shortest paths between points, typically a Floyd algorithm [59]). In discrete problems, a matrix of interdistances is usually available.

As a result, continuous location problems tend to be nonlinear optimization problems, while network and discrete location problems are integer programming/combinatorial optimization problems.

4.1.2 Number of new facilities

We distinguish between single-facility problems, where only one facility is to be located, and multi-facility problems, where more than one facility is to be located. It is also possible that the number of facilities is not known in advance, and its final value depends on other factors, for instance, when the establishment of each facility carries a given cost and a fixed budget is available (see [27]). In multi-facility problems, apart from the location aspect of the problem, one must also decide how to *allocate* costumers to facilities. In some cases, facility planners are the ones who decide to which facility costumers are allocated (specially in the public sector), but in other cases the customers decide which facility to patronize (a system appropriate in the retail sector).

4.1.3 Objectives

In many problems the facilities to be located are assumed to be desirable, in the sense that the closer they are to the costumers, the better the value of the objective function. These models are said to have *pull* objectives [40]. Some classical examples are the minisum (or median or Weber) problems, in which the objective is to minimize the sum of weighted distances between customers and facilities, the minimax (or center or Rawls) problems, where the aim is to make the longest customer-facility distance as short as possible, or covering problems, in which the idea is to locate facilities so as to *cover* customers (usually a customer is said to be covered if (s)he is within a given distance to a facility).

In other models, the facilities are considered to be (ob)noxious or undesirable: customers wish to *push* the facilities as far from them as possible (see [45]). Maximin and maxisum objectives are commonly used for these type of problems, although many other objectives have also been proposed in literature (see for instance [46]).

Other type of models try to achieve *equity*, in the sense that the customer-to-facility distances are as similar to each other as possible. However, although many balancing objectives have been proposed (for a review see [40, 44]), there is no unanimity on how to measure equity. Many other types of objectives exist (those needed to handle hierarchical siting problems, hub location problems, integrated location-routing problems, extensive facilities...) but doing a review of all them is out of the scope of this thesis.

This thesis is about *continuous location models*. There is a vast theory and optimization techniques on continuous location. Since the pioneering works of Weber [151] in 1909 (who introduced the minisum problem) and Weiszfeld [152] in 1937 (who offered the earliest location algorithm), literature on that topic has always been increasing, specially since the 70s, to the point that it now has its own entry

(90B85) in the *Mathematics Subject Classification* used by *Mathematical Reviews* and *Zentralblatt für Mathematic*. Whereas some of the continuous location models are easy to solve (for instance, the classical minisum or Weber problem), other more realistic models lead to global optimization problems which are very hard to solve. Many and diverse global optimization techniques have been proposed to cope with those difficult problems. Among the deterministic methods, the branch-and-bound algorithms have been the most used, for instance, the Big Square Small Square (BSSS) method [76], the Generalized BSSS [120], the Big Triangle Small Triangle (BTST) [34] or interval branch-and-bound methods [46, 49], although other techniques, like the outer approximation (which makes use of the D.C. representation of the objective function, see [140]), have proved to be a good alternative to solve some problems. See [75] for a review of other deterministic global optimization techniques applied to location problems. However, in some cases those exact algorithms are unable to solve some problems: either they are too slow or the computer runs out of memory. Thus, heuristic procedures have to be developed to solve those problems. Usually, researchers propose ad-hoc heuristics for the specific problem they are handling (see [26, 30, 115]) although well known meta-heuristics, such as tabu search, simulated annealing, genetic algorithms, or variable neighborhood search have also been applied to different location problems.

4.2 Multiobjective location problems

In many contexts the consideration of only one objective to be minimized or maximized is not enough to derive good decisions. The judgments of several decision makers can lead to several, usually conflicting, criteria, e.g. maximize the distance of a dumping site to residential areas while at the same time keeping transportation costs as low as possible. That is why apart from the single objective problems, we can also find *multicriteria* optimization approaches in literature, specially for discrete and network location problems (see [20, 113] for reviews of the topic).

In the location of an attractive facility z, every user dp_i, $i = 1 \ldots, n$, wants to have the service as close as possible. This behavior gives rise to a trade-off among users that leads to the multicriteria problem:

$$\begin{array}{ll} \min & \{d_{iz} : i = 1, \ldots, n\} \\ \text{s.t.} & z \in R \end{array} \tag{4.1}$$

where d_{iz} denotes the distance between user dp_i and facility z and R is the set of feasible locations. In location theory, (4.1) is known as *point-objective location problem* and there are many publications dealing with it, either in continuous or discrete space or in networks. Most of them give characterizations of different types of efficient sets, and in some papers procedures are also given to find them (see for instance [10, 78, 116, 117]).

In the point-objective problem the aim is to obtain a general approach to the problem of locating a desirable facility, independent of the criterion. When particular criteria are used, the *median* and the *center* objectives have usually been considered. Specifically, the biobjective problem in which both objectives are to be minimized has attracted special attention in researchers. For instance, to locate a fire station one goal may be to locate the station as close as possible to the farthest potential customer, while another goal may be to locate it as close as possible to the majority of customers. A common approach to cope with this problem is the so called *cent-dian problem*, which consists of minimizing a convex combination of median and center objective functions [79] (a kind of weighting method). In the plane many papers have been devoted to the cent-dian problem, although other biobjective problems have also been considered, specially for the location of semi-desirable facilities, in which the facility is both attractive for some users and undesirable for some residents [9, 136]. Something similar has happened when the location space is a network. In discrete location models, more varied applications have been proposed, and accordingly, more approaches have been proposed to tackle them (interactive methods, goal programming, enumeration procedures, multiple attribute decision making methods,...).

Multiobjective problems with more than two criteria have also been considered. The multicriteria median problem, in which each of the involved objectives is of the median type (see for instance [67]), has been studied in both the plane and networks. Multicriteria problems where all the objectives are either median or center ones have been addressed, too. Other location models involving different objectives have also been proposed, for instance, those dealing with *equity measures* (see [39, 101]).

4.3 Competitive location problems

When locating a new facility, one of the most important points is the existence of *competitors* in the market providing the same goods or service. When no other competitor exists, the locating firm will have the monopoly of the market in that area. In such a case the profit the firm obtains is affected by its decision on location. However, if in the area there already exist other firms offering the same goods, then the location firm will have to compete for the market. Then the profit the firm obtains is also affected by its competitors decisions. Therefore, maximization of profit in presence of competitors is a problem much more difficult to solve than in a monopolistic scenario.

The competition may be *static*, which means that the competitors are already in the market and the owner of the new facility knows their characteristics, or *with foresight*, in which the competitors are not in the market yet but they will be soon after the new facilities enters. In this case, it is necessary to make decisions with foresight about this competition, leading to Stackelberg-type models. Furthermore, if the competitors can change their decisions, then we have a *dynamic model*, in which the existence of *equilibrium* situations is of major concern.

Many competitive location models are available in literature, see for instance, the survey papers [41, 43, 121] and the references therein. Apart from the taxonomy described for any location problem, competitive location problems have other more specific ingredients, which differentiate the existing models.

Demand: It is important to know how demand is distributed. In some models it is assumed that demand is uniformly distributed over a given set, whereas in others it is concentrated at some specific points of the space (called demand points, see [60]). On the other hand, demand can be either *inelastic*, which means that it is known with certainty (this is the case when the goods are *essential* for the costumers) or *elastic* (when the goods are inessential, and then the level of demand depends on other factors, such as the proximity of the facility or the price of the goods, or other externalities).

Customer behavior: Knowing how customers purchase goods among the existing facilities helps to estimate the *market share* captured by each facility (see [31, 42]). An often used rule is that customers only travel to the *nearest/cheapest facility* to make their purchases, as occurs in Hotelling-like models (see [62, 134]). In other models, individuals travel farther than necessary to purchase goods, or they do not buy at the lowest price occasionally. Another rule frequently used in retailing is that each customer patronizes all available facilities offering the goods in a *probabilistic way*, with a probability proportional to her/his *attraction* to each facility, as in Huff-like models.

Attraction function: Although in some models the attraction of a customer towards a given facility is only determined by the distance between them, it is commonly accepted that the choice of a facility usually depends on customer characteristics, facility attributes, and spatial separation between the customer and the facility. One of the first researchers to propose such an attraction function was Huff [83, 84], who assumed that the probability that a customer patronizes a certain facility is proportional to its floor area and inversely proportional to a certain power of its distance from the costumer. Nakanishi and Cooper [109] improved the model by replacing the floor area by a product of several attraction factors and Jain and Mahajan [86] applied the model to food retailing. Hodgson [81] suggested replacing the power of the distance by an exponential function of the distance, which accelerates distance decay. Other authors further generalized the model by considering the patronizing probability of a customer for a facility to be proportional to its attraction towards the facility, expressed as the rate of the facility quality, a parameter which depends on the characteristics of the facility, over a non-negative, non-decreasing function of its distance from the customer (see [26, 114]).

Prices: Some competitive problems may also incorporate pricing decisions as variables in the model [119]. Depending on market conditions, firms typically use one of the two following pricing policies: *mill pricing* (the seller sets a factory price, equal for all the customers in the market, and the buyer takes care of transportation) or *delivered pricing* (the seller charges a specific price in each

market area, which includes the freight cost, and takes care of transport). The former normally occurs when customers are close to firms, such as shopping centers, restaurants, sport centers, etc; the latter may occur when the ratio of transportation cost to the total price paid by the customers is high, as happens in some markets such as cement, sugar, steel, etc. Under both price policies, customers are supposed to buy from the cheapest facility (the one with the lowest 'mill price + transportation cost' or the one offering the lowest *price* in the area they belong to, respectively). Due to its difficulty, most of the models including prices are on discrete space or on networks.

In this thesis we deal with static competitive facility location problems, in which demand is assumed to be inelastic and concentrated on a finite set of demand points. Customers patronize all the available facilities in a probabilistic way, and the attraction function follows a Huff pattern (it depends on both the location and the quality of the facilities, which are the variables of the problems). Prices are not considered as decision variables, but they can be included as a part of the attraction factors that determine the facility quality. Most of existing models of this type are oriented to maximization of market share. A brief revision of the existing literature follows.

In networks, maximizing the market share with location and/or quality as decision variables reduces to find optimal solutions in the set of nodes, i.e. it works as in discrete space (see [114, 138]). Therefore the problem is solved using enumeration (in the single facility case) and/or integer linear programming techniques (in multiple facility location situations), see [1, 16].

In the plane, however, the simple objective of maximizing market share is already neither convex nor concave. So finding optimal locations in planar location space is a hard-to-solve problem even for the simplest case of a single new facility with fixed quality. For this case, the objective function already has a complicated multi-modal shape as can be seen for example in [26]. However, recent research has allowed to solve that problem with exactness: branch and bound algorithms have been proposed in [29] and [100], based on the Big Triangle Small Triangle and on the Big Square Small Square approaches, respectively. As far as we know, apart from the models in this thesis, the quality of the new facility has been considered a variable of the problem only in [123], but there, a Hotelling-type model was considered.

The Huff model in [26] was extended in [27] to the multifacility case, but still assuming that the qualities of the new facilities are fixed and given in advance. For this model some heuristics are offered in [30]. Literature on multiobjective competitive facility location is rather scarce. In fact, apart from the models in this thesis, the paper by Ghosh and Craig [65], which analyzes the case of a franchise in discrete space, seems to be the only work considering more than one objective.

In the following chapters we formulate a general Huff-like model for which we study both single-facility and two-facility problems as well as both single-objective and biobjective problems. As mentioned above, multiobjective competitive location is an unexplored area in continuous competitive location, and this thesis fill this gap in part.

Chapter 5

The single facility planar location and design problem

In this chapter, we consider a single facility location problem with static competition and inelastic demand concentrated in some points. The new facility is to be located in the plane, where forbidden regions are present to avoid locating at the demand points. The costumer behavior is probabilistic, and it is based on an attraction function depending on both the location and the quality of the facilities. The aim is to find the location and the quality of the new facility to be located so as to maximize the *profit* obtained by the chain, to be understood as the income resulting from the market share captured by the chain minus its operational costs.

The structure of the chapter is as follows. In the first section we introduce the new model and discuss its properties [51, 52]. Later, we specify the interval Branch and Bound method used to solve the problem in Section 5.2, where the developed Weiszfeld-like local search algorithm is also described. The effectiveness of the used elimination rules in the interval B&B method are examined for a large set of test problems in Section 5.3, where the two presented algorithms are compared as well. In Section 5.4, the best inclusion function for this class of problems [143] is examined by means of the empirical convergence speed introduced in Section 2.1. Finally, for a complete analysis of the model, we check the sensitivity of the parameters [146] in Section 5.5.

5.1 The model

A chain wants to locate a new single facility in a given area of the plane, where there already exist m facilities offering the same goods or product. The first k of those m facilities ($0 \leq k < m$) belong to the chain. The demand, supposed to be inelastic, is concentrated at n demand points, whose locations dp_i and buying power ω_i are known ($i = 1, \ldots, n$). The location ef_j of the existing facility j and its quality as perceived by demand point i, α_{ij} are also known ($j = 1, \ldots, m$).

The location $z = (z_1, z_2)$ and the quality α of the new facility are to be found. We will denote the three variables of the problem as $nf = (z, \alpha)$. The distance from demand point i to facility j is denoted as d_{ij}, while to the new facility as d_{iz}. The attraction that demand point i feels for an existing facility j is $\frac{\alpha_{ij}}{u_i(d_{ij})}$, while for the new facility is $\frac{\gamma_i \alpha}{u_i(d_{iz})}$, where $u_i(\cdot)$ is a non-negative non-decreasing function and γ_i is the weight for the quality of nf as perceived by demand point i.

These particular attraction functions generalize the proposals in [28, 109, 86, 83] and we may assume that $u_i(d_{ij}) > 0 \; \forall i, j$. Indeed, if in a given problem $u_i(d_{ij}) = 0$ for some i and j, which may only happen if $d_{ij} = 0$, i.e., the facility j coincides with demand point i, then we should consider that the attraction of i towards j is $+\infty$, that is, it is impossible for i to be attracted more by some other facility (including the new one) located elsewhere.

In the spirit of Huff [83], and later generalized in [86, 109], we consider that the patronizing behavior of customers is probabilistic, that is, demand points split their buying power among the facilities pro-

portionally to the attraction they feel for them. In [26] T. Drezner presented a location model following Huff's formulation, in which the quality of the new facility is assumed to be known; profit maximization then reduces to market share maximization. Our model generalizes her work by deciding both on the location and on the quality of the facility to be located. Contrary to other attraction functions, we consider that attractiveness depends on the demand point (in addition to distance), which may be reasonable when socio-economic population characteristics (e.g. average age, income class, accessibility,...) are different from place to place (e.g. city, suburb, village,..). As happens with other ingredients of facility attraction, the parameter that measures the attractiveness of each point can be estimated by sample surveys in the different customers' areas.

By these assumptions the market share captured by the new facility is given by

$$\sum_{i=1}^{n} \omega_i \frac{\dfrac{\gamma_i \alpha}{u_i(d_{iz})}}{\dfrac{\gamma_i \alpha}{u_i(d_{iz})} + \displaystyle\sum_{j=1}^{m} \frac{\alpha_{ij}}{u_i(d_{ij})}}$$

and the total market share attracted by the chain is

$$M(nf) = \sum_{i=1}^{n} \omega_i \frac{\dfrac{\gamma_i \alpha}{u_i(d_{iz})} + \displaystyle\sum_{j=1}^{k} \frac{\alpha_{ij}}{u_i(d_{ij})}}{\dfrac{\gamma_i \alpha}{u_i(d_{iz})} + \displaystyle\sum_{j=1}^{m} \frac{\alpha_{ij}}{u_i(d_{ij})}}.$$

The previous expression can be rewritten as

$$M(nf) = \sum_{i=1}^{n} \omega_i \left(1 - \frac{\displaystyle\sum_{j=k+1}^{m} \frac{\alpha_{ij}}{u_i(d_{ij})}}{\dfrac{\gamma_i \alpha}{u_i(d_{iz})} + \displaystyle\sum_{j=1}^{m} \frac{\alpha_{ij}}{u_i(d_{ij})}} \right),$$

and setting

$$r_i = \sum_{j=1}^{m} \frac{\alpha_{ij}}{u_i(d_{ij})}, \quad s_i = \sum_{j=k+1}^{m} \frac{\alpha_{ij}}{u_i(d_{ij})}, \quad t_i = \omega_i s_i, \quad \Omega = \sum_{i=1}^{n} \omega_i$$

we finally have

$$M(nf) = \Omega - \sum_{i=1}^{n} \frac{t_i u_i(d_{iz})}{\gamma_i \alpha + r_i u_i(d_{iz})}.$$

The problem to be solved is then

$$\begin{array}{ll} \max & \Pi(nf) = F(M(nf)) - G(nf) \\ \text{s.t.} & d_{iz} \geq d_i^{\min},\ i = 1, \ldots, n \\ & \alpha \in [q_{\min}, q_{\max}] \\ & z \in R \subset \mathbb{R}^2 \end{array} \tag{5.1}$$

where $F(\cdot)$ is a strictly increasing differentiable function which transforms the market share into expected sales, $G(nf)$ is a differentiable function which gives the operating cost of the facility nf, and $\Pi(nf)$ is the profit obtained by the chain. Note that this profit disregards the operating costs of the existing facilities of the chain, since these are considered to be constant. The parameters $d_i^{\min} > 0$ and $q_{\min} > 0$ are given thresholds, which guarantee that the new facility is not located over a demand point and that it has a minimum level of quality, respectively. The parameter $q_{\max}$ is the maximum value that

the quality of a facility may take in practice. By R we denote the region of the plane where the new facility can be located.

Function F will often be linear (in problems without economics of scale), $F(M(nf)) = c \cdot M(nf)$, where c is the income per unit of goods sold. Of course, other functions can be more suitable depending on the real problem considered.

Function $G(nf)$ should increase as z approaches one of the demand points, since it is rather likely that around those locations the operational cost of the facility will be higher (due to the value of land and premises, which will make the cost of buying or renting the location higher). On the other hand, G should be a non-decreasing and convex function in variable α, since the more quality we require from the facility, the higher the costs will be, at an increasing rate. We will consider G to be separable, in the form $G(nf) = G_1(z) + G_2(\alpha)$. Possible expressions for G_1 may be $G_1(z) = \sum_{i=1}^{n} \Phi_i(d_{iz})$, with $\Phi_i(d_{iz}) = \omega_i/((d_{iz})^{\phi_{i0}} + \phi_{i1})$, $\phi_{i0}, \phi_{i1} > 0$, or $\Phi_i(d_{iz}) = \omega_i/(e^{\frac{d_{iz}}{\phi_{i0}}} - 1 + \phi_{i1})$, with $\phi_{i0}, \phi_{i1} > 0$ given parameters. As for G_2, it could be in the form $G_2(\alpha) = (\alpha/\rho_0)^{\rho_1}$, $\rho_0 > 0, \rho_1 \geq 1$, or $G_2(\alpha) = e^{\frac{\alpha}{\rho_0}+\rho_1} - e^{\rho_1}$, with $\rho_0 > 0$ and $\rho_1 \in \mathbb{R}$ given values.

As for functions u_i, some examples already proposed in literature are $u_i(d_{iz}) = e^{\lambda_i d_{iz}}$ (see [81]) or $u_i(d_{iz}) = (d_{iz})^{\lambda_i}$ (see [26]), with $\lambda_i > 0$ given parameters.

An example of the objective function for $n = 100, m = 10, k = 4$ can be seen in Figure 5.1 (the rest of the settings are as in Section 5.3.1). As our function is three-dimensional, we can only depict it for a chosen α or for a chosen location.

We show the contours of the objective function together with the existing facilities and forbidden regions around the demand points in Figure 5.1(a). It can be seen from Figure 5.1(b) that the objective function Π is neither convex nor concave and may have several local optima. Figure 5.1(c) illustrates the concavity of Π as a function of α, which can be easily proved for any fixed z.

5.2 The used interval Branch and Bound method

In order to have problem (5.1) in the general form as (1.1) with constraints as in (1.6), we rewrite it into the following form:

$$\begin{aligned} \min \quad & f(x) = -\Pi(nf) \\ \text{s.t.} \quad & h_j(x) \leq 0, \quad j = 1, \ldots, q, \\ & x \in \boldsymbol{s} \subseteq \mathbb{R}^3, \end{aligned}$$

where among the general constraints $h_j(x) \leq 0, j = 1 \ldots, q$ we have, as in the previous section, the inequalities $d_{iz} \geq d_i^{\min}\ \forall i = 1, \ldots, n$ and the description of R. By $\boldsymbol{s}$ we denote an initial box containing the region to which the search for optimal vectors can be restricted. Note that $\alpha \in [q_{\min}, q_{\max}]$ is given by $\boldsymbol{s}_3 = [q_{\min}, q_{\max}]$.

Due to the non-concavity (and non-convexity) of the objective function, a global optimization technique is required in order to obtain its globally optimal solution. Therefore we used an interval Branch and Bound method to enclose the solution, namely Algorithm 1.1. We have used the following accelerating devices: feasibility test, cut-off test (see Section 1.4), pruning test (Section 2.2.2), monotonicity test for feasible and undetermined boxes (Section 2.4.1), and the projected one-dimensional interval Newton method (see Section 2.4.2) for the quality variable (we know that the objective is concave as a function of α, see Figure 5.1(c)). In order to find a good upper bound on the minimum as soon as possible, we have used a Weiszfeld-like local search method, described in Section 5.2.1

For this study, we used the pruning method of Section 2.2.2 combined with an *adaptive multisection* rule presented in Markót *et al.* [97]. The rationale behind the use of the adaptive multisection is as follows. If for a given box we knew in advance the number of subboxes into which it should be subdivided in order to be able to fathom the region covered by the box, then we could save computational effort by immediately dividing the box into that number of subboxes. This means that using the same subdivision rule (bisection, tetrasection, multisection, ...) for all the boxes leads to unnecessary computation. This fact was pointed out by Casado *et al.* in [11], where a heuristic *adaptive* multisection rule is proposed, based on experiences that suggest the subdivision of the current box into a larger number

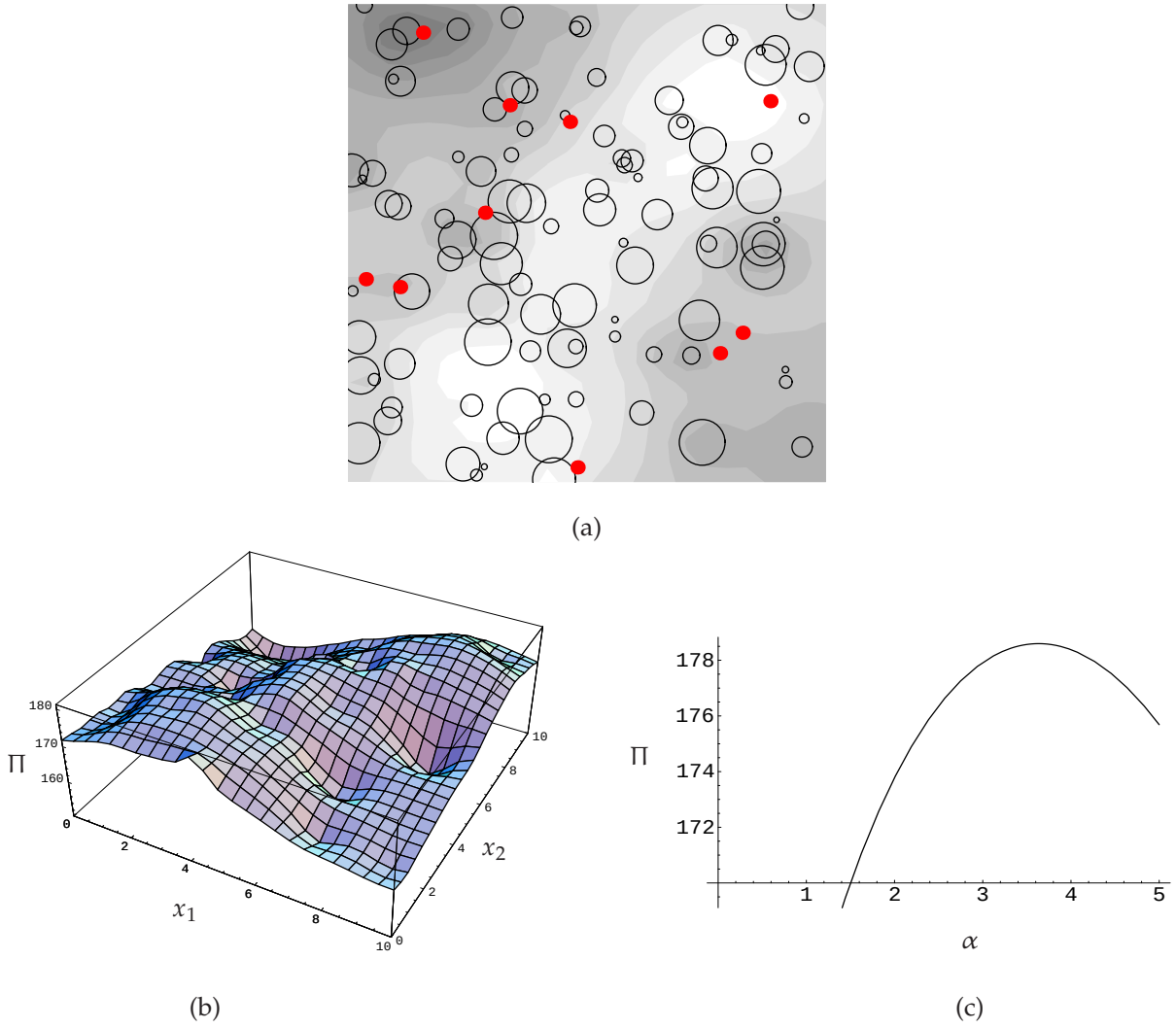


Figure 5.1: (a) Contours of the objective function $\Pi(z, 2.9)$ (shades of gray), existing facilities (solid circles) and constraints around the demand points (ellipses),(b) the objective function $\Pi(z, 2.9)$, and (b) for $\Pi((5,5), \alpha)$.

of pieces only if it is located in the neighborhood of a maximizer point. Of course, an index measuring the closeness of a box to a maximizer point is needed. Let $\tilde{f}$ denote the available best upper bound of the global maximum, as usual. For constrained problems, Markót *et al.* [97] propose the index

$$pup(\boldsymbol{x}, \tilde{f}) = \frac{\tilde{f} - \underline{\boldsymbol{f}}(\boldsymbol{x})}{\mathrm{w}(\boldsymbol{f}(\boldsymbol{x}))} \prod_{j=1}^{q} \min \left\{ \frac{-\underline{\boldsymbol{h}_j}(\boldsymbol{x})}{\mathrm{w}(\boldsymbol{h}_j(\boldsymbol{x}))}, 1 \right\}, \tag{5.2}$$

which depends on the *quality* of the inclusion of the objective value over the box as well as on its *feasibility degree*. The higher the index value, the closer the box to the region of attraction to a minimizer point. Our subdivision strategy is then as follows. If $pup(\boldsymbol{x}, \tilde{f}) < c_1$ we subdivide $\boldsymbol{x}$ along one coordinate direction generating at most two subboxes; if $c_1 \leq pup(\boldsymbol{x}, \tilde{f}) < c_2$ we subdivide along two directions generating at most four subboxes; otherwise we subdivide along all the three directions generating at most eight subboxes (in our computational studies we have set $c_1 = 0.01, c_2 = 0.47$). But we have not performed a simple bisection along the selected directions, as done in [97]. Instead, we have applied the pruning test introduced in Section 2.2.2.

We also applied an easy but quite useful trick to accelerate the method. After dividing a box (or applying the pruning method), an inclusion of the objective is calculated for the new subboxes. This evaluation is necessary because we use the lower bound of the objective function to store the boxes in order and to select the next box to be processed. Usually, the natural interval extension is used for this evaluation (the gradient is only evaluated after the box is selected from the working list). We propose using the centered form using the gradient and center point of the parent box. In this way we can obtain a good inclusion of the objective without additional evaluation. Let us call this accelerating device *Evaluation with Parent Gradient* (EPG).

5.2.1 A Weiszfeld-like local search algorithm

A necessary condition for a vector nf^* to be a local or global optimum of the profit function $\Pi(nf) = F(M(nf)) - \sum_{i=1}^{n} \Phi_i(d_{iz}) - G_2(\alpha)$ is that the partial derivatives of Π at that point must vanish.

With regard to the locational variables, z_p, $p = 1, 2$, we have that

$$\frac{\partial \Pi}{\partial z_p} = 0 \Longleftrightarrow \frac{\mathrm{d}F}{\mathrm{d}M} \cdot \frac{\partial M}{\partial z_p} - \sum_{i=1}^{n} \frac{\partial \Phi_i}{\partial z_p} = 0, \qquad p = 1, 2.$$

Doing some calculations, the previous expressions are equivalent to

$$\frac{\mathrm{d}F}{\mathrm{d}M} \cdot \sum_{i=1}^{n} \frac{-\alpha\gamma_i t_i u_i'(d_{iz})}{(\gamma_i\alpha + r_i u_i(d_{iz}))^2} \cdot \frac{\partial d_{iz}}{\partial z_p} - \sum_{i=1}^{n} \frac{\mathrm{d}\Phi_i}{\mathrm{d}d_{iz}} \frac{\partial d_{iz}}{\partial z_p} = 0, \qquad p = 1, 2,$$

which can be rewritten as

$$\sum_{i=1}^{n} H_i(nf) \frac{\partial d_{iz}}{\partial z_p} = 0, \qquad p = 1, 2, \tag{5.3}$$

where $H_i(nf) = -\dfrac{\partial \Pi}{\partial d_{iz}} = \dfrac{\mathrm{d}F}{\mathrm{d}M} \cdot \dfrac{\alpha\gamma_i t_i u_i'(d_{iz})}{(\gamma_i\alpha + r_i u_i(d_{iz}))^2} + \dfrac{\mathrm{d}\Phi_i}{\mathrm{d}d_{iz}}$. Furthermore, if d_{iz} is a distance function such that

$$\frac{\partial d_{iz}}{\partial z_p} = z_p A_{ip}(z) - B_{ip}(z), \qquad p = 1, 2,$$

where $A_{il}(z)$ and $B_{il}(z)$ are functions of z, then

$$\frac{\partial \Pi}{\partial z_p} = 0 \Longleftrightarrow z_p = \frac{\sum_{i=1}^{n} H_i(nf) B_{ip}(z)}{\sum_{i=1}^{n} H_i(nf) A_{ip}(z)}, \qquad p = 1, 2.$$

With regard to quality variable α,

$$\frac{\partial \Pi}{\partial \alpha} = 0 \Longleftrightarrow \frac{\mathrm{d}F}{\mathrm{d}M} \cdot \sum_{i=1}^{n} \frac{\gamma_i t_i u_i(d_{iz})}{(\gamma_i\alpha + r_i u_i(d_{iz}))^2} - \frac{\mathrm{d}G_2}{\mathrm{d}\alpha} = 0.$$

Notice that this last equation, when we fix $z = (z_1, z_2)$, has just one variable, α. Thus we could solve it by using any algorithm for solving equations of a single variable. Furthermore, Π, as a function of the α variable alone, is concave: regardless the strictly increasing differentiable function $F(\cdot)$ considered, the second derivative of Π with regard to α is negative, since

$$\frac{\partial^2 \Pi}{\partial \alpha^2} = -\frac{\mathrm{d}F}{\mathrm{d}M} \cdot \sum_{i=1}^{n} \frac{2\gamma_i^2 t_i u_i(d_{iz})(\gamma_i\alpha + r_i u_i(d_{iz}))}{(\gamma_i\alpha + r_i u_i(d_{iz}))^4} - \frac{\mathrm{d}^2 G_2}{\mathrm{d}\alpha^2},$$

and $\mathrm{d}^2 G_2 / \mathrm{d}\alpha^2 > 0 \ \forall \alpha > 0$ for any of the expressions of G_2 previously proposed. So, any solution of the previous equation is guaranteed to be a global optimum.

Among the distance functions that satisfy the conditions (5.3) we have the inflated Euclidean distance or its rescaled version, the l_{2b} norm, given by

$$d_{iz} = \sqrt{b_1(z_1 - dp_{i1})^2 + b_2(z_2 - dp_{i2})^2},$$

where $b_1, b_2 > 0$ are given parameters (see [47]). In this case $A_{i1} = \frac{b_1}{d_{iz}}$ and $B_{i1} = \frac{p_{i1}b_1}{d_{iz}}$. Thus, the following Weiszfeld-like algorithm can be constructed.

Weiszfeld-like local search algorithm

1. Begin with a starting vector $nf^{(0)} = (z_1^{(0)}, z_2^{(0)}, \alpha^{(0)})$.
   Set iteration counter $r = 0$.
2. Calculate the next iterate $nf^{(r+1)} = (z_1^{(r+1)}, z_2^{(r+1)}, \alpha^{(r+1)})$ as follows:

$$z_p^{(r+1)} = \frac{\sum_{i=1}^{n} H_i(nf^{(r)}) B_{ip}(z^{(r)})}{\sum_{i=1}^{n} H_i(nf^{(r)}) A_{ip}(z^{(r)})}, \qquad p = 1, 2$$

   and $\alpha^{(r+1)}$ as a solution of the equation

$$\frac{\mathrm{d}F}{\mathrm{d}M} \cdot \sum_{i=1}^{n} \frac{\gamma_i t_i u_i(d_{iz^{(r+1)}})}{(\gamma_i \alpha + r_i u_i(d_{iz^{(r+1)}}))^2} - \frac{\mathrm{d}G_2}{\mathrm{d}\alpha} = 0$$

3. If $nf^{(r+1)}$ is infeasible then set $nf^{(r+1)}$ equal to a point in the segment $[nf^{(r)}, nf^{(r+1)}]$ which is on the border of the feasible region.
4. If the distance between the consecutive iterative vectors $nf^{(r)}$ and $nf^{(r+1)}$ is less than a given tolerance or $r > r_{max}$ then

   STOP: accept $nf^{(r+1)}$ as a potential local maximum.

   Else set $r = r + 1$ and go to step 2.

Let us give some comments on the algorithm. When solving the equation for $\alpha^{(r+1)}$ in Step 2, we do not need to obtain $\alpha^{(r+1)}$ exactly: an approximation is sufficient. So, when using a solution procedure for the equation, we do not need to wait for the convergence of the procedure: a few iterations may be enough. Similarly, in Step 3, it is not necessary to obtain an exact boundary point; any close but feasible point suffices. For the first stopping criterion in Step 4, different functions can be used to measure the closeness of two consecutive vectors. For instance, one of them may be to stop the algorithm if $\|z^{(r)} - z^{(r+1)}\|_2 < \varepsilon_1$ and $|\alpha^{(r)} - \alpha^{(r+1)}| < \varepsilon_2$, for given tolerances $\varepsilon_1, \varepsilon_2 > 0$. The second stopping criterion setting a maximum number of iterations r_{max} is necessary because the convergence of the algorithm cannot be guaranteed.

5.3 Computational experiments

All the computational results in this section have been obtained under Linux on a Pentium IV with 2.66GHz CPU and 2GB memory. The algorithms have been implemented in C++. For the Interval B&B method we used the interval arithmetic in the PROFIL/BIAS library [91], and the automatic differentiation of the C++ Toolbox library [69].

5.3.1 The test problems

In order to have an overall view on the performance of the different discarding tests, we have generated different types of problems, varying the number of demand points ($n = 50$ or 100), the number of existing facilities ($m = 2$, 5 or 10) and the number of those facilities belonging to the chain ($k = 0$ or 1 for $m = 2$, $k = 0$, 1 or 2 for $m = 5$ and $k = 0$, 2 or 4 for $m = 10$). For every type of setting 10 problems were generated, by randomly choosing the parameters of the problems uniformly within the following intervals:

- $ef_j, dp_i \in [0, 10]^2$,
- $\omega_i \in [1, 10]$,
- $\gamma_i \in [0.75, 1.25]$,
- $\alpha_{ij} \in [0.5, 5]$,
- $\phi_{i0} = \phi_0 = 2, \phi_{i1} \in [0.5, 2]$, the parameters for $\Phi_i(d_{iz}) = \omega_i \frac{1}{(d_{iz})^{\phi_{i0}+\phi_{i1}}}$
- $\rho_0 \in [7, 9], \rho_1 \in [4, 4.5]$, the parameters for $G_2(\alpha) = e^{\frac{\alpha}{\rho_0}+\rho_1} - e^{\rho_1}$,
- $c \in [1, 2]$, the parameter for $F(M(nf)) = c \cdot M(nf)$,
- $b_1, b_2 \in [1, 2]$, parameters for $d_{iz} = \sqrt{b_1(z_1 - dp_{i1})^2 + b_2(z_2 - dp_{i2})^2}$,
- $\lambda_i = \lambda = 2$, the parameter for $u_i(d_{iz}) = (d_{iz})^{\lambda_i}$.

The search space for every problem was $z \in [0, 10]^2, \alpha \in [0.5, 5]$. The algorithm was run with all equal tolerances, set to 10^{-4} or 10^{-2}.

5.3.2 Numerical results

The performance of the applied discarding tests is examined. We denote as **Basic** the traditional interval B&B method described in Algorithm 1.1, using the centered form as inclusion function, and only the cut-off test as discarding test. **Basic+Mono** denotes the algorithm, where in addition to the *Basic* method the monotonicity test for feasible and undetermined boxes (see Section 2.4.1) is applied. Similarly, in **Basic+Prune**, in addition to the *Basic* method we apply the pruning method (described in Section 2.2.2) combined with the pup index described in the previous section. Finally, the **Basic+Newton** method applies the projected one-dimensional interval Newton method (see Section 2.4.2) for the variable of the quality.

Table 5.1 shows the usefulness of each discarding test. In rows we give the average of the results of all problems with the given number of demand points and existing facilities (regardless the chain length), and in addition we give the overall average for all the problems in the last row. For all methods

Table 5.1: Efficiency of the different discarding tests.

Problem		Basic		Basic+Mono			Basic+Prune			Basic+Newton		
Type	CPU	NE	ML	CPU	NE	ML	CPU	NE	ML	CPU	NE	ML
n, m	(sec)	#	#	%	%	%	%	%	%	%	%	%
50, 2	29.5	124 770	6597	48.1	51.3	51.9	65.1	68.4	57.4	64.1	65.7	58.7
50, 5	63.0	277 028	14453	58.3	57.9	61.3	52.5	52.8	45.5	72.7	72.6	65.0
50, 10	92.6	414 387	27285	60.8	58.9	66.8	43.5	41.8	40.3	78.1	74.4	69.7
100, 2	52.7	120 653	6868	35.9	36.4	38.4	55.4	54.6	50.2	49.0	46.6	46.9
100, 5	112.8	243 583	15534	39.0	38.6	36.8	51.3	51.9	44.1	44.6	46.4	39.8
100, 10	279.6	610 192	39870	51.6	52.0	55.0	37.8	37.9	36.3	67.0	65.5	57.1
Average:	105.0	298 435	18434	49.9	51.6	55.0	45.3	46.2	41.7	63.6	64.8	58.3

Table 5.2: Efficiency of the Evaluation with Parent Gradient, the local search and all the accelerating devices together.

Problem	Basic			Basic+EPG			Basic+Local			Basic+All		
Type	CPU	NE	ML	CPU	NE	ML	CPU	NE	ML	CPU	NE	ML
n, m	(sec)	#	#	%	%	%	%	%	%	%	%	%
50, 2	29.5	124770	6597	59.0	59.0	74.5	101.4	98.8	99.5	27.8	27.8	30.3
50, 5	63.0	277028	14453	65.2	62.4	83.2	102.1	99.5	99.5	24.9	23.8	27.8
50, 10	92.6	414387	27285	63.5	62.6	87.5	104.8	99.9	99.4	21.9	20.8	27.1
100, 2	52.7	120653	6868	59.2	58.0	76.4	93.5	88.0	97.5	19.0	17.9	21.4
100, 5	112.8	243583	15534	61.9	62.7	84.2	94.6	95.1	98.0	15.5	16.1	17.1
100, 10	279.6	610192	39870	65.8	65.6	87.3	96.0	99.2	99.4	15.3	15.5	18.9
Average:	105.0	298435	18434	63.9	63.1	84.9	97.7	98.0	99.1	18.2	19.1	22.7

we give information about the time required (CPU), the maximal list length (ML) and the number of evaluations (NE), where NE=NF+n·NG+n^2·NH (NF is the number of inclusion function evaluations, NG the number of gradient evaluations and NH the number of evaluations for the Hessian matrix). Notice that using the projected one-dimensional interval Newton method only one element of the Hessian is calculated, therefore in that case the coefficient of NH is one.) In Table 5.1, for the *basic* method we give the obtained values, while for the rest of the methods the proportion of time, evaluation and memory requirements with respect to the *Basic* method are given in percentages. We can see that all the discarding tests work very well, and reduce the required time by 40-50%. For the number of evaluations and the maximal list length similar reductions are achieved. Let us note, that for the *Basic+Mono* and *Basic+Newton* methods for smaller m we achieve higher reduction, while for the *Basic+Prune* method the higher the m, the higher the reduction. Conversely, for all three methods it happens that if n is higher, the reduction is also higher.

The performance of the rest of the accelerating devices are examined in Table 5.2. **Basic+EPG** denotes the *Basic* method used together with the *Evaluation with Parent Gradient* accelerating device, i.e. evaluating the centered form with the gradient and center point of the parent box for the new subboxes after division (or pruning test). With **Basic+Local** we denote the method where in addition to the *Basic* method the Weiszfeld-like local search algorithm is used to speed up the process. In the last columns we give the results for the method called **Basic+All**, where all the discarding tests, the local search and the Evaluation with Parent Gradient accelerating device was used. One can see, that using EPG we can save nearly 40% of the required time and evaluation in average, and 15% of the memory requirements. This means that not only the inclusion function evaluations for the new subboxes are saved, but also that the inclusion by the parent's gradient gave a better inclusion than the natural extension. Hence, the algorithm was able to remove some of those boxes before storing them in the working list. Notice, that the number of demand points (n) does not affect the speed-up, while for fewer existing facilities (m) we achieve smaller ratio of work. The Weiszfeld-like local search could help only in the problems with 100 demand points (better for fewer existing facilities), and in average 2% of the running time is reduced. It means that the interval B&B method finds the region of optimal solutions quite early, so the Weiszfeld-like method does not give much better upper bound on the minimum. In the last columns we can see, that using all the accelerating devices the optimization process can be reduced by more than 80%. The harder the problem is, the more reduction we obtained, therefore the running times for the different types of problems become more homogeneous using all the accelerating devices as compared to the Basic method. We can see that the achieved reduction rate is not far from the multiplication of the reduction rates of each accelarating devices, therefore we can conclude that the discarding tests usually eliminate different type of boxes or different parts of boxes and so the accelerating devices are more or less independent.

In Table 5.3 we examine the usefulness of the modifications we made on the existing discarding tests. We examined the Basic method together with one of the following discarding tests: the monotonicity

Table 5.3: Efficiency of the modified discarding tests compared to the original ones.

Problem	Monotonicity			Pruning			Newton			Mon+Pru+New		
Type	CPU	NE	ML	CPU	NE	ML	CPU	NE	ML	CPU	NE	ML
n, m	%	%	%	%	%	%	%	%	%	%	%	%
50, 2	71.4	81.3	71.9	89.3	90.1	141.9	82.2	85.6	76.6	67.8	75.6	115.0
50, 5	80.7	81.8	77.7	90.4	91.7	149.6	80.9	84.4	74.5	78.0	83.5	123.8
50, 10	78.1	76.1	82.9	96.0	92.7	148.7	65.4	65.6	74.7	62.6	70.3	132.7
100, 2	65.2	65.6	54.4	82.3	88.3	139.5	60.7	60.3	64.3	52.0	57.4	83.4
100, 5	64.5	65.2	52.9	93.4	92.7	146.4	46.7	48.2	46.3	57.7	64.4	92.4
100, 10	73.0	73.9	75.1	94.5	93.0	150.7	59.5	58.0	63.9	66.7	70.4	118.9
Average:	72.7	74.8	73.2	92.2	92.0	147.8	61.1	63.7	66.4	64.3	71.3	116.7

test of Section 1.4, and monotonicity test for feasible and undetermined boxes of Section 2.4.1; the pruning method of Section 2.2.2, and the pruning method of Section 2.2.2 combined with the pup index; the interval Newton step of Section 1.4, and the projected one-dimensional interval Newton method of Section 2.4.2. For every modified discarding test we give, in percentages, the reduced time, evaluation and memory requirements compared to the original ones. For instance, in the CPU column of Monotonicity we show the proportion of time in percentages of the Basic+Mono method with respect to the Basic method using the monotonicity test of Section 1.4. We can see that while both the modified monotonicity and Newton test speed up the method considerably, the modified pruning test does not achieve more than an 8% reduction in time and evalutaion. It is even seen that, in memory requirements it worsens the results of the original pruning method. This happens because using the pup index, for promising boxes, the algorithm generates more new subboxes, thus increasing the length of the working list. Using all the modified discarding tests the reduction is more than 35% compared to the original tests.

As a final numerical study, we compared the multi-start Weiszfeld-like local search algorithm with the interval B&B method. The algorithms were run with all equal tolerances, either 10^{-2} or 10^{-4} as indicated. The Weiszfeld-like algorithm was run from 100 and 1000 random starting points. In Table 5.4 we present the average results for all the settings regardless the chain length, running the interval B&B method with all the accelerating devices examined so far. We show the difference between the optimal objective value obtained by the interval B&B algorithm and the best solution obtained by the Weiszfeld-like algorithm, in percentages. The column *Times found* refers to the number of times that the Weiszfeld-like algorithm found the best solution it could find; The last two columns show the CPU time in seconds spent by the interval Branch and Bound (B&B) and the Weiszfeld-like (Local) algorithms, respectively.

We can see that increasing n or m implies an increase in the CPU time required by the interval B&B algorithm, while the Weiszfeld-like algorithm is less sensitive to these changes.

Concerning tolerance, the increase form 10^{-2} to 10^{-4} implies an increase about 1.5 times in the CPU time required by both algorithms. However, the quality of the objective value offered by the Weiszfeld-like algorithm usually worsens as the accuracy increases.

Increasing the number of times that the Weiszfeld-like algorithm is run from 100 to 1000 improves the obtained objective value around 50%. However, for every one of the types of problems generated there are always instances in which the difference between the objective value offered by the Weiszfeld-like algorithm and the best upper bound obtained by the interval method is much greater than the average (in 18 of the 24 settings, the maximum is greater than 1% and in 3 cases, it is greater than 4.6%). Even performing 1000 runs, the best solution found by the Weiszfeld-like algorithm is the one found by the interval B&B in only half of the problems. As an average, the best solution found by the Weiszfeld-like algorithm is found only in 7 % to 20 % of the runs, and for every one of the settings there are instances in which the best solution provided by the Weiszfeld-like algorithm is found only once. For instance, for $n = 50$, $m = 10$ and $\varepsilon = 10^{-2}$, an average of only 9% of the 1000 runs of the Weiszfeld-like

Table 5.4: Comparison of the Weiszfeld-like local search and the interval B&B method. Numbers in brackets (.) show the standard deviation. Lines in *italics* are results for 1000 runs of the Weiszfeld-like algorithm (otherwise 100). Lines in **boldface** are results for accuracy 10^{-4} (otherwise 10^{-2}).

Prob.Type n, m	Accuracy	Difference in obj avg (%)	Difference in obj max (%)	Times found	Time spent (sec) B&B	Time spent (sec) Local
50, 2	10^{-2}	0.35 (0.5)	1.60	18.6 (16.8)	6.0 (3.6)	0.6 (0.5)
		0.10 (0.2)	*0.86*	*18.8 (18.5)*		*5.7 (6.1)*
	$\mathbf{10^{-4}}$	**0.31 (0.4)**	**1.12**	**20.6 (19.1)**	**8.0 (4.6)**	**0.8 (0.5)**
		0.19 (0.3)	***1.10***	***19.8 (19.6)***		***8.5 (5.4)***
50, 5	10^{-2}	0.20 (0.6)	3.29	19.2 (21.1)	11.3 (4.9)	0.7 (0.7)
		0.02 (0.1)	*0.24*	*17.6 (20.2)*		*6.8 (7.1)*
	$\mathbf{10^{-4}}$	**0.14 (0.3)**	**1.57**	**19.1 (22.0)**	**14.9 (7.3)**	**1.2 (1.3)**
		0.07 (0.2)	***1.30***	***17.3 (21.5)***		***11.7 (12.4)***
50, 10	10^{-2}	0.44 (0.9)	4.41	9.7 (16.0)	15.3 (5.8)	0.7 (0.9)
		0.29 (0.7)	*3.64*	*9.3 (15.3)*		*6.6 (9.7)*
	$\mathbf{10^{-4}}$	**0.53 (1.1)**	**4.98**	**8.4 (14.9)**	**18.3 (8.3)**	**1.3 (1.6)**
		0.25 (0.7)	***3.53***	***8.2 (15.6)***		***11.8 (13.0)***
100, 2	10^{-2}	0.13 (0.2)	0.65	12.3 (19.1)	7.3 (8.1)	0.9 (0.2)
		0.07 (0.1)	*0.59*	*11.0 (19.0)*		*6.9 (1.6)*
	$\mathbf{10^{-4}}$	**0.16 (0.3)**	**1.20**	**11.8 (19.2)**	**9.7 (10.8)**	**1.2 (0.5)**
		0.05 (0.1)	***0.28***	***10.5 (19.4)***		***11.6 (4.8)***
100, 5	10^{-2}	0.21 (0.5)	2.78	17.0 (18.4)	12.8 (10.1)	1.0 (0.3)
		0.05 (0.2)	*0.89*	*15.0 (16.7)*		*8.7 (1.8)*
	$\mathbf{10^{-4}}$	**0.31 (0.9)**	**4.61**	**14.4 (18.6)**	**17.3 (14.1)**	**1.5 (0.5)**
		0.15 (0.5)	***2.62***	***13.6 (17.9)***		***14.9 (4.5)***
100, 10	10^{-2}	0.16 (0.5)	2.63	8.8 (9.5)	32.4 (20.5)	1.2 (1.0)
		0.09 (0.4)	*2.36*	*7.4 (9.5)*		*9.1 (3.7)*
	$\mathbf{10^{-4}}$	**0.34 (1.1)**	**5.59**	**6.9 (8.7)**	**37.8 (26.9)**	**1.6 (0.4)**
		0.11 (0.5)	***2.61***	***7.0 (9.3)***		***16.6 (5.8)***

algorithm found its best solution. And for those problems, in a computational experiment in which the Weiszfeld-like algorithm was run 100000 times, this could not guarantee that the global optimum was reached.

We also want to point out that although the Weiszfeld-like algorithm produces feasible solutions whose objective value is close to the optimal, the respective locations are not necessarily close to the region of near optimality offered by the interval method. Therefore, the results obtained from the Weiszfeld method cannot be trusted.

5.4 Study of the best inclusion function for the interval B&B method

One of the key points in interval global optimization is the selection of a suitable inclusion function which allows to solve the problem efficiently. Usually, the tighter the inclusions provided by the inclusion function, the better, because this will make the accelerating devices used in the algorithm more effective at discarding boxes. On the other hand, whereas more sophisticated inclusion functions may give tighter inclusions, they require more computational effort than others providing larger overestimations. In Section 2.1 the *empirical convergence speed* of inclusion functions was defined and studied, and proved to be a good indicator of the inclusion precision. If the empirical convergence speed is analyzed

Table 5.5: The determined empirical convergence speed $(\alpha, \log_{10} c)$ values for the studied inclusion functions, using all the examined facility location problems. The determined α values are followed in brackets by the correlation coefficient of the regression.

Param.	Sequence	Natural	Centered	Baumann	Affine	Slope
α	Opt	1.026 (.99)	2.667 (.89)	3.555 (.80)	2.662 (.85)	1.997 (.98)
	Rand	1.067 (.99)	2.368 (.94)	4.546 (.81)	2.901 (.87)	1.993 (.99)
	All	1.060 (.99)	2.422 (.93)	4.300 (.77)	2.853 (.86)	1.993 (.99)
$\log_{10} c$	Opt	2.608	3.937	3.936	3.461	2.052
	Rand	2.712	3.450	-3.379	3.245	2.010
	All	2.697	3.518	-2.161	3.288	2.017

for a given type of functions, then one can select the appropriate inclusion function to be used when solving those type of problems. In this section we present such a study, dealing with the competitive facility location problem described before.

In this case two different types of sets have been considered: sequences of embedded boxes, all containing a global minimizer point, and sequences of random intervals converging to a point. The inclusion functions examined in the study were the natural interval extension, the centered form, Baumann's optimal centered form, the forward slope arithmetic form, and the affine arithmetic form.

For the experiments, the C-XSC-2.0 library [82] was used with the automatic differentiation tool of CToolbox-2.0 software [69]. For the affine arithmetic form, the freely available libaa library [64] was used. When an inclusion was accurate (compared to our range approximation, see Section 2.1), and hence the overestimation was zero, its logarithm was set to -30.

We examined 12 objective functions of the form (5.1). In 6 of them we considered 50 demand points and 5 existing facilities (generated in a random way, as described in Section 5.3.1), and in the other functions 100 demand points and 10 existing facilities. The results obtained were very similar for all the cases but one. In Table 5.5 we can see the approximated α and c values of the empirical convergence speed (see (2.2)) for the inclusion functions studied, when all the computed widths of all the examined functions are considered.

We can see that for both the optimum following sequences (Opt) and the random sequences of boxes (Rand) the Baumann form is the inclusion function with the highest convergence speed, as well as when we consider the computed widths of all the sequences together (All). This is due to the fact that on monotonous boxes the Baumann form is accurate. The second best inclusion is the affine arithmetic form, and in the only analyzed case in which the Baumann was not the best function, it was the affine arithmetic form the one providing the best values. The centered and slope forms follow in this ranking, and finally, as expected, we have the natural interval extension, with an α value very close to the theoretical convergence order of 1.

Figure 5.4 illustrates the investigated empirical convergence speed when we considered the 12 investigated functions and all the sequences. From the picture we can conclude that for the examined functions the empirical convergence speed suggests using the Baumann form for every box with a width smaller than 19.56 and the natural interval extension if the width is larger (see the *Summary* graph in Figure 5.4, in which we can see the breaking point at which changing the inclusion function is recommended).

To check, in practice, the usefulness of the empirical convergence speed, we applied an adaptive multi-inclusion algorithm (Algorithm 2.1), and measured the average time needed for solving the location problems when using different inclusion functions. The obtained results can be seen in Table 5.6. We give the average time needed for solving all the problems. The first four columns refer to the cases when only one inclusion function was used (the results for the natural extension are not given, since the program lasted more than two hours). In the next three columns we have used the natural interval extension in addition to the given inclusion function (notice that the natural interval extension is given as a by-product when evaluating the gradients or slopes with the used libraries). In the last

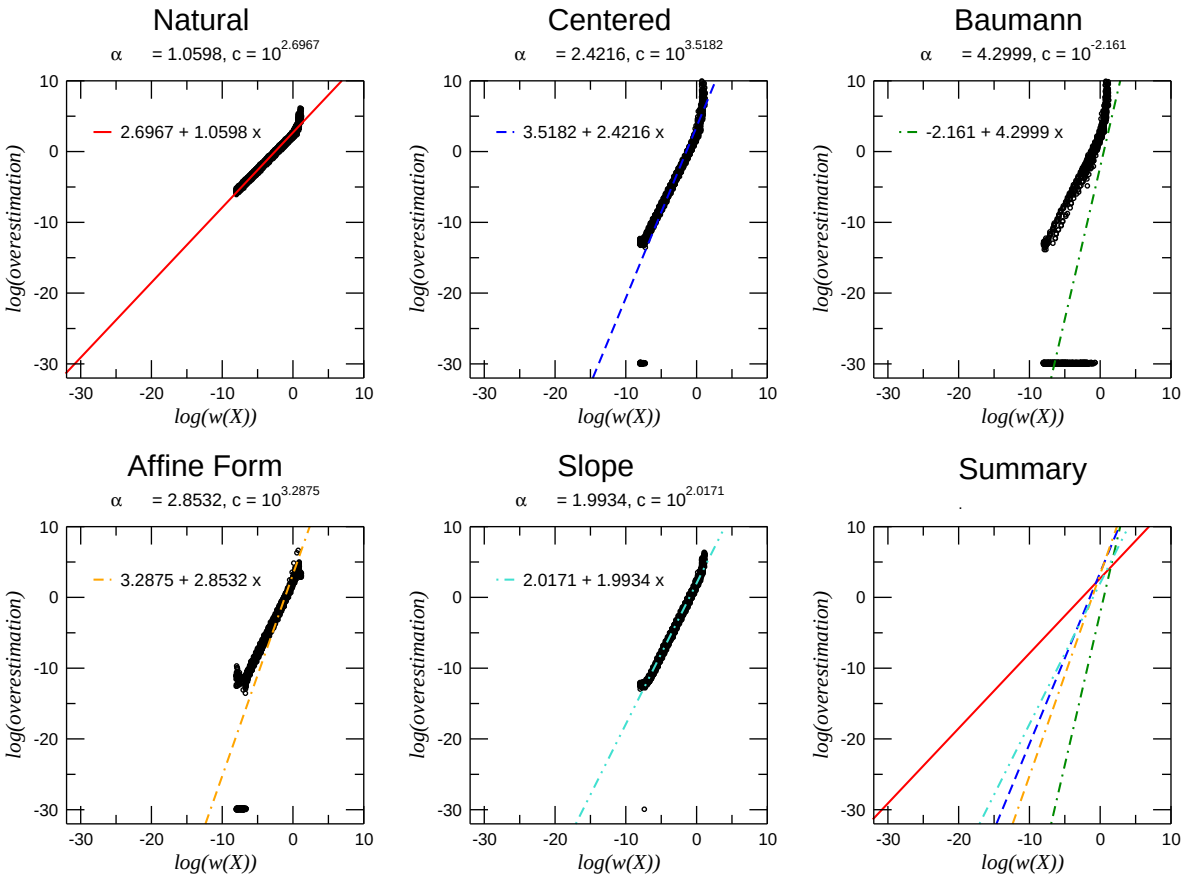


Figure 5.2: The results of the linear regression for all the data on log-log scales indicating that the Baumann form and the natural interval extension are the best choices depending on the width of the box.

column we show an example when we change from one set of inclusion functions to an other. Since the suggestion of the empirical convergence speed is not feasible for this type of problems (the width of the initial box is 10, thus we cannot change at width 19.56), the following setting has been examined only. We change from the natural interval extension to the intersection of the natural, affine and Baumann inclusion when the width of the box to be evaluated becomes less than 1 and the required accuracy is 10^{-8}.

In the rows we give the time needed for solving the problems when the required accuracy is 10^{-2}, 10^{-4} and 10^{-8}, respectively. It can be seen from the table that the results are in correspondence with the suggestion from the empirical convergence speed, i.e. when only one inclusion function is used, the Baumann form is the best. Together with the natural extension, the same holds, and for this type of

Table 5.6: The running times for the solution of the facility location problems in seconds. In the rows we give the results for accuracy 10^{-2}, 10^{-4} and 10^{-8}, respectively.

Accuracy	Only one inclusion function				With natural extension			Switching
	Center	Slope	Affine	Baumann	Center	Slope	Baumann	
10^{-2}	362.2	2683.6	492.7	294.6	346.0	2685.7	281.8	
10^{-4}	480.1	3695.1	642.6	385.6	457.6	3721.0	368.6	
10^{-8}	990.7	—	1653.2	840.9	943.5	—	806.8	922.3

function, as the empirical convergence speed shows, switching from some inclusion functions to others depending on the width of the current box cannot improve the results.

As a future research, we plan to examine other inclusion functions, e.g. the Taylor model [5]. We also plan to consider the effect of other aspects related to the inclusion functions when used in a global optimization algorithm, e.g., other discarding tests.

5.5 Sensitivity analysis

As we have seen, our model is a highly nonlinear optimization problem, and its behavior is not really well understood. It is the aim of this section to contribute to this study, in particular by investigating the changes in optimal design/location when the model environment and parameters change [146]. Additionally, our model typically contains many parameters which might be difficult to estimate correctly. It is therefore important to know how stable the solution is with respect to these parameters, in order to be able to appreciate whether the uncertainty in certain parameters may be ignored or not. This calls for a sensitivity analysis of the optimal solution in terms of changes in the parameters. But how can one correctly appreciate the results of such an analysis without a clear distinction between jumps in the solution due to parameter modification rather than due to algorithm inaccuracy? Therefore, existing local optimization approaches are not tenable for an adequate sensitivity analysis. One way of tackling this difficulty was described by Fernández and Pelegrín [48]: they incorporate the uncertainty on the parameter-values as interval-valued data, specifying both an upper and a lower bound on each uncertain parameter, and apply interval analysis based optimization tools, which construct a close approximation to the set of all possible optimal solutions for any setting of each parameter within its bounds.

On the other hand, recognizing that the model captures only an approximate real-world situation, Plastria [120] argues that, in practice, a good idea of the full set of almost-optimal solutions is more useful than knowledge of an optimal solution, and proceeds to obtain such information by extending the global optimization method BSSS devised by Hansen et al. [76].

But both methods, as well as a more recent variant based on triangle subdivisions [29, 34], were described for a pure location problem only. The method described in Section 5.2 and used in Section 5.3 yields a complete view of the set of close-to-optimal solutions to any desired degree of accuracy, and therefore lends itself better to the proposed sensitivity analysis, although a number of new interpretation difficulties arise, as will be explained in the sequel.

This tool has allowed us to investigate the behavior of the Huff-type competitive location and design problem under various circumstances. We have conducted all experiments on data-sets based on real-world data from the region of Murcia in South-eastern Spain, and observed the variation of the optimal solution set, both in design and in location.

Broadly speaking, we have studied the effects of the following environments and parameters:

- aggregation of demand
- chain superiority versus strong competition
- homogeneous versus widely different facility qualities
- the impact of distance effect decay
- scale effects in income valuation of market share
- the influence of a restricted budget for initial investments
- the impact of push forces away from population centers

A more precise description of these experiments and their results is given in a series of separate sections, after the case data and some necessary concepts are formalized in the next section. Finally we conclude with an overview of the experience gathered through these results and a number of recommendations for analysts and practitioners.

5.5.1 Preliminaries

5.5.1.1 The basic case data

The actual case consisted of an area around the city of Murcia in South-Eastern Spain. The city of Murcia is the capital of the Autonomous Region of Murcia (A.R.M.), a province whose area is 11314 Km^2 and has over one million inhabitants. More than half of the inhabitants of A.R.M. live in the city of Murcia or in the villages close to it.

A working radius of 25 km around the main city was considered suitable for being used in a Huff-type model. Indeed, in this type of model, customers are assumed to patronize *all* existing facilities, so none of the facilities should be too far away from a population center (e.g. within a 15-minute drive, say), because otherwise none of its inhabitants would go there. On the other hand, in our case study, increasing the radius of the circle to 30 km increases the number of covered inhabitants only slightly, whereas reducing the radius to 20 km decreases it considerably.

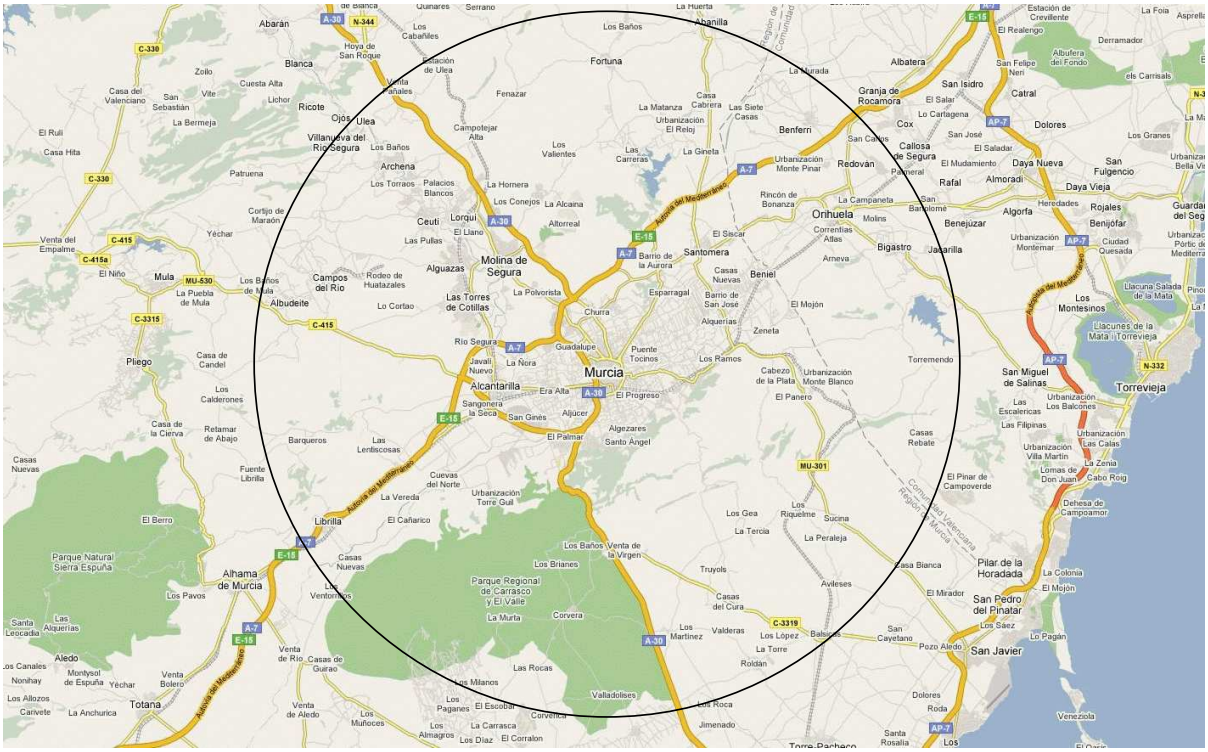


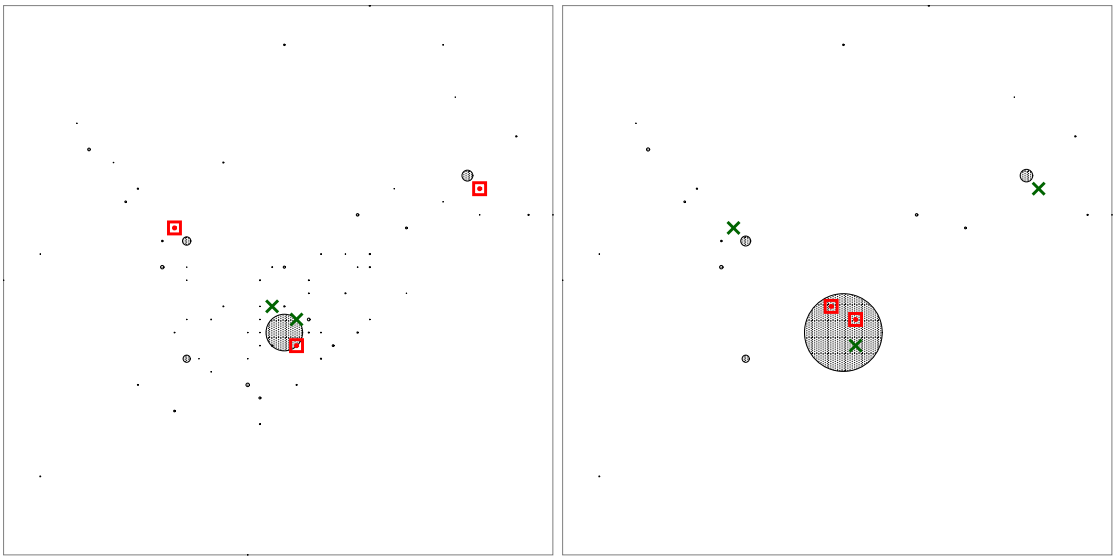
Figure 5.3: Population centers in the neighborhoods of the city of Murcia.

Within a radius of 25 km, centered at Murcia, 557,746 inhabitants of A.R.M. and other 74,812 inhabitants from the neighboring province of Alicante, on the East of A.R.M. live there (see Figure 5.3). These 632,558 inhabitants form our primary set of customers. They are distributed over 71 population centers, with population varying between 1,138 and 178,013 inhabitants. In this study we have considered each population center as a demand point, with buying power proportional to its total population (one unit of buying power per 17,800 inhabitants). Their position and population can be seen in Figure 5.4(a): each demand point is shown as a circle of radius proportional to the buying power.

We have also constructed a reduced data set by aggregation of this demand as follows. Several population centers do not have a city hall, so for public administrative tasks they depend on another town's city hall. The 21 towns with a city hall form our reduced set of demand points, with population obtained by aggregating all population centers at the town on which they administratively depend. The resulting reduced demand distribution is shown in Figure 5.4(b).

Our study deals with the location of supermarkets. There are already five supermarkets present in

the area, two of which belong to one chain, that we will call Small Chain, the three other from another chain, called Large Chain from now on. Figure 5.4 shows the location of each supermarket as viewed from the current chains' point of view: firms belonging to the current chain are marked by a $\times$, and firms from the competing chain are shown by a $\boxdot$ marker on the map.



(a) Full problem for the Small Chain. (b) Small problem for the Large Chain.

Figure 5.4: Some variants of the problem.

For both data sets, the feasible set R was taken exactly as depicted in Figure 5.4, i.e. the smallest rectangle containing all the unaggregated demand points. This is approximately a square centered in Murcia and of sides close to 45 km.

The coordinates of the population centers and the supermarkets were obtained with the Geographical Information System called VisualMap (see [149]), and were rescaled from coordinates $[200, 245] \times [243, 285]$ to an approximate standard square $[0, 10] \times [0, 9.3]$, thus the units correspond approximately to 4.5km. The population data were obtained from the *Regional Center of Statistics of Murcia* (see [13]) and correspond to the year 2002, rescaled to $]0, 10]$. The minimum distance $d_i^{\min}$ at which the new facility must be from the population center i was chosen to be $\frac{\omega_i}{\delta}$, where δ is set to 30. This means that for a demand point with 1,500 inhabitants ($\omega_i = 0.067$) $d_i^{\min} = 0.00223$, and for the largest demand point (Murcia, $\omega_i = 10$) $d_i^{\min} = 0.\dot{3}$. In the aggregated problems, however, each remaining weight is larger, so the corresponding $d_i^{\min}$ is also larger, e.g. aggregated Murcia has weight 21, so $d_i^{\min} = 0.7$. All final data are given in Appendix B.

In order to obtain the quality-parameters α_j for the five existing supermarkets, we carried out a small survey asking people to rank all five supermarkets according to their qualities, and then asking whether the difference between a consecutive pair is smaller, equal or greater than the next consecutive pair. We then translated the results to the interval $[3, 4]$, obtaining the values shown in Appendix B. The interval $[q_{\min}, q_{\max}]$ within which the quality of the new facility must lie was considered $[0.5, 5]$ for all the problems.

Due to the lack of real data from the chains (they consider that data sensitive to them, and are not willing to make them public), the other parameters have been validated in an ad hoc way to obtain *reasonable* results. In particular, we used the following initial settings: the income per unit sold $c = 12$, for the location cost function G_1 we chose the parameter $\phi_{i0} = 2$ for all i, while the value of ϕ_{i1} decreases as the population increases (the actual values can be found in Appendix B). Finally, the parameters of the cost of quality function G_2 were initially set to $\rho_0 = 7$ and $\rho_1 = 3.75$.

5.5.1.2 Representation of the results

Results of an optimization problem may be understood in two ways. On the one hand, there is the optimal value found, and on the other hand, one may look at the optimal solution. In our location-design problem we are clearly more interested in the latter, since it is the choice of the site that matters most, although the actual profit figure is also important for judging the profitability of the project. Therefore we want to represent both types of results, so as to be able to judge the sensitivity of these results when the parameter settings change. Theoretically this seems relatively simple, particularly for the optimal values: these are single values which are easily compared. But the outcome of a traditional local optimization algorithm usually only gives an approximation to the optimal value, very often without any information about the precision of the approximation or, even worse, without a guarantee of having found (an approximation of) the globally optimal value. However, with the global optimization method we have used, we do have quite precise information about the global optimal value: during the optimization stage both an upper and lower bound on the global optimal value are available, and gradually improved until their difference is below a pre-specified precision ε. The resulting upper bound is therefore guaranteed to be an overestimation within ε-accuracy of the actual global optimal value. In our computations we used $\varepsilon = 0.05$. Consequently, when we compare the *optimal* objective values obtained for two different parameter settings, the conclusions can be trusted, as long as they differ by significantly more than 0.05.

The situation is quite more complicated in variable space, when we look at the optimal sites and corresponding qualities found. Traditional methods simply give an approximate solution, which might be close to a local optimal solution (but not necessarily a globally optimal one), without information about how far it might be from such a solution (except possibly some very fuzzy indication one might derive from a Lipschitz constant, which is often awfully overestimated), nor about the possible existence of other (near)-optimal solutions. To cope with these difficulties, we have computed an outer approximation (up to a precision η) of the set of δ-optimal solutions (see Section 1.5) by using the interval version of the GBSSS method [120] described in Section 3.2 and 3.3. To be more precise, it yields a list of solution boxes, and for each box an upper and lower bound on the objective values at any solution within the box. The final result is the list of all boxes lying fully within this δ-optimal region and those on its *boundary*, i.e. probably containing solutions which are not δ-optimal, but not too far, as indicated by the second precision parameter η. The first set contains all boxes for which the lower bound is at least a fraction $1 - \delta$ of the (approximate) global optimal value, the second set consists of boxes with upper bound at least reaching this fraction $1 - \delta$, but lower bound below it, but not by more than η%. This implies that in case the objective is quite flat around a global optimum these boxes may cover a non-negligible area. The advantage of this approach is that if alternative near-global optimal solutions existed anywhere else in the area, this would immediately be apparent from the results, since the lists of boxes would then define several disconnected regions. In our computational tests we used $\delta = 0.01$ and $\eta = 0.002$.

5.5.1.3 Dissimilarity measure

In order to compare the sets of optimal solutions for two or more parameter-settings of our model, we must be able to judge the *likeness* of the outcomes we obtain in each case. Therefore we need a dissimilarity measure between lists of (pairwise non-overlapping) boxes. A list of boxes $\mathcal{L}$ in $\mathbb{R}^d$ is a finite set of non-overlapping d-dimensional boxes. Clearly any box $\boldsymbol{x} = (\boldsymbol{x}_i)_{i=1,\ldots,d}$ has a volume $vol(\boldsymbol{x})$, and the volume of a list is the sum of volumes of its boxes: $vol(\mathcal{L}) = \sum_{\boldsymbol{x}\in\mathcal{L}} vol(\boldsymbol{x})$.

Let us first introduce the following asymmetric connector distance from a box $\boldsymbol{x}$ to the box $\boldsymbol{y}$:

$$\Delta(\boldsymbol{x}, \boldsymbol{y}) = \max_{x\in\boldsymbol{x}} \min_{y\in\boldsymbol{y}} \|x - y\|$$

where $\|.\|$ is the standard Euclidean norm. We then define the following dissimilarity measure for two lists $\mathcal{L}$ and $\mathcal{L}'$:

$$diss(\mathcal{L}, \mathcal{L}') = \sum_{\boldsymbol{x}\in\mathcal{L}} \frac{vol(\boldsymbol{x})}{vol(\mathcal{L})} \min_{\boldsymbol{y}\in\mathcal{L}'} \Delta(\boldsymbol{x}, \boldsymbol{y}) + \sum_{\boldsymbol{y}\in\mathcal{L}'} \frac{vol(\boldsymbol{y})}{vol(\mathcal{L}')} \min_{\boldsymbol{x}\in\mathcal{L}} \Delta(\boldsymbol{y}, \boldsymbol{x})$$

Applied to two single boxes (singleton lists of boxes), this formula yields the symmetrization of Δ as follows

$$diss(\boldsymbol{x}, \boldsymbol{y}) = \Delta(\boldsymbol{x}, \boldsymbol{y}) + \Delta(\boldsymbol{y}, \boldsymbol{x})$$

For instance, in dimension 1, i.e. for two (disjoint) intervals on a line, this corresponds to the sum of the lengths of the two arrows in Figure 5.5. It is similar to the traditional Hausdorff distance between the intervals, but uses a sum instead of a maximum. This means that the one-dimensional dissimilarity roughly doubles the distances between the corresponding bounds.

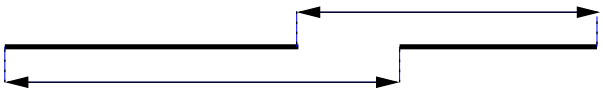


Figure 5.5: The dissimilarity measure for two intervals

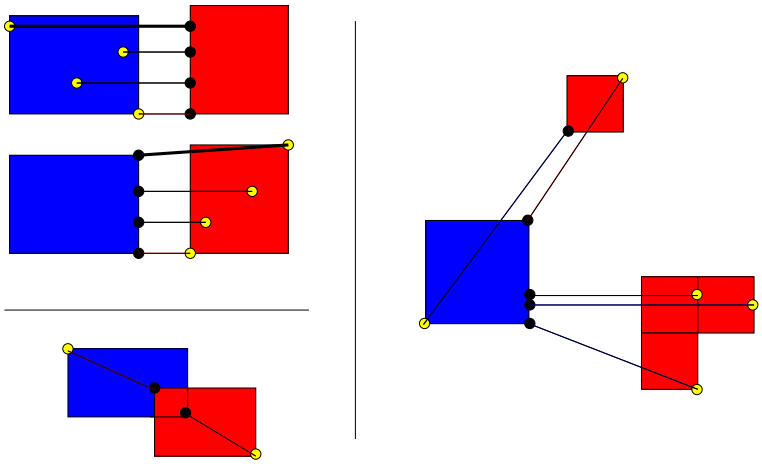


Figure 5.6: The dissimilarity measure for two intervals

Figure 5.6 illustrates this measure for two-dimensional lists of boxes. On the left we have the simplest cases of two singleton lists. At the top, the situation of two non-overlapping boxes is shown twice to differentiate between the two terms of the measure: for each box we take the active connector from this box to the other box separately, i.e. the largest (indicated by the thick connecting line) among all the distances of some point of it to the other box, and then add the resulting lengths. On the bottom left, only the active connectors are shown for two overlapping boxes (the distance $\Delta(\boldsymbol{x}, \boldsymbol{y})$ showed from the yellow dot belonging to $\boldsymbol{x}$ to the black dot belonging to $\boldsymbol{y}$).

An example with two lists of boxes (one being singleton, one consisting of four boxes) is shown on the right hand side. For each box only the shortest of all active connectors to all boxes of the other list is used. Finally, for each list, a convex sum over all its boxes of the lengths of these active connectors is calculated, with coefficients proportional to the volume of the boxes, and these two results are added. This means that large boxes contribute proportionally more to the dissimilarity than small boxes, and that we make some kind of integral over the union of the whole list.

This is not a metric on the set of all lists of boxes, but still has some nice properties. Noting that $\Delta(\boldsymbol{x}, \boldsymbol{y}) = 0$ iff $\boldsymbol{x} \subset \boldsymbol{y}$, we see that $diss(\boldsymbol{x}, \boldsymbol{y}) = 0$ iff $\boldsymbol{x} = \boldsymbol{y}$. It then also readily follows that $diss(\mathcal{L}, \mathcal{L}') = 0$ iff $\cup\mathcal{L} = \cup\mathcal{L}'$, a quite interesting property.

This does not mean, however, that non-zero distance also remains unchanged when one list is replaced by another with the same union, as shown by the following example in two dimensions. As a first list $\mathcal{L}_1$ take the single point (i.e. a singleton list with a degenerate box) $[0, 0]$ and as second list $\mathcal{L}_2$ the single square box $([1, 3], [-1, 1])$. This yields $diss(\mathcal{L}_1, \mathcal{L}_2) = 1 + \sqrt{10}$. But a vertical central split of the box of $\mathcal{L}_2$ yields the alternate list $\mathcal{L}'_2 = \{([1, 2], [-1, 1]), ([2, 3], [-1, 1])\}$ for which

$diss(\mathcal{L}_1, \mathcal{L}'_2) = 1 + \frac{1}{2}(\sqrt{5} + \sqrt{10})$. Note that any horizontal split of $\mathcal{L}_2$ does not change this dissimilarity.

A homothety of all data will multiply the dissimilarity by its factor, and any translation of one of the lists will add (or subtract) at most the translation vector's length to the dissimilarity. A full study falls outside the scope of this work, but it should be clear that lists describing close shapes will be much less dissimilar than lists of very different shapes or lying far apart, while the dissimilarity also says something about the complexity of the list. Therefore this dissimilarity is quite appropriate for our purpose to be able to compare, in a synthetic way, the outcome from optimizing two (apparently slightly different) objectives, as obtained by changing some parameter settings.

This is sufficient to compare the results in variable space. However, for objective values one should not only be able to see the amount of change, but also the direction of the change, either an improvement or a worsening. Since the results come as intervals of values known to bracket the corresponding optimal value, we have to compare two such intervals. However, since the bracket has an (almost) fixed relative size, both extremes will either have increased, and this will be indicated by a $+$, or worsened, indicated by a $-$. In the few cases when the extremes change with different signs, which only happens when the change is very small, no sign notations are used.

5.5.1.4 Structure of the experiments and results

The basic data set described in Section 5.5.1.1 has been used in different ways to produce many variants of the model. First, three chain-structures have been considered:

the *Newcomer* case competing with all five existing facilities,

the *Small Chain* case where the smaller chain wants to strengthen its two existing facilities with a new one to compete on a more equal basis against the three facilities of the Large Chain,

the *Large Chain* case where the larger chain wants to extend its dominance in the market by moving from the current 3 against 2 to a 4 against 2 facility situation.

Moreover, each so defined instance has been solved both in the unaggregated and in the aggregated case. The results obtained in these six problem instances are described in Table 5.7 and depicted in Figures 5.7-5.9.

In the table we first see the maximal value found for the locating chain's profit objective, which is guaranteed to be within $\varepsilon = 0.05$ of the global optimal value. Next we indicate the interval of objective values reached within the $\delta = 1\%$-optimal region, up to a precision of $\eta = 0.2\%$.

When comparing the objective values over the cases, bear in mind these concern the total income from the chain minus the costs of the new facility. Therefore one can compare between unaggregated and corresponding aggregated cases, but not directly between the Newcomer, Small and Large Chain cases.

A much better point of comparison is obtained when looking at the change in market shares after entry of the new facility, which is indicated on the third line of the table. The current values were estimated by applying the model parameter settings to the current location of the existing facilities and our estimation of their quality as given in the appendix. The Newcomer's single facility would be able to obtain a market share of 8.5 out of a total of approx. 35.5, i.e. 24%, a quite amazing success. The Small Chain would see its market share expand from its (estimated) current value of 44% to 53%, thus doing slightly better than its competitor with the same number of facilities. Expansion of the Large Chain raises its current market share of 56% to only 59%, which is rather low considering that this results from twice as many facilities as the competition. This may be explained when looking at the corresponding best solutions found on the fourth row of the table, indicated as the triplet (horizontal position x_1, vertical position x_2, and quality α).

In the unaggregated case, the Newcomer's best strategy turns out to be to go for the largest demand concentration, although the competition there is quite fierce. That is why its quality should be taken as high as possible: the best quality is at the maximum value of 5. The facility is next to Murcia, at the southwestern boundary of the forbidden region, as far as possible from both competitors already

Table 5.7: Results for the basic problems.

		Newcomer	**Small Chain**	**Large Chain**
Optimum (best found)		44.931	210.386	242.957
δ-opt interval		[44.392, 45.021]	[207.866, 210.808]	[240.046, 243.442]
Market share		$0 \rightarrow 8.466$	$15.774 \rightarrow 18.701$	$19.744 \rightarrow 21.144$
Best point found		(4.817, 6.110, 5.000)	(8.398, 3.183, 1.432)	(3.294, 6.481, 0.518)
Hull of results:	x_1	[4.780273, 4.877930]	[8.324116, 8.565161]	[3.074128, 3.572259]
	x_2	[5.991210, 6.189249]	[2.975017, 3.216390]	[6.192808, 6.699219]
	α	[4.560546, 5.000000]	[0.693359, 2.468750]	[0.500000, 1.695313]
Second region:	x_1		[3.253469, 3.310547]	[4.765625, 5.449219]
	x_2		[4.257813, 4.355469]	[5.613055, 6.259766]
	α		[1.343750, 2.082032]	[2.011718, 4.226563]
Volume of the δ-opt		1.7e-04	2.1e-02	1.8e-01
		Aggr. Newcomer	**Aggr. Small Chain**	**Aggr. Large Chain**
Optimum (best found)		26.392	208.301	240.687
δ-opt interval		[26.075, 26.445]	[205.805, 208.718]	[237.803, 241.162]
Market share		$0 \rightarrow 3.961$	$15.524 \rightarrow 18.524$	$19.696 \rightarrow 20.827$
Best point found		(3.247, 4.315, 2.064)	(8.548, 3.0467, 1.467)	(3.280, 6.467, 0.500)
Hull of results:	x_1	[3.238971, 3.286133]	[8.312550, 8.577400]	[3.109303, 3.515625]
	x_2	[4.260253, 4.362793]	[2.963212, 3.226924]	[6.230468, 6.640625]
	α	[1.695312, 2.488910]	[0.763671, 2.433594]	[0.500000, 1.027344]
Second region:	x_1		[3.234792, 3.310547]	
	x_2		[4.218750, 4.371573]	
	α		[1.132812, 2.117188]	
Volume of the δ-opt		2.6e-04	2.0e-02	4.0e-02

present around the city. Since the competitors do not have quality better than 4, it obtains almost one third of Murcia's market, and a quite large part of the demand points in the south-west. This comes with an important cost of around 57, though, as can be evaluated by the difference between sales ($12 * 8.5 = 102$) and actual profit (45).

The Small Chain, however, is already present around Murcia, and can better aim at any of the next largest and cheaper city, Orihuela (note that the vertical dimension is measured southwards), directly in competition with the existing facility of the Large Chain next to it, and at the quite low quality of 1.4, quite easily grabbing part of its competitor's market. This explains the relatively better gain/cost ratio as compared to the Newcomer's strategy.

For the Large Chain the most interesting place is in Alcantarilla, a region currently badly served, and here minimum quality is sufficient to take over the south-eastern market.

The next rows in the table give much more precise information about the near-optimal location and qualities: for each of the three basic variables of the problem it indicates the *hull*, i.e. the range within which it may vary within the near-optimal region. There is a serious problem though: this region might be disconnected, which happens if there exist more than one local optima yielding almost the same global optimal value. In such a case the *naive* hull would enclose all of these connected components, and would be uninformative, since it would yield wide overestimations of the range, but the disconnectedness would not be apparent. Therefore this is given for each connected component separately. This phenomenon happens in the Small Chain case and in the Large Chain case, which explains the "second region" appearing in the table, and is more clearly illustrated in Figures 5.8 and 5.9.

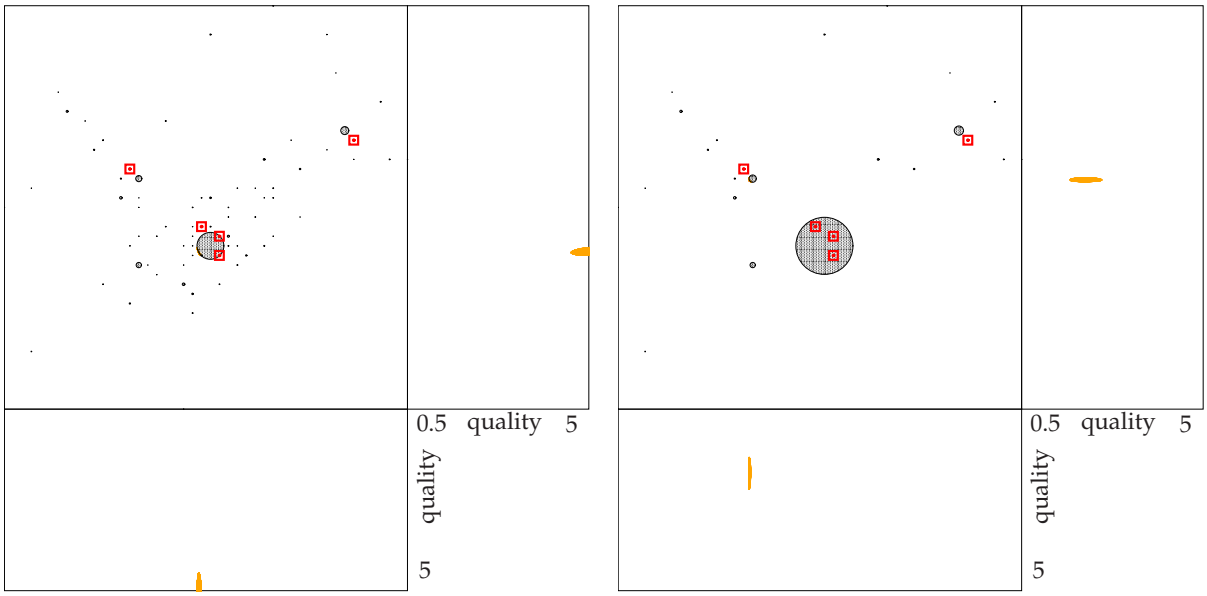


Figure 5.7: Near optimal regions for the Newcomer with unaggregated and aggregated demand.

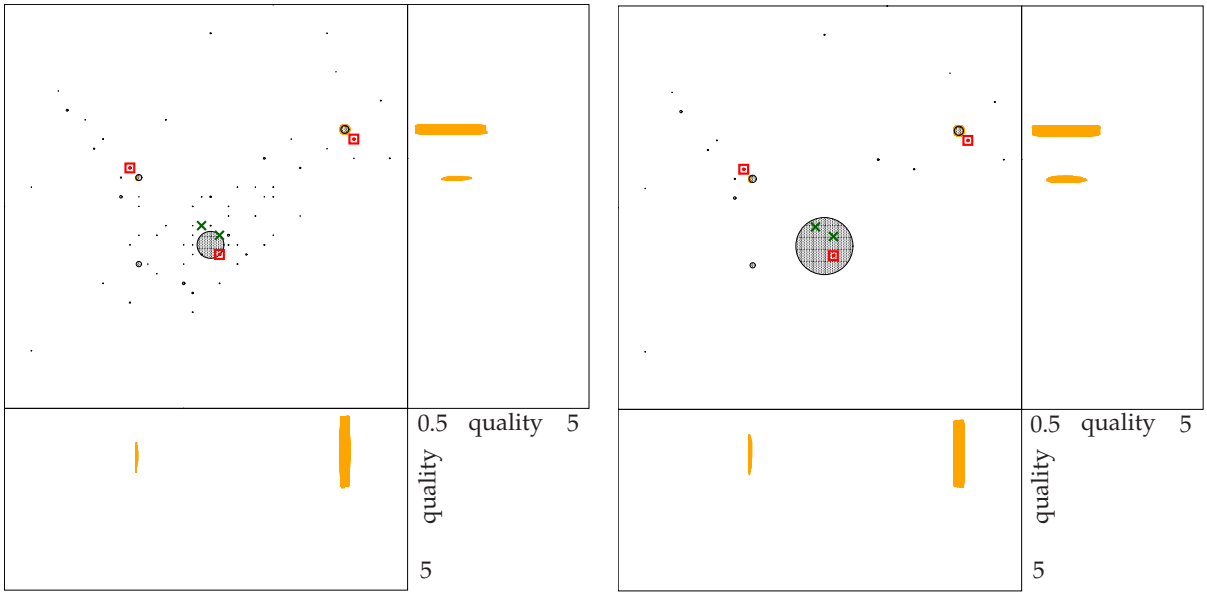


Figure 5.8: Near optimal regions for the Small Chain with unaggregated and aggregated demand.

These figures give a more precise idea of the near-optimal region, reporting two-dimensional information instead of the one-dimensional one in the tables. They describe the interactions between each pair of variables (note that the actual results, being lists of 3-dimensional boxes, are not easily displayed). This is obtained by adding two windows on the right and bottom of the map, allowing to view all three 2-dimensional projections of the 3-dimensional solution set: the map itself shows the 2-dimensional spatial part, without the quality (x_1 increases from left (0) to right (10) and x_2 increases from top (0) to bottom (10)); the right and bottom pane shows the quality projected either to the vertical or to the horizontal part of the space.

Note that often the spatial solution set is quite small or right on the boundary of a forbidden region, and therefore not well visible; one may however know where it is, by combining the regions shown in

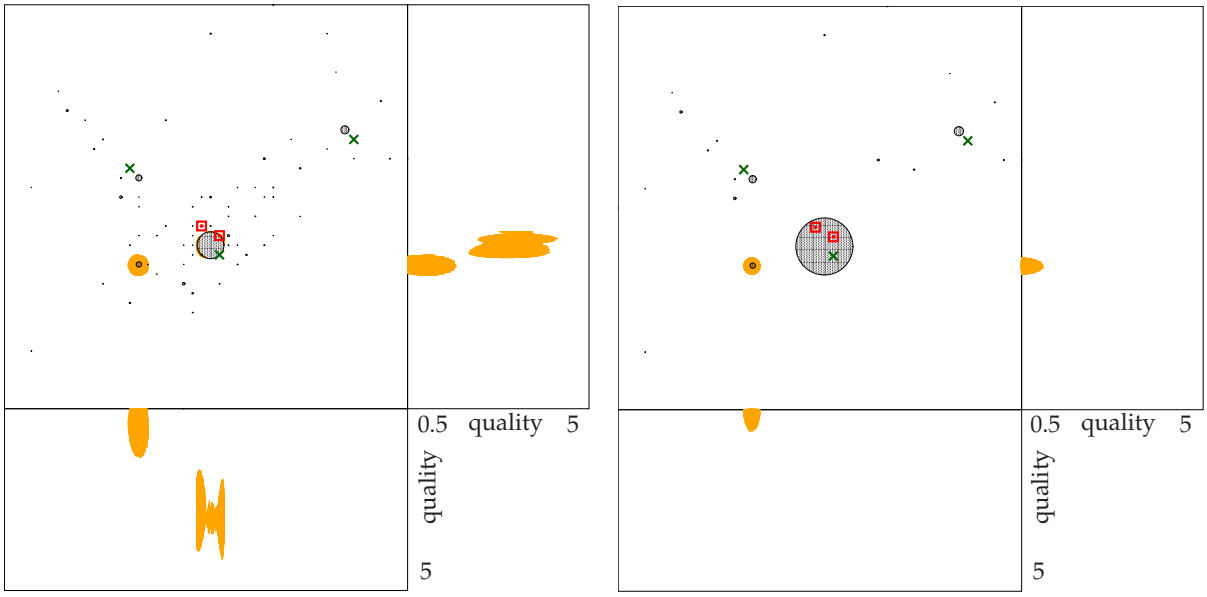


Figure 5.9: Near optimal regions for the Large Chain with unaggregated and aggregated demand.

the two side-panes.

As already observed before, and more clearly visible in the figures, the two last cases are quite interesting since they show that there are two near-optimal regions which are quite far apart. The Small Chain has two almost equivalent options, in Orihuela and in Molina.

Similarly, the Large Chain also has two main options which are almost equivalent, but of quite different type: locating an additional facility around the main city, Murcia, increasing its presence against the two-facility strong competition there, or rather grabbing the market around the fourth city Alcantarilla, since the second and third largest (Orihuela and Molina) are already well served by it. Observe that several positions around Murcia's border are available, needing slightly different but rather upper average quality, while Alcantarilla is already conquered at almost minimum quality.

These last two examples demonstrate the power of global optimization: all (nearly) global optima are found. Any standard local optimization method might have found each of these (and not even necessarily so), but will never have found out there are almost totally equivalent solutions elsewhere. This is extremely useful in order to use alternative location criteria.

The last line in upper and lower parts of Table 5.7 give the total volume of the three-dimensional boxes defining the near-optimal region. The ratio between the third root of this value and the size of the near optimal objective interval gives a more precise impression of the flatness of the objective function around the optimum value, although boundary effects on quality somewhat cloud the issue. Still it is quite clear that the largest chain has the flattest objective, the Small Chain case comes next, and the Newcomer case seems quite steep. These differences are enhanced by aggregation.

The aggregated problems are similar, but with an important difference: now Murcia's forbidden region is much larger, and contains several facilities already. For the Newcomer it is no longer possible to come sufficiently close to Murcia's demand to obtain an important market share. A better option now turns out to be to move to Molina, where lower quality is sufficient to take over appreciable market share from the competitor, who is not close enough to the center to avoid this. For the Small Chain, the aggregated case does not change the optimal solution much, but for the Large Chain the best option is now clearly to aim for Alcantarilla, because its interesting sites around Murcia are now forbidden.

The remainder of the section is devoted to experiment five types of sensitivity analysis, carried out for each of these six basic cases. The following five sections discuss the exact setup in each of these sensitivity studies and describe the results obtained. They have been designed to observe the effects when varying the quality of the existing facilities (section 5.5.2), the distance decay function (section

5.5.3), the shape of the income function of the market share (section 5.5.4), the cost function (section 5.5.5) and the available budget (section 5.5.6). The last section then gives an analysis with respect to data aggregation and chain structure, and terminates with an overview of the global results and the overall conclusions and recommendations reached.

5.5.2 Facility quality spread

The first sensitivity analysis concerns the influence of the quality of each existing facility, as given by the parameters α_j. These were varied in several ways as described by the following scenarios:

Original: In the basic case all existing qualities lie in the interval $[3, 4]$ (as shown in Appendix B). Since the new facility's quality may vary between 0.5 and 5, this means that it may be chosen as having a quality quite different from the existing qualities in the market.

Rescaled: In this scenario, we rescaled all the α_j to the interval $[1, 4.5]$, to make the existing facilities' range of qualities closer to the new facility's possibilities.

S>L: Here we set $\alpha_j = 4$ for all the existing facilities of the Small Chain and $\alpha_j = 2$ for all the existing facilities of the Large Chain. In other words, the Small chain has much higher quality facilities than the Large Chain.

S<L: This is the opposite case, where the largest chain also offers the highest quality: Large Chain's existing facilities are of quality 4 and those of the Small Chain of quality 2.

Closer better: Now all existing facilities close to Murcia (two of the Large Chain and one of the Small Chain) have high quality 4, and those far from Murcia only quality 2.

Closer worse: This final scenario inverses this scheme and gives lowest quality 2 to all existing facilities in the surroundings of the city of Murcia, but high quality 4 to all those far from the main city.

Table 5.8 gives an overview of the comparative results for each of the six considered cases: Newcomer, Small Chain, Large Chain, each both in unaggregated and in aggregated form. For the unaggregated case, the top rows recall the obtained range of (approximate) objective values of the original problem reached within the near-optimal region given by the final list of solution-boxes. All other rows show dissimilarities between the newly found list of boxes enclosing the optimal solution set with another list of boxes, and these in three ways: *3D* considers both quality and location variables together, while *Loc* (resp. *Qual*) considers only the locational aspect (resp. only the quality aspect) by using the projections of all boxes on the 2-dimensional location space (resp. 1-dimensional quality space). In the rows *rescaled*, *S>L*, *S<L*, *closer better* and *closer worse* this is done against the solution of the original problem. The other two rows give the comparison between two specified scenarios: *S>L / S<L* and *closer better/closer worse* (noted by *cl.b./cl.w.*). For the aggregated case, the dissimilarity is always given with the solutions of the corresponding unaggregated case, and also includes the original problem. Recall also the sign convention for dissimilarity in objective value ranges, as explained in Section 5.5.1.3.

As we can see, for the Newcomer and the Small Chain cases the variations in the quality of the existing facilities have no significant influence on the optimal solution. In particular this means that the Small Chain always keeps its two almost equivalent options, a quite remarkable fact.

It is only for the Large Chain case that some changes seem to arise in the optimal solution. In particular, when the two competing facilities close to Murcia have a lower quality (case *closer worse*), it becomes interesting for the Large Chain to add a second facility around Murcia with a rather high quality, because this sufficiently grabs its opponent's market share.

It seems therefore that for small players quality of existing facilities does not strongly influence its decision, which goes towards the large demand areas where they are not yet present, taking over part of the competitor's market. But for chains having many and dispersed facilities (as the Large Chain has) almost any new facility may lead to a lot of cannibalization (loss of market share of the existing facilities of the chain), and therefore the qualities of own and competing facilities should be as accurate as possible if a correct decision is to be made.

Table 5.8: Sensitivity of the results to the quality of the facilities.

	Loc.	Qual.	3D	Obj.	Loc.	Qual.	3D	Obj.	Loc.	Qual.	3D	Obj.
	Newcomer [44.39,45.02]				Small Chain [207.87,210.81]				Large Chain [240.05,243.44]			
rescaled	0.001	0.006	0.007	+8.2	0.042	0.002	0.118	+42.2	0.079	0.066	0.202	−60.3
S>L	0.002	0.008	0.010	+27.9	0.005	0.001	0.019	+82.0	1.274	0.814	2.283	−47.7
S>L / S<L	*0.000*	*0.000*	*0.001*	*+12.8*	*0.005*	*0.017*	*0.079*	*−111.3*	*0.081*	*0.177*	*0.104*	*+77.4*
S<L	0.001	0.003	0.006	−15.0	0.006	0.022	0.080	−193.3	1.939	2.811	3.612	+125.0
closer better	0.000	0.002	0.002	+2.5	0.003	0.000	0.011	+9.9	0.251	0.844	0.264	−2.3
cl.b./cl.w.	*0.078*	*0.002*	*0.064*	*+44.1*	*0.010*	*0.023*	*0.132*	*−37.7*	*1.282*	*0.564*	*1.983*	*+34.9*
closer worse	0.077	0.000	0.062	+41.6	0.003	0.027	0.106	−47.5	3.181	3.203	4.724	+37.2
	Aggr. Newcom. [26.08,26.44]				Aggr. Small Ch. [205.8,208.7]				Aggr. Large Ch. [237.8,241.2]			
original	4.593	4.855	6.795	−36.9	0.011	0.001	0.021	−4.2	0.270	1.350	0.492	−4.5
rescaled	4.551	4.941	6.813	−31.4	0.005	0.001	0.027	+2.0	1.344	2.631	2.714	−7.8
S>L	4.518	4.750	6.669	−53.8	0.015	0.000	0.027	+2.1	3.134	4.986	6.056	−32.9
S<L	4.574	4.834	6.776	−53.5	0.020	0.008	0.090	−14.1	0.013	0.115	0.133	+4.3
closer better	4.577	4.908	6.805	−31.3	0.015	0.001	0.026	−11.5	0.016	0.139	0.226	+2.9
closer worse	4.541	4.632	6.614	−81.9	0.015	0.012	0.058	+3.2	3.220	4.237	5.580	−38.7

Optimal objective values vary much more. Profit may be doubled compared to the original case (see the Newcomer in case *closer worse*) or reduced to more than half (see the Small chain in case $S<L$). As any small chain has to attain a given minimum level of increase in profit in order to survive, qualities of the existing facilities will have to be estimated very well, to be able to have confidence in the estimation of future profit.

From the cases $S>L$ and $S<L$, we can see that the chain that increases the quality of its facilities also has an increase in profit, and vice-versa, when the competitor increases quality, profit drops. These increases are almost perfectly proportional to the number of existing facilities with higher quality, but not the drops: the Small Chain loses relatively much more profit when the Large Chain increases the quality of its three facility, than the Large Chain in the opposite situation.

One may also observe, that when existing facilities close to the global optimum increase (decrease) their quality, the optimal profit of the new facility decreases (increases) accordingly.

To understand the case of rescaled qualities, we should recall the original qualities of the existing facilities. In the case of the Small Chain, the qualities are 3.25 and 4, whereas for the Large Chain they are 3.1, 3.5 and 3.15. When we rescaled [3,4] to [1,4.5] the lower bound 3 goes down to 1, but the upper bound 4 goes up to only 4.5. Thus, the rescaled facilities of the Large Chain have lost more quality, when compared to the facilities of the Small Chain. That's why the Newcomer and the Small Chain increase their profit, whereas the Large Chain experiments a decrease.

5.5.3 Distance decay

In this section we examine the impact of the function $u_i(d_{iz}) = d_{iz}^{\lambda}$ in the solution of the problems. The three examined cases were $\lambda = 1, 2$, and 4. The case $\lambda = 2$ is considered the original problem. Thus, as in the previous table, in the second row we give the interval giving the range of the objective function at the solution boxes offered by the algorithm (when $\lambda = 2$). Next, we have solved the problems setting $\lambda = 1$ and $\lambda = 4$. As before, the dissimilarity between the final box-lists are given in location, quality and full 3D terms, as well as the change in objective. For the three unaggregated cases, comparison is with the $\lambda = 2$ case, while for the aggregated cases comparison is against the corresponding unaggregated case.

When $\lambda = 1$ the changes in the solution are negligible, except for the Large Chain in the aggregated case. The explanation for that case is the following. Notice that when $\lambda = 1$, if the distance between a demand point and a facility is greater than one then the attraction increases (as compared to the attraction when $\lambda = 2$). This makes places far from Murcia more attractive and suitable to be chosen for the location of the new facility, as can be seen in Figure 5.10 (compared to the original case in Figure 5.9). In fact, those places are attractive even with a low value of α. However, in the corresponding

Table 5.9: Results for $g(x) = x^\lambda, \lambda = 1, 2, 4$.

	Loc.	Qual.	3D	Obj.	Loc.	Qual.	3D	Obj.	Loc.	Qual.	3D	Obj.
	Newcomer [44.39,45.02]				Small Chain [207.87,210.81]				Large Chain [240.05,243.44]			
$\lambda = 1$	0.032	0.033	0.088	+5.9	0.006	0.101	0.128	−3.3	0.107	0.019	0.179	−27.7
$\lambda = 4$	0.001	0.142	0.145	+0.4	0.130	0.098	0.293	−7.5	0.368	1.093	0.504	+37.0
	Aggr. Newcomer [26.08,26.44]				Aggr. Small Chain [205.8,208.7]				Aggr. Large Chain [237.8,241.2]			
orig.	4.593	4.855	6.795	−36.9	0.011	0.001	0.021	−4.1	0.270	1.350	0.492	−4.5
$\lambda = 1$	4.698	2.339	5.384	−47.3	0.011	0.002	0.023	+4.7	2.563	0.903	2.710	−23.0
$\lambda = 4$	9.277	6.464	11.328	−22.4	0.006	0.000	0.014	−8.8	0.015	0.063	0.114	+0.3

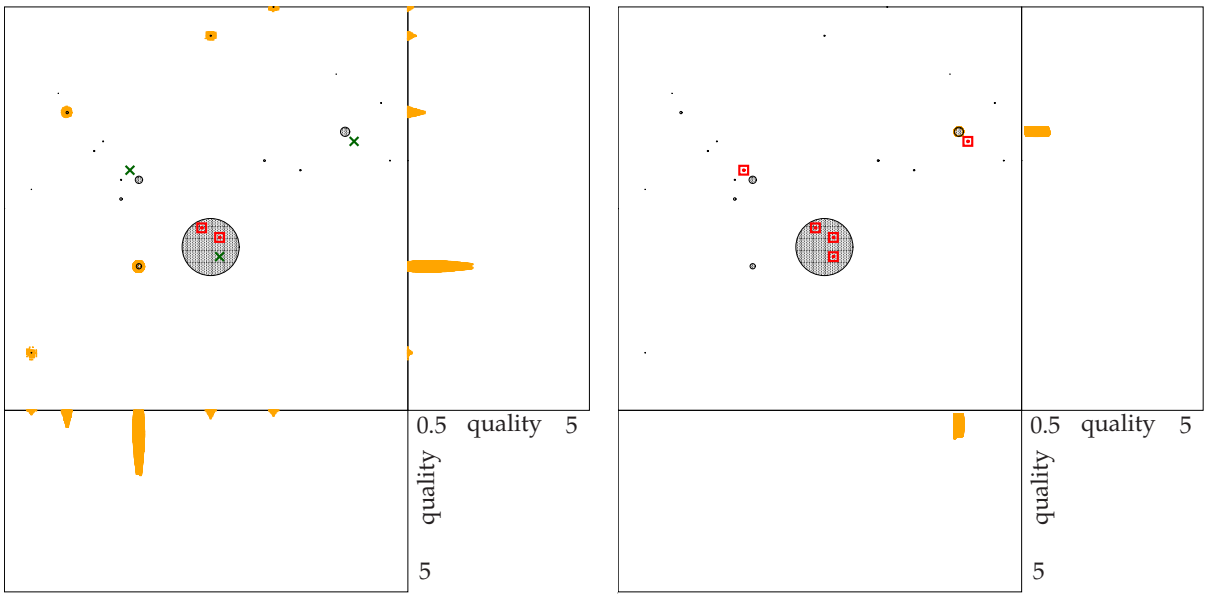


Figure 5.10: Results for Aggr. Large Chain, when $\lambda = 1$, and Aggr. Newcomer when $\lambda = 4$.

unaggregated case this is no longer true. In the original case (see Figure 5.9) the best places are close to Murcia or Alcantarilla. The places around Murcia are at a distance less than 1 from Murcia and when the distance is less than one, then the attraction decreases (as compared to the attraction when $\lambda = 2$) but it is still greater than in places far from Murcia.

When $\lambda = 4$ the solutions are affected a bit more, but still the changes are very small, except for the aggregated Newcomer case (see Figures 5.7 and 5.10), where the optimal solution moves from Molina (in the original aggregated case) to Orihuela (in the aggregated case when $\lambda = 4$), and from having a medium quality to having a low quality. The reason can be that, in the original case, the Newcomer obtained market share from Molina and some part of the market from Murcia, since the attraction of Murcia to the facility in Molina is reduced. With this move the Newcomer obtains a large part of the market from Orihuela, and since there, the quality of the facility is smaller, the costs are reduced and the profit is higher than in Molina.

5.5.4 Market share scaling

We have considered two different types of expected sales function to analyze the sensitivity of the solutions to the way in which the market share is converted into income.

Firstly, the case $F(M) = c \cdot M$ is considered. Initially we have set $c = 12$, which is considered the original case. Then, we have varied c 30% up to 15.6 and 30% down to 8.4. Secondly, in order to include cost increases at decreasing rate with quantity, we consider the case $F(M) = c_1 + c_2 \cdot M^2$,

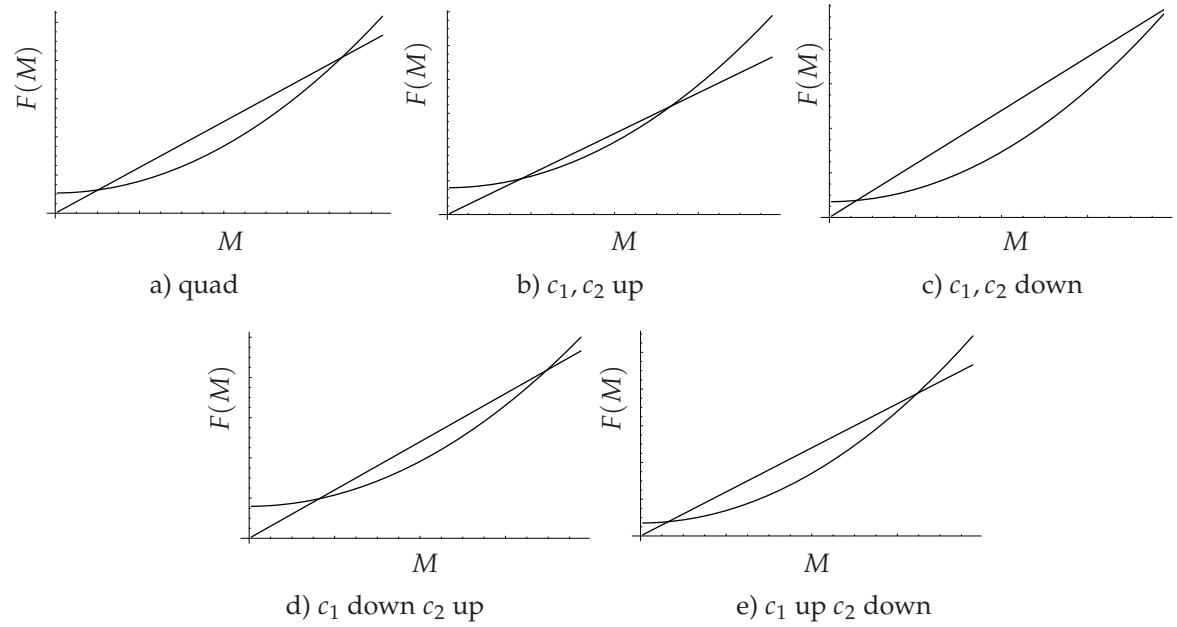


Figure 5.11: Comparing the different distance decay functions tested.

with F convex. Initially we have set the parameters to have similar scale as for the linear case. We used $c_1 = 10.599, c_2 = 1.535$ for the Newcomer, $c_1 = 106.985, c_2 = 0.334$ for the Small Chain, and $c_1 = 131.121, c_2 = 0.273$ for the Large Chain. Then we have varied the parameters c_1 50% up and down and c_2 10% up and down, leading to four possible combinations. Figure 5.11 shows and compares the general shapes of the income functions pictorially.

In the first parts of the table the comparison is made to the original case, and in the second parts taking the quadratic F function as original.

The influence of the F function is small, in fact negligible in most of the cases (specially for the Small Chain case). However, in the few cases where it has some influence (see for instance Large Chain in the

Table 5.10: Results for $F(M) = c \cdot M$, $F(M) = c_1 + c_2 \cdot M^2$.

	Loc.	Qual.	3D	Obj.	Loc.	Qual.	3D	Obj.	Loc.	Qual.	3D	Obj.
	Newcomer [44.39,45.02]				Small Chain [207.87,210.81]				Large Chain [240.05,243.44]			
c ↑	0.000	0.049	0.043	+60.6	0.010	0.048	0.170	+134.7	1.274	0.733	2.168	+166.2
c ↓	0.002	0.664	0.700	−44.5	0.126	0.032	0.183	−102.5	0.267	1.189	0.387	−116.5
quad	0.001	0.152	0.136	+37.9	0.000	0.000	0.003	−1.4	0.001	0.003	0.030	−1.1
$c_1 \uparrow c_2 \uparrow$	0.001	0.149	0.133	+70.3	0.006	0.014	0.066	+128.4	0.024	0.084	0.131	+157.2
$c_1 \downarrow c_2 \uparrow$	0.001	0.148	0.132	+28.5	0.003	0.002	0.008	+83.9	0.063	0.130	0.094	+107.3
$c_1 \uparrow c_2 \downarrow$	0.001	0.154	0.138	+52.7	0.002	0.007	0.015	−49.0	0.071	0.092	0.235	−60.2
$c_1 \downarrow c_2 \downarrow$	0.002	0.155	0.139	+10.9	0.127	0.003	0.135	−93.5	0.268	1.094	0.346	−110.2
	Aggr. Newcom. [26.08,26.44]				Aggr. Small Ch. [205.8,208.7]				Aggr. Large Ch. [237.8,241.2]			
original	4.593	4.855	6.795	−36.9	0.011	0.001	0.021	−4.1	0.270	1.350	0.492	−4.5
c ↑	4.564	3.899	6.125	−67.7	0.009	0.001	0.022	−5.5	3.156	4.460	5.670	−21.5
c ↓	6.950	4.854	8.602	−13.0	0.012	0.001	0.020	−3.3	0.002	0.051	0.095	−2.8
quad	4.614	2.188	5.182	−76.9	0.011	0.001	0.021	−3.8	0.260	1.657	0.562	−3.8
$c_1 \uparrow c_2 \uparrow$	4.602	1.325	4.872	−91.2	0.010	0.001	0.020	−5.8	0.959	2.314	1.808	−13.3
$c_1 \downarrow c_2 \uparrow$	4.614	2.965	5.559	−69.9	0.010	0.001	0.019	−5.9	0.012	0.102	0.127	−10.9
$c_1 \uparrow c_2 \downarrow$	4.612	1.373	4.888	−85.3	0.013	0.001	0.022	−2.4	1.364	2.649	2.646	−2.4
$c_1 \downarrow c_2 \downarrow$	4.629	3.029	5.593	−64.1	0.022	0.001	0.031	−2.5	0.005	0.078	0.109	−0.0

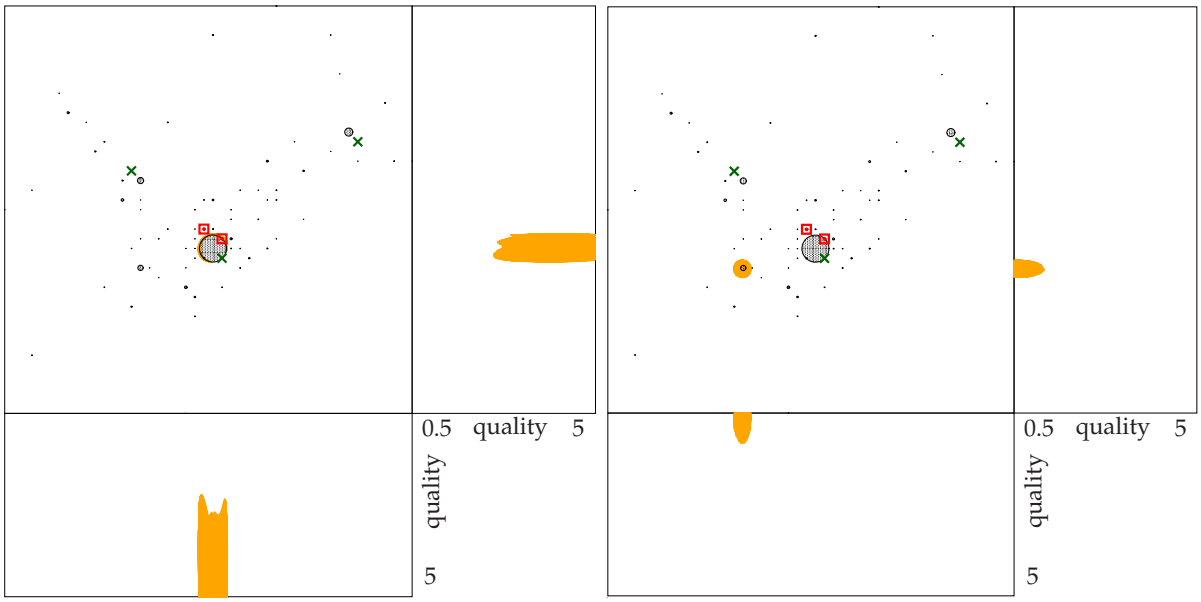


Figure 5.12: Results for the Large Chain when c is increased and decreased.

cases *c up* and *c down*) what is going on is clear: when the parameter c increases, the quality of the new facility also increases, and when c decreases, the quality decreases as well. This is not surprising: since we are going to earn more (less) money from the market share that we capture, we can spend more (less) money to improve the quality of the new facility (our investment implies an increase (decrease) of our profit since we are going to get more (less) market share). This, sometimes, may even lead to a change in the location when the quality of the facility cannot be modified more or when the variation in the quality leads to a high variation in the market share captured. Consider, for instance, the Large Chain case. In the original case (Figure 5.9) the δ-optimal location is in Murcia (with medium quality) or in Alcantarilla (with low quality). However, on the left, in Figure 5.12, the δ-optimal location is only in Murcia with a high quality, meanwhile for the case *c down* (see right side of Figure 5.12) the δ-optimal region is only around Alcantarilla. Thus, we can say that the higher the income obtained form the market share, the higher the quality of the new facility will be (and this, in turn, may provoke a change in the optimal location), and vice versa. Something similar can be said for the quadratic function.

5.5.5 Push force

Next, we have studied the influence of the cost function, by using the modified cost function $G(x, \alpha) = \mu_x G_1(x) + \mu_\alpha G_2(\alpha)$, with $\mu_\alpha = 2 - \mu_x$. The original case was considered to be $\mu_x = \mu_\alpha = 1$, and the test values analyzed were μ_x equal to 0.9, 0.7 ...0.1 and 1.1, 1.3, ..., 1.9. The results are given in Table 5.11, following a structure similar to the previous tables. This time, for each case, first we give the dissimilarity to the original case (as in the previous table), and second, the dissimilarity to the previous setting in the small lines in italics.

One may expect the following phenomenon. When μ_x is very small, quality is quite expensive, so rather low quality settings will be chosen. When μ_x increases, and accordingly in this series of tests μ_α decreases it becomes easier and easier to increase quality, if possible, which will increase sales anyway as long as the site does not change, thus increasing profits. This is exactly what happens in the unaggregated Newcomer's and Small Chain cases, where the best location remains quite stable throughout.

For the Large Chain, in the basic case, the optimal solution is at Murcia and Alcantarilla, but when μ_x decreases, Murcia is no longer a suitable place to locate (since the facility should have a high quality, and it would be too expensive), and vice versa, when μ_x increases, then Alcantarilla is no longer a

Table 5.11: Cost sensitivity results changing parameter μ_x ($\mu_\alpha = 2 - \mu_x$). For each set of parameters the first line shows the distance from the original solutions and the second from the previous setting.

μ_x	Loc.	Qual.	3D	Obj.	Loc.	Qual.	3D	Obj.	Loc.	Qual.	3D	Obj.
1.0	Newcomer [44.39,45.02]				Small Chain [207.87,210.81]				Large Chain [240.05,243.44]			
0.1	0.008	3.363	3.400	−34.2	0.022	0.180	0.507	−4.7	0.265	1.461	0.557	+8.1
0.1-0.3	*0.000*	*0.176*	*0.173*	*−4.3*	*0.001*	*0.002*	*0.008*	*−0.5*	*0.000*	*0.003*	*0.018*	*+1.9*
0.3	0.005	2.583	2.619	−29.9	0.013	0.126	0.345	−4.3	0.264	1.399	0.510	+6.3
0.3-0.5	*0.000*	*0.217*	*0.215*	*−5.8*	*0.001*	*0.004*	*0.013*	*−0.7*	*0.001*	*0.007*	*0.026*	*+1.8*
0.5	0.002	1.682	1.717	−24.1	0.005	0.076	0.185	−3.5	0.264	1.300	0.446	+4.4
0.5-0.7	*0.000*	*0.280*	*0.276*	*−7.7*	*0.001*	*0.007*	*0.017*	*−1.1*	*0.002*	*0.008*	*0.031*	*+1.8*
0.7	0.001	0.612	0.649	−16.4	0.002	0.032	0.070	−2.5	0.261	1.165	0.373	+2.6
0.7-0.9	*0.001*	*0.275*	*0.291*	*−10.2*	*0.001*	*0.012*	*0.025*	*−1.5*	*0.005*	*0.040*	*0.047*	*+1.8*
0.9	0.000	0.043	0.037	−6.2	0.000	0.004	0.010	−1.0	0.051	0.182	0.079	+0.9
1.1	0.000	0.020	0.018	+6.4	0.000	0.004	0.011	+1.1	0.007	0.033	0.043	0.0
1.1-1.3	*0.000*	*0.015*	*0.016*	*+12.7*	*0.001*	*0.036*	*0.075*	*+3.0*	*0.826*	*0.306*	*1.606*	*+8.1*
1.3	0.000	0.075	0.065	+19.1	0.002	0.076	0.150	+4.1	1.277	0.944	2.466	+8.1
1.3-1.5	*0.000*	*0.003*	*0.005*	*+12.7*	*0.006*	*0.101*	*0.204*	*+4.5*	*0.001*	*0.100*	*0.139*	*+12.1*
1.5	0.001	0.104	0.090	+31.9	0.010	0.436	0.773	+8.6	1.279	1.929	3.612	+20.2
1.5-1.7	*0.000*	*0.001*	*0.002*	*+12.7*	*0.013*	*0.131*	*0.194*	*+8.1*	*0.000*	*0.026*	*0.047*	*+12.7*
1.7	0.001	0.119	0.104	+44.6	0.031	1.816	2.170	+16.8	1.278	2.605	4.243	+33.0
1.7-1.9	*0.000*	*0.000*	*0.004*	*+12.7*	*0.000*	*0.392*	*0.268*	*+14.9*	*0.000*	*0.005*	*0.013*	*+12.7*
1.9	0.002	0.126	0.114	+57.4	0.033	4.419	4.519	+31.7	1.278	2.921	4.498	+45.7
	Aggr. Newcom. [26.08,26.44]				Aggr. Small Ch. [205.8,208.7]				Aggr. Large Ch. [237.8,241.2]			
0.1	4.682	3.022	5.672	−8.8	0.013	0.000	0.020	−4.6	0.001	0.014	0.038	−7.4
0.3	4.669	3.454	5.907	−12.7	0.016	0.000	0.025	−5.2	0.001	0.025	0.049	−6.7
0.5	4.651	3.915	6.178	−17.7	0.015	0.001	0.025	−4.9	0.001	0.036	0.067	−6.0
0.7	4.626	4.403	6.488	−24.1	0.014	0.001	0.023	−4.6	0.003	0.055	0.098	−5.4
0.9	4.604	4.832	6.776	−32.2	0.012	0.001	0.021	−4.3	0.009	0.131	0.159	−4.8
original	4.593	4.855	6.795	−36.9	0.011	0.001	0.021	−4.1	0.270	1.350	0.492	−4.5
1.1	4.589	4.620	6.623	−41.5	0.011	0.001	0.020	−4.0	0.756	2.072	1.209	−5.1
1.3	4.584	3.474	5.868	−49.2	0.009	0.001	0.021	−3.9	3.196	4.878	6.054	−14.4
1.5	4.574	1.482	4.939	−54.2	0.010	0.005	0.049	−3.9	3.237	5.727	6.886	−27.7
1.7	4.568	0.117	4.579	−54.0	0.016	0.004	0.030	−5.2	3.247	4.617	6.043	−41.6
1.9	4.560	0.013	4.566	−51.7	0.012	0.046	0.058	−6.9	1.844	0.893	2.517	−49.6

suitable place (in Murcia we can locate with a higher quality, and obtain more profit).

For the Newcomer, the quality starts around 2.5 and steadily increases to approximate the maximum value of 5, which is almost reached in the basic case, after which quality can almost not increase anymore, resulting in a corresponding linear increase in profit, since both site and quality remain fixed.

With the aggregated data the best site for the Newcomer is systematically at Alcantarilla, and the quality steadily increases, while profits change much less than in the unaggregated case.

For the Small Chain location remains stable too, but now quality starts at around minimum value, and very slowly increases towards the still low 1.4 in the basic case. This slow increase goes on but accelerates towards the end, not quite reaching the maximum of 5, even when quality is lowest. A similar behavior is seen for the aggregated data.

It may also happen that profits increase and also that it then becomes better at some point to migrate to another area. This can be seen in the Large Chain's case (see Figures 5.13 and 5.14). For low μ_x the best site is Molina with minimum quality, rising slowly to average values when $\mu_x = 0.9$, and with slightly augmenting profits. At this point a very small near-optimal area appears close to Murcia, which takes over the global optimum around $\mu_x = 1.1$ with similar average quality, and quickly moves to the maximum quality. After this move to Murcia, profits are increasing again.

With the aggregated data the same shift in best site for the Large Chain happens, but at a much later stage.

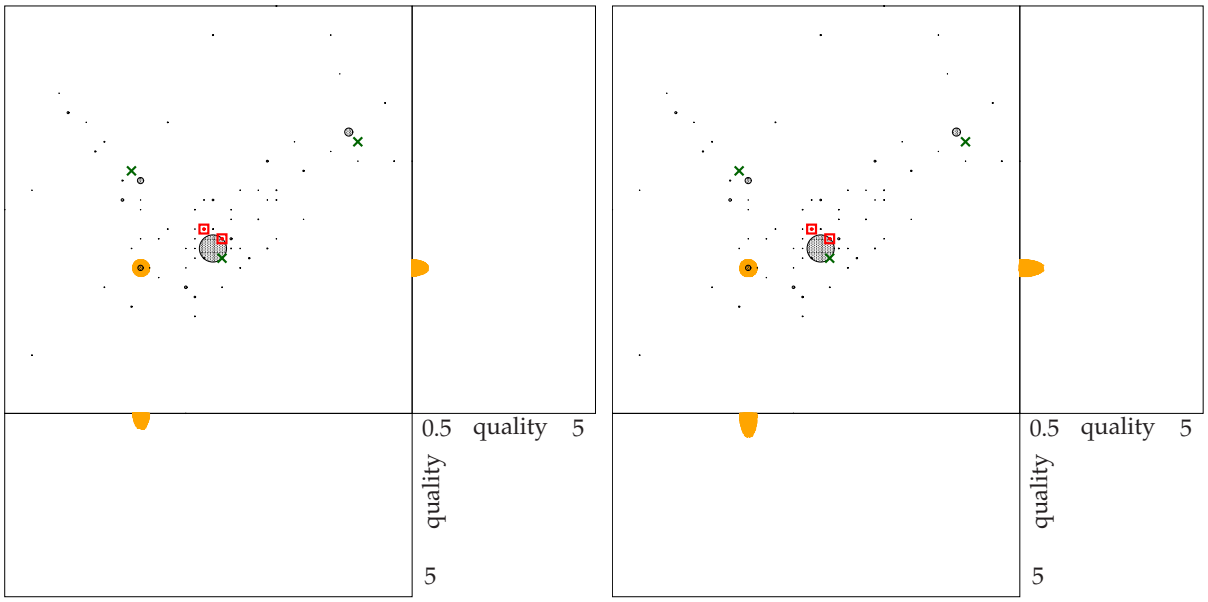


Figure 5.13: The results for Large Chain for $\mu_x = 0.1, 0.5$, respectively.

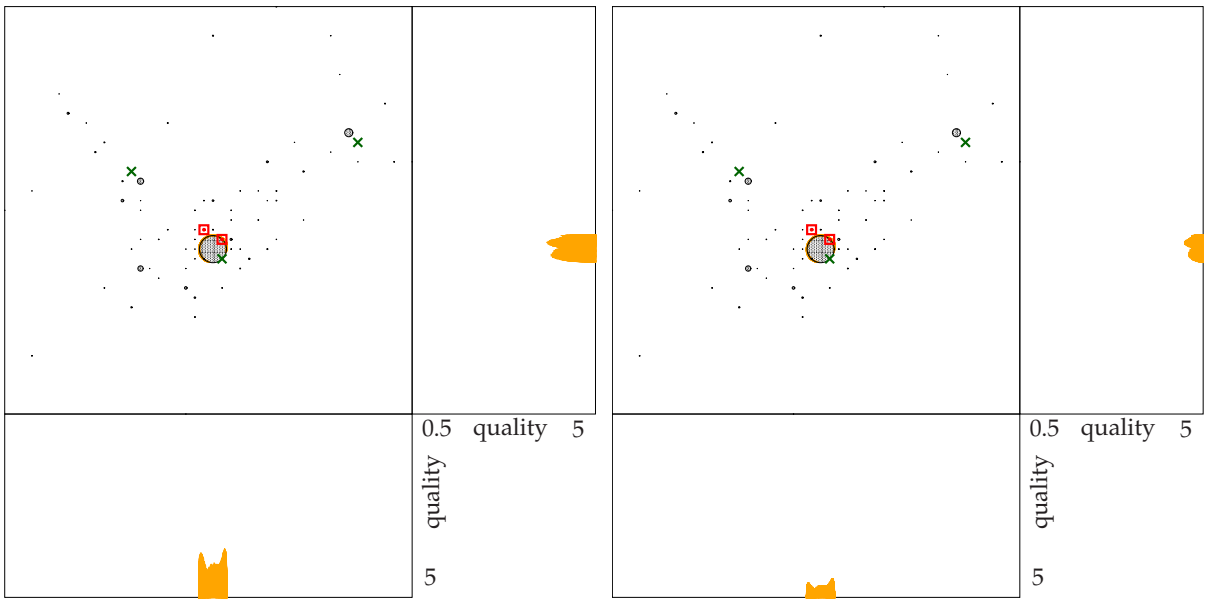


Figure 5.14: The results for Large Chain for $\mu_x = 1.5, 1.9$, respectively.

We also studied the influence of the cost function in an overall sense: using the cost function $G(x, \alpha) = \mu\big(G_1(x) + G_2(\alpha)\big)$. The original case corresponds to $\mu = 1$, and we tested both lower and higher costs relative to income from sales, as shown in Table 5.12.

When costs are negligible compared to income from sales, one can easily choose high quality and expensive sites to increase sales, resulting in quite high profits. When μ increases this will have the opposite effect, usually lowering quality, thus sales, and possibly a move to another cheaper site, all resulting in a clear loss in profit.

Table 5.12: Cost sensitivity results changing parameter μ in $G(x,\alpha) = \mu\big(G_1(x) + G_2(\alpha)\big)$.

	Loc.	Qual.	3D	Obj.	Loc.	Qual.	3D	Obj.	Loc.	Qual.	3D	Obj.
	Newcomer [44.39,45.02]				Small Chain [207.87,210.81]				Large Chain [240.05,243.44]			
0.5	0.002	0.090	0.077	+56.4	0.044	0.847	1.379	+19.9	1.279	1.875	3.568	+44.8
0.5-0.7	*0.000*	*0.002*	*0.003*	*−22.5*	*0.009*	*0.142*	*0.201*	*−10.6*	*0.001*	*0.098*	*0.129*	*−21.9*
0.7	0.001	0.065	0.055	+33.8	0.019	0.115	0.345	+9.6	1.277	0.941	2.454	+22.9
0.7-0.9	*0.000*	*0.012*	*0.012*	*−22.5*	*0.011*	*0.062*	*0.149*	*−6.5*	*0.511*	*0.187*	*1.107*	*−18.0*
0.9	0.000	0.017	0.015	+11.3	0.002	0.004	0.021	+2.9	0.031	0.039	0.102	+4.9
1.1	0.000	0.039	0.034	−11.1	0.125	0.004	0.135	−2.7	0.257	1.021	0.300	−2.1
1.1-1.3	*0.001*	*0.315*	*0.334*	*−20.0*	*0.001*	*0.011*	*0.023*	*−4.9*	*0.005*	*0.015*	*0.051*	*−4.2*
1.3	0.002	0.664	0.701	−31.1	0.126	0.032	0.183	−7.6	0.267	1.189	0.387	−6.4
1.3-1.5	*0.001*	*0.354*	*0.353*	*−17.5*	*0.000*	*0.008*	*0.015*	*−4.5*	*0.003*	*0.008*	*0.033*	*−4.2*
1.5	0.008	1.774	1.807	−48.6	0.127	0.083	0.264	−12.0	0.272	1.321	0.467	−10.6
	Aggr. Newcom. [26.08,26.44]				Aggr. Small Ch. [205.8,208.7]				Aggr. Large Ch. [237.8,241.2]			
0.5	4.529	1.391	4.884	−65.6	0.013	0.010	0.044	−6.1	3.153	5.613	6.715	−40.2
0.7	4.556	3.417	5.819	−56.1	0.009	0.002	0.026	−4.2	3.160	4.852	5.966	−21.9
0.9	4.579	4.590	6.596	−43.8	0.012	0.001	0.021	−4.2	1.097	2.229	1.797	−7.6
original	4.593	4.855	6.795	−36.9	0.011	0.001	0.021	−4.1	0.270	1.350	0.492	−4.5
1.1	4.615	4.853	6.804	−29.9	0.124	0.001	0.134	−4.2	0.008	0.096	0.156	−4.2
1.3	6.958	4.872	8.614	−16.9	0.012	0.001	0.020	−4.3	0.002	0.051	0.095	−3.7
1.5	9.335	4.474	10.389	−4.3	0.012	0.001	0.021	−4.5	0.001	0.032	0.068	−3.1

5.5.6 Budget restrictions

In our last study we check the influence of a limited budget in the optimal solution. In particular, we have added a constraint of the form $G(\boldsymbol{x},\alpha) \leq B$, where B is the available budget. We first solved the different problems without any constraint, obtained their solutions (x^*,α^*), and determined the budget B^* needed to locate a facility at x^* with a quality α^*, by setting $B^* = G(x^*,\alpha^*)$. This was considered the original case. Then, we solved the problems by reducing the available budget B, setting it equal to 95%, 90%, 85%, ..., 55% of B^*. As in the previous table, for each case, first we give the dissimilarity to the original case, and second, the dissimilarity to the previous setting. The value of B^* for the aggregated case was set equal to the value of B^* for the unaggregated case. The results are summarized in Table 5.13.

When the budget decreases slightly, the only effect observed is that quality is lowered in order to stay within the budget, resulting in a corresponding loss in sales, thus in profit. But at some point the budget becomes so small that lowering quality does not suffice anymore and one also has to move to cheaper places. This typically means moving away from the population centers (for instance, for the Newcomer and Small Chain, the optimal location remains at the same place, we can only see a reduction in the quality of the new facility as B decreases, although in the Small chain, from 100% to 95% we lose one of the two optimal locations). This effect may become so great that the facility is pushed towards the boundaries of the feasible region. An example may be seen in Figure 5.15, where we can see how flat the function is around these edges also (the inner part is now an unfeasible region due to the added constraint). Also, whereas for the Newcomer and Small Chain cases the objective decreases at an increasing rate as B decreases (this proves that the objective function for those problems is not flat), for the large chain the opposite happens (this proves that the objective becomes flatter as the budget decreases).

The strange looking value in the Large Chain aggregated list at 90%, is due to the fact that in the corresponding unaggregated case to which it is compared to calculate the dissimilarity, the large boundary region did not appear yet, but it does when aggregating.

Overall the profit losses due to budget restrictions may be considered reasonable. The Small Chain does not lose too much, compared to the Newcomer and Large Chain cases, when B decreases. This is because from the beginning it chooses a not too expensive location (Orihuela) and a not too high quality. Thus, when the budget decreases it only has to decrease the quality a bit and keep capturing

Table 5.13: Results on budget restriction.

	Loc.	Qual.	3D	Obj.	Loc.	Qual.	3D	Obj.	Loc.	Qual.	3D	Obj.
	Newcomer [44.39,45.02]				Small Chain [207.87,210.81]				Large Chain [240.05,243.44]			
95%	0.000	0.081	0.098	−0.391	0.127	0.192	0.443	−0.054	0.490	1.711	0.905	−4.494
	0.000	*0.179*	*0.196*	*−0.791*	*0.000*	*0.037*	*0.014*	*−0.154*	*0.403*	*0.017*	*0.411*	*−6.142*
90%	0.000	0.400	0.436	−1.181	0.127	0.261	0.517	−0.208	0.986	1.551	1.318	−10.637
	0.000	*0.267*	*0.281*	*−1.212*	*0.000*	*0.023*	*0.017*	*−0.272*	*4.400*	*0.014*	*3.820*	*−3.692*
85%	0.001	0.876	0.904	−2.394	0.128	0.311	0.584	−0.480	4.914	1.445	4.732	−14.329
	0.000	*0.329*	*0.342*	*−1.709*	*0.000*	*0.023*	*0.020*	*−0.445*	*0.240*	*0.000*	*0.189*	*−1.153*
80%	0.001	1.365	1.391	−4.103	0.128	0.391	0.667	−0.925	5.037	1.326	5.199	−15.482
	0.000	*0.405*	*0.420*	*−2.263*	*0.000*	*0.027*	*0.026*	*−0.673*	*0.022*	*0.000*	*0.055*	*−0.485*
75%	0.002	1.906	1.932	−6.366	0.129	0.482	0.761	−1.598	5.155	1.327	5.381	−15.966
	0.000	*0.450*	*0.460*	*−2.900*	*0.000*	*0.032*	*0.032*	*−0.922*	*0.031*	*0.000*	*0.076*	*−0.979*
70%	0.003	2.458	2.484	−9.265	0.129	0.586	0.868	−2.520	5.271	1.420	5.592	−16.945
	0.000	*0.485*	*0.494*	*−3.623*	*0.000*	*0.036*	*0.040*	*−1.281*	*0.005*	*0.000*	*0.037*	*0.002*
65%	0.005	3.023	3.050	−12.888	0.130	0.713	1.000	−3.800	5.525	1.439	5.826	−16.947
	0.000	*0.520*	*0.530*	*−4.448*	*0.000*	*0.041*	*0.048*	*−1.747*	*0.007*	*0.000*	*0.041*	*+0.002*
60%	0.008	3.619	3.646	−17.336	0.131	0.881	1.173	−5.548	5.851	1.497	6.158	−16.945
	0.000	*0.560*	*0.567*	*−5.415*	*0.001*	*0.045*	*0.056*	*−2.313*	*0.016*	*0.000*	*0.055*	*−0.001*
55%	0.012	4.247	4.273	−22.751	0.132	1.049	1.339	−7.861	6.281	1.526	6.610	−16.946
	Aggr. Newcom. [26.08,26.44]				Aggr. Small Ch. [205.8,208.7]				Aggr. Large Ch. [237.8,241.2]			
95%	4.604	5.348	7.156	−36.596	0.012	0.002	0.019	−4.164	0.061	0.000	0.084	−6.946
90%	4.618	5.222	7.066	−36.051	0.012	0.006	0.021	−4.214	4.893	0.001	4.891	−6.186
85%	4.632	5.048	6.939	−35.240	0.013	0.002	0.022	−4.312	0.438	0.000	0.380	−3.761
80%	4.642	4.852	6.789	−34.154	0.013	0.002	0.022	−4.442	0.056	0.000	0.091	−2.610
75%	4.656	4.618	6.640	−32.759	0.013	0.002	0.023	−4.582	0.021	0.000	0.062	−2.125
70%	7.082	3.802	8.363	−30.538	0.014	0.002	0.023	−4.773	0.022	0.002	0.085	−1.147
65%	9.314	3.482	9.963	−27.097	0.015	0.003	0.025	−4.979	0.025	0.002	0.086	−1.166
60%	9.326	3.155	9.859	−23.117	0.015	0.002	0.025	−5.194	0.037	0.001	0.091	−1.462
55%	9.343	2.811	9.778	−18.557	0.017	0.002	0.025	−5.480	0.095	0.000	0.152	−1.462

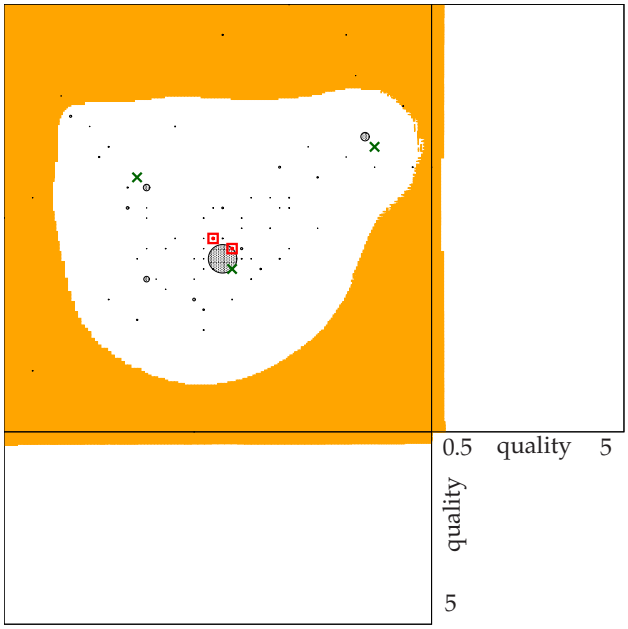


Figure 5.15: Results for Large Chain when the budget is restricted by 55%.

most of its original market share. However the Newcomer has to decrease its quality more, so it loses more, and the Large Chain has to even change its location (since the reduction in quality is not enough to compensate the reduction of the budget).

5.5.7 Evaluation

Chain structure. The chain structure of the locating player clearly plays an important role in the shape of the objective function. One easily observes that the Large Chain has the flattest function, and thus larger near-optimal regions, often showing more than one almost equivalent option, the best choice among all which may vary with relatively small parameter changes.

A newcomer in the market is working in a considerably hillier environment, and usually only a single best option appears, remaining quite insensitive to data changes. In a sense this may be seen as the newcomer having the easiest decision to take. But we can only infer this using the global optimization method proposed here. A local optimizer might end up quite elsewhere, without knowing that considerably better options exist.

Demand aggregation. Looking through the different tables in previous sections one may compare the unaggregated cases with the aggregated ones, thus getting an idea of the effects of data aggregation on results. These are not always the same, although there seems to be an almost systematic tendency to slightly underestimate profits after data aggregation. However, this effect seems to be more due to the increased sizes of the forbidden regions around each demand point than to aggregation.

Particularly a very small player entering the market will need the more precise description of the market in order to be able to detect the most promising *holes* in the market where it may benefit most.

Recommendations. The main conclusion that we can derive from the computational studies is that it is very important to decide the location and the quality simultaneously because each one influences the other considerably. Our experiments have shown that the use of rigorous methods is necessary in order to know with guarantee that the found optimum is global. We have also seen that the near-optimal region is a great help to the decision maker in choosing the best location and quality and may even lead to suggesting to locate more than one facility.

From the extensive analysis one can conclude that for real applications the estimation of the data and the parameters plays an important role, therefore it has to be done carefully. For cases when the owner's chain is already present, wrongly approximated data can cause wrong estimation not only in profit, but in location and design as well. Thus, apart from accurate approximation of the data and parameters, the decision maker has to look not only at the globally optimal location and design, but also at the near-optimal regions. For larger chains it is recommended to compute a larger near-optimal region, and select from the possible new insights according to the chain's requirements and facilities.

5.6 Conclusions

In this chapter, a new realistic model has been introduced for single facility planar competitive location problems. In this model, the quality parameter of the new facility is included as a variable so that the location and quality of the new facility are determined simultaneously.

To solve the model, we have applied the interval B&B method using all the available discarding tests. We have shown the usefulness of every test, which altogether reduced the computational time considerably, by 80% in average. The interval B&B method was also compared to the Weiszfeld-like local search algorithm, and proved that although the interval B&B method is somewhat slower, the achieved results are better.

We have examined the best inclusion function by the empirical convergence speed, and showed that the Baumann form is the best choice for this class of problems, even if the algorithm uses the discarding tests available for each inclusion function.

As a last study, we have conducted an exhaustive sensitivity analysis on the parameters of the model. One of the main conclusions we have found is that for problems where the number of existing chain owned facilities is higher the solution is more sensitive to changes in the parameters.

Chapter 6

The two facilities problem

In real world applications, dynamically growing companies sometimes decide to build not only one new facility at a time, but several, or one facility just after the other. Frequently, the location and design of the new facilities are decided in a sequential way. That is, the company first decides on the location and quality for the first new facility, later for the second facility, and so on. However, it might be better to decide the locations and qualities of all the new facilities simultaneously.

In order to know how good the myopic approach is compared to the approach of determining the optimal location and quality of all new facilities simultaneously, one needs global optimization techniques. The traditional techniques that only obtain a proposal of sites apparently close to a local optimum solution, but without a clear indication of the actual degree of optimality, cannot be used to make an adequate and reliable comparison of the different strategies. Since the complexity of determining, simultaneously, the optimal location and quality of more than one new facility with exactness is enormous, we only consider the case of two new facilities.

The main purpose of this chapter is to compare the different strategies when locating more than one facility (contrary to the comparison of different algorithms for a same model made in Chapter 5). We thereby also demonstrate, however, that the technique of global optimization used in Chapter 5 for the single facility case, may also be used, after suitable adaptation, for two-facility location problems, having a much heavier computational effort [144].

The chapter is organized as follows. The location model is presented in Section 6.1. Different approaches to cope with it are presented in Section 6.2. The solution methods employed to carry out the approaches are described in Section 6.3. In Section 6.4 we present a comparison of the solutions obtained by the different strategies on some computational tests, and the difference in performance of the algorithm. The chapter ends in Section 6.5 with some conclusions and directions for future research.

6.1 The model

A chain wants to locate two new facilities in a given area of the plane, where there already exist m facilities offering the same goods or product. The first k of those m facilities belong to the chain, $0 \leq k < m$. The demand, supposed to be inelastic, is concentrated at n demand points, whose locations and buying power are known. The location and quality of the existing facilities is also known.

Basically we use the same notation as in Chapter 5, but because of the two new facilities, their variables now have an index, l, to distinguish between them: z_l denotes the location of the new facility l, $z_l = (z_{l1}, z_{l2}), l = 1, 2$, and similarly, α_l denotes the quality of the new facility l. Thus, the variables of the problem are $nf = (nf_1, nf_2)$, where $nf_l = (z_l, \alpha_l), l = 1, 2$. Now the distance between demand point i and the new facility nf_l is denoted by d_{iz_l}, and so the attraction that demand point i feels for the new facility nf_l is $\dfrac{\gamma_i \alpha_l}{u_i(d_{iz_l})}$.

We consider again that the patronizing behavior of customers is probabilistic. In [27], T. Drezner presents a related multifacility location model, in which the qualities of the new facilities are assumed

to be known; profit maximization then reduces to market share maximization. Our model generalizes her work by deciding on both the locations and on the qualities of the two facilities to be located.

With the previous notation and assumptions, the total market share attracted by the chain is

$$M(nf) = \sum_{i=1}^{n} \omega_i \frac{\frac{\gamma_i\alpha_1}{u_i(d_{iz_1})} + \frac{\gamma_i\alpha_2}{u_i(d_{iz_2})} + \sum_{j=1}^{k} \frac{\alpha_{ij}}{u_i(d_{ij})}}{\frac{\gamma_i\alpha_1}{u_i(d_{iz_1})} + \frac{\gamma_i\alpha_2}{u_i(d_{iz_2})} + \sum_{j=1}^{m} \frac{\alpha_{ij}}{u_i(d_{ij})}}.$$

6.2 The different approaches

6.2.1 The general simultaneous location problem

The general problem, denoted by $\text{Sim}_{\alpha_1 \neq \alpha_2}$, is

$$\begin{array}{lll} \max & \Pi(nf) = F(M(nf)) - G(nf_1) - G(nf_2) & \\ \text{s.t.} & d_{iz_l} \geq d_i^{\min}, & i = 1 \ldots, n,\ l = 1, 2 \\ & \alpha_l \in [q_{\min}, q_{\max}], & l = 1, 2 \\ & z_l \in R \subset \mathbb{R}^2, & l = 1, 2 \end{array} \tag{6.1}$$

where $\Pi(nf_1, nf_2)$ is as in the previous chapter, the profit obtained by the chain, equal to the total income minus the costs. Again $F(M)$ is a strictly increasing (twice) differentiable function which transforms the market share M into expected sales and $G(nf_l)$ is a (twice) differentiable function which gives the operating cost of a facility located at z_l with quality α_l (see Section 5.1).

6.2.2 The restricted simultaneous location problem

In the restricted version $\text{Sim}_{\alpha_1 = \alpha_2}$ both new facilities are obliged to have the same quality. The formulation of the problem is the same as the general form (6.1), except for the additional constraint $\alpha_1 = \alpha_2$.

6.2.3 The sequential location problem

The sequential location problem, called Seq, is a simple two-step process. First the best location and quality for a first additional facility is obtained solving the single facility problem (5.1), which is a simplified expression of (6.1), without all variables z_2 and α_2 and terms involving them. This yields an optimal location z_1^* and quality α_1^*. Then the best location and quality for a second additional facility are determined, considering the first facility fixed at the optimal solution found in the first step. In other words, the second step is solving the single facility problem obtained from (6.1) when z_1 and α_1 are fixed to z_1^* and α_1^*, respectively.

6.3 The solution methods

For *more than one new facility*, to our knowledge, no exact algorithm has been proposed in literature. Heuristics do exist which use a cyclic search in which only one location is optimized independently, holding all other locations fixed. Then, the solutions they find are only guaranteed to be local optima and do not necessarily obtain the best locations for the new facilities. Some cyclic ascent algorithms for finding new facility locations for fixed qualities are provided in [27, 30, 100]. In a recent work by J.L. Redondo et al. [129] different heuristics were proposed for solving (6.1).

All three approaches presents rather general models (only second order differentiability is supposed) and can be specialized for many real situations. Of course, their ability to model general situations comes at the expense of the difficulty of their solution: the problems are all neither concave

nor convex. Even finding a global optimum for the corresponding single facility location and design problem (which has 3 variables) is a difficult task (see Chapter 5). The sequential approach consists of two successive single facility location and design steps, so it is of the same difficulty as the single facility problem. The general simultaneous location problem, however, has 6 variables, so it is much harder to solve. The restricted simultaneous location problem has one fewer variable, so should be somewhat more tractable.

An interval B&B algorithm (Algorithm 1.1) was used to solve both the simultaneous and the sequential approaches. Since our algorithms are written for minimization problems, for the sake of simplicity the objective function in (6.1) is converted to minimization by negating it. For the sequential approach we have used the algorithm Basic+All in Section 5.3.2. In the simultaneous case the centered form is used together with the natural interval extension as inclusion functions, while the used discarding tests were the following: feasibility test, cut-off test (Section 1.4), monotonicity test for feasible and undetermined boxes (Section 2.4.1), simplified pruning method (Section 2.2.4), projected one-dimensional interval Newton method (Section 2.4.2), and projected one-dimensional non-convexity test. In addition to them, a lexicographical order test was also used to discard symmetric solutions. It works as follows. We order the solutions lexicographically, i.e., we require that $z_{11} \leq z_{21}$ and if $z_{11} = z_{22}$ then $z_{12} \leq z_{22}$. Thus, if for a given box $(z_{11}, z_{12}, \boldsymbol{\alpha}_1, z_{21}, z_{22}, \boldsymbol{\alpha}_2)$ we have that $\underline{z_{11}} > \overline{z_{21}}$, then the box can be discarded.

The interval B&B method for our 6-dimensional problem generates too many boxes in the working list, and the computer usually runs out of memory. This difficulty can be solved with an easy trick. We have divided the search space at the four location variables into 5 smaller intervals, thus generating 625 subproblems. As we require the two facilities to have lexicographical order, there are cases which should not be taken into account. For instance, we should not solve the subproblem with initial box $([8, 10], [0, 2], [0.5, 5], [2, 4], [6, 8], [0.5, 5])$, since $\underline{z_{11}} > \overline{z_{21}}$. Thus, we can discard those initial boxes, therefore we have to solve only 325 subproblems.

To have a good initial solution, we first do a local search in the whole search region. Then we start with the subproblem containing the local solution. Solving the subproblems one by one, the upper bound on the best solution is always updated and used for the next subproblem. The solution boxes are stored for every subproblem, if any, and after solving all the subproblems a final cut-off test is done with the best upper bound, to guarantee that only the best boxes remain in the solution list. As a local search procedure, we have used a cyclic algorithm similar to Algorithm 2 described in [27], but using the Weiszfeld-like algorithm introduced in Section 5.2.1 (which takes the quality variable and the constraints of the problem into account). It works as follows. The new variable values for one facility at a time are found while holding the other facility fixed in its current values. Once this is accomplished, the optimized facility is fixed and the other facility's variables are optimized. One iteration consists of finding the new values, one at a time, for both facilities.

6.4 Computational experiments

In this section we present some computational results to compare the solutions obtained with the three strategies $\text{Sim}_{\alpha_1 \neq \alpha_2}$, $\text{Sim}_{\alpha_1 = \alpha_2}$, and Seq, on a set of instances.

All the computational results have been obtained under Linux on a Pentium IV computer with 3.0 GHz CPU and 2 GB memory. The algorithms have been implemented in C++. We used the interval arithmetic implementation of the PROFIL/BIAS library [91], and the automatic differentiation of the C++ Toolbox library [69].

6.4.1 The test problems

In order to have an overall view of the type of solutions offered by the three strategies, we have generated different types of settings, varying the number of demand points ($n = 50$ or 100), the number of existing facilities ($m = 2$, 5, or 10) and the number of those facilities belonging to the chain ($k = 0$ or 1 for $m = 2$, $k = 0$, 1 or 2 for $m = 5$ and $k = 0$, 2 or 4 for $m = 10$). For every type of setting one instance was generated, by randomly choosing the parameters of the model uniformly within the following intervals:

- $ef_j, dp_i \in [0, 10]^2$,
- $\omega_i \in [1, 10]$,
- $\gamma_i \in [0.75, 1.25]$,
- $\alpha_{ij} \in [0.5, 5]$,
- $G(nf_l) = \sum_{i=1}^{n} \Phi_i(d_{iz_l}) + G_2(\alpha_l)$ where
  - $\Phi_i(d_{iz_l}) = \omega_i \frac{1}{(d_{iz_l})^{\phi_{i0}+\phi_{i1}}}$ with $\phi_{i0} = \phi_0 = 2, \phi_{i1} \in [0.5, 2]$,
  - $G_2(\alpha_l) = e^{\frac{\alpha_l}{\rho_0}+\rho_1} - e^{\rho_1}$ with $\rho_0 \in [7, 9], \rho_1 \in [4, 4.5]$,
- $c \in [1, 2]$, the parameter for $F(M) = c \cdot M$,
- $b_1, b_2 \in [1, 2]$, parameters for $d_{iz_l} = \sqrt{b_1(z_{l1} - dp_{i1})^2 + b_2(z_{l2} - dp_{i2})^2}$ (see [47]).
- $\lambda_i = \lambda = 2$, the parameter for $u_i(d_{iz}) = (d_{iz})^{\lambda_i}$.

We have also considered four quasi-real instances, adapted from a case study about location of supermarkets in the Region of Murcia, in the South-East of Spain (see Section 5.5.1.1). The complete data for all instances used in the current study can be obtained from [145]. The search space for every problem was

$$z_l \in [0, 10]^2, \quad \alpha_l \in [0.5, 5], \quad l = 1, 2.$$

The three algorithms were run with equal tolerances, $\varepsilon_1 = \varepsilon_2 = 10^{-2}$.

In Figure 6.1 we show the differences in the solutions obtained with the different approaches in two problems. The gray circles are the forbidden regions around the demand points, their radii grow with the buying power of the demand points. The existing facilities are denoted by $\boxdot$ and $\times$ for the competing and for the own chain facilities, respectively. The solution is shown by orange or blue regions projected in the location space and also in the quality dimension.

The left picture shows the solutions for a problem with $n = 100$, $m = 10$, and $k = 4$ with the $\text{Sim}_{\alpha_1 \neq \alpha_2}$ (orange) and Seq (blue) approaches. Their quality can be read in the upper part of the figure with the same x value. The picture on the right shows the solutions with the two simultaneous approaches, $\text{Sim}_{\alpha_1 \neq \alpha_2}$ (orange) and $\text{Sim}_{\alpha_1 = \alpha_2}$ (blue), for a case with $n = 50$, $m = 2$, $k = 0$. It is clear from the examples that the different strategies may lead to very different solutions.

6.4.2 Comparison of objective values

First, we have compared the objective values obtained by $\text{Sim}_{\alpha_1 \neq \alpha_2}$ with those obtained by $\text{Sim}_{\alpha_1 = \alpha_2}$ (see the columns under $\text{Sim}_{\alpha_1 = \alpha_2}$ in Table 1). When the optimal solution of a problem with $\text{Sim}_{\alpha_1 \neq \alpha_2}$ was such that $\alpha_1 = \alpha_2$, we did not run $\text{Sim}_{\alpha_1 = \alpha_2}$, because the result would be the same (the corresponding values in Table 1 are empty). Second, we have compared the objective values obtained by $\text{Sim}_{\alpha_1 \neq \alpha_2}$ with the ones obtained by the sequential approach, Seq (see the columns under Seq in Table 1).

We observe that the differences in the objective function value between the solutions obtained by $\text{Sim}_{\alpha_1 \neq \alpha_2}$ and $\text{Sim}_{\alpha_1 = \alpha_2}$ are rather small, always less than 0.75%. Although in half of the problems the solution with $\text{Sim}_{\alpha_1 \neq \alpha_2}$ was such that $\alpha_1 = \alpha_2$, this is a bit misleading. In those cases, both α_1 and α_2 reach the upper bound $q_{\max} = 5$. This is due to the way of generating the parameters of the model: choosing the parameters in other intervals would have led to solutions with $\alpha_1 \neq \alpha_2$.

On the other hand, the differences between $\text{Sim}_{\alpha_1 \neq \alpha_2}$ and Seq are usually much greater, with relative differences as high as 4.94%. In 4 of the 16 problems the relative differences were greater than 1%. This proves the importance of selecting the correct strategy when selecting more than one competing facility.

The running times corresponding to each of the three approaches are also shown in Table 6.1. The interval B&B algorithm described in Section 6.3 used to solve the simultaneous approaches is rather time consuming, as compared to the CPU time needed to solve the two 3-dimensional problems of the sequential approach with the Basic+All interval B&B algorithm described in Section 5.3.2. For this type of problems, however, what matters is obtaining the optimal solutions, and not how long it takes.

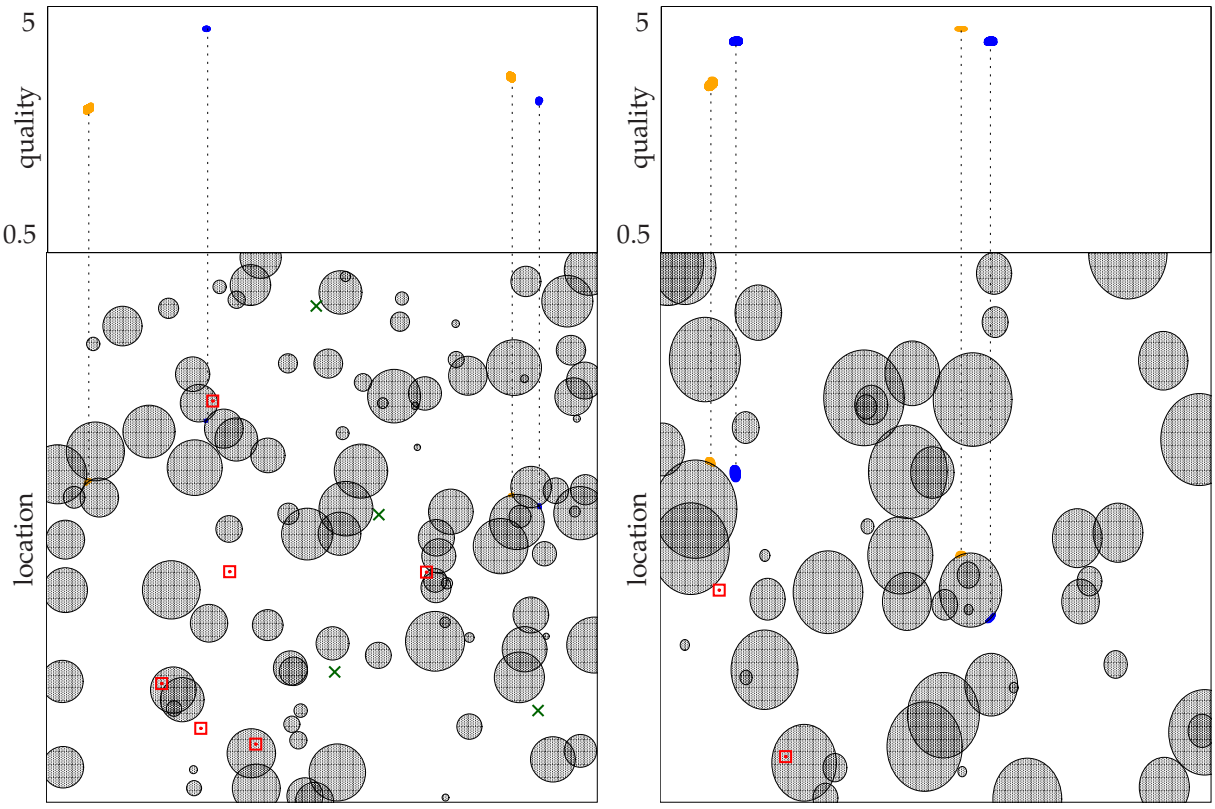


Figure 6.1: Examples of the differences of the solutions in quality and location.

6.4.3 Comparison of the solutions

We have used the Hausdorff metric to measure the difference between the locations and the qualities obtained by the three different strategies. The difference between two sets of solutions A and B is measured as follows. The solution sets are given by lists of boxes, i.e. $A = \cup_{x \in \mathcal{L}_A} X$, and $B = \cup_{y \in \mathcal{L}_B} Y$. To measure the distance between the lists of boxes we used the measure

$$\text{diff}(\mathcal{L}_A, \mathcal{L}_B) = \max \left\{ \max_{x \in \mathcal{L}_A} \min_{y \in \mathcal{L}_B} \Delta(x, y), \max_{Y \in \mathcal{L}_B} \min_{x \in \mathcal{L}_A} \Delta(y, x) \right\}, \tag{6.2}$$

where $\Delta(x, y) = \max_{x \in x} \min_{y \in y} \|x - y\|$ and $\| \cdot \|$ is the Euclidean norm. We computed two types of distances, the 4-dimensional distance in location, where only the locational variables are taken into account, and the two dimensional distance in quality, computing distances for the quality boxes. To avoid the problem of symmetrical solutions we checked the distance changing the order of the facilities so as to obtain the minimal difference in location and in quality.

In Table 6.2 we can see the differences between the solutions for the three approaches. We took the $\text{Sim}_{\alpha_1 \neq \alpha_2}$ approach for the base solutions, and computed the distances from those. In the table, column diff(loc) shows the distance in location, while column diff(qual), the quality distance. We can see that although in general the distances are not large, in some cases they can be more than 20% of the search space. Again, the differences between the approximate solutions provided by $\text{Sim}_{\alpha_1 \neq \alpha_2}$ and the other two approaches are bigger for Seq, for both the locational and the quality variables. But even the differences with $\text{Sim}_{\alpha_1 = \alpha_2}$ are considerable in some cases (see for instance cases (50,2,0) or (21,5,2)). Also notice that the cases with higher differences in the solutions do not necessarily correspond to those for which the difference in objective was higher. This can be explained by the high nonlinearity of the problem.

Table 6.1: Differences of the objective values and running times for the three approaches. Column *abs* shows the absolute, while column *rel* the relative differences in percentages of the objective values.

Problem			Objective difference				Required Time		
			$\text{Sim}_{\alpha_1=\alpha_2}$		Seq		$\text{Sim}_{\alpha_1\neq\alpha_2}$	$\text{Sim}_{\alpha_1=\alpha_2}$	Seq
n	m	k	abs	rel	abs.	rel.	hours	hours	sec
21*	5	2	0.488	0.158	0.267	0.087	15:36:48	9:53:15	6.9
21*	5	3	0.166	0.038	0.057	0.013	26:11:16	16:09:26	7.6
50	2	0	0.515	0.249	1.496	0.723	30:21:03	19:40:17	9.0
50	2	1	0.124	0.057	10.828	4.941	21:58:39	6:11:21	9.8
50	5	0	0.036	0.024	1.639	1.097	11:04:06	5:43:59	4.4
50	5	2	0.033	0.013	1.093	0.431	17:20:11	6:06:03	11.3
50	10	0			0.507	0.570	19:03:11		11.7
50	10	4	1.264	0.706	0.488	0.273	126:56:37	82:31:15	27.1
71*	5	0			1.299	0.434	23:37:49		10.3
71*	5	2			0.483	0.070	53:13:02		18.0
100	2	0			11.207	2.525	6:11:50		6.0
100	2	1			2.379	0.440	28:48:30		11.7
100	5	0			1.885	0.726	12:31:27		10.1
100	5	2			6.103	0.993	5:44:29		6.4
100	10	0			2.865	1.094	13:07:45		10.7
100	10	4	0.264	0.066	2.151	0.538	140:58:39	116:49:52	41.4

* These are real-life problems from the region of Murcia, Spain.

Table 6.2: Comparison of results in solution space and in quality. The differences are computed with respect to $\text{Sim}_{\alpha_1\neq\alpha_2}$.

Problem			$\text{Sim}_{\alpha_1=\alpha_2}$		Sequential	
n	m	k	diff(loc)	diff(qual)	diff(loc)	diff(qual)
21*	5	2	0.391	0.703	0.394	0.516
21*	5	3	0.110	0.298	0.330	0.195
50	2	0	1.378	1.063	0.637	0.271
50	2	1	0.100	0.397	1.990	1.943
50	5	0	0.031	0.337	0.631	0.625
50	5	2	0.031	0.328	0.396	0.806
50	10	0			0.427	0.227
50	10	4	0.324	1.103	0.206	0.642
71*	5	0			0.258	0.284
71*	5	2			0.342	0.000
100	2	0			2.394	0.000
100	2	1			1.534	0.391
100	5	0			0.517	0.000
100	5	2			2.301	0.000
100	10	0			0.702	0.000
100	10	4	0.282	0.592	2.523	1.144

* These are real-life problems from the region of Murcia, Spain.

6.5 Conclusions

In this chapter, we have presented the two-facilities location problem for profit maximization. Market share is estimated by using Huff-like models and costs associated to both the quality and the location of the new facilities are taken into account. These two factors are the variables of the problem, which also includes constraints to delimit the search region. The model includes most of the features found in literature for this kind of models. The precise mathematical form of these features is extremely open (only second order differentiability is supposed), producing a very general setting, so as to be adaptable to many practical situations. This shows the power of the general global optimization methodology used.

However, this chapter mainly demonstrates that methodology is useful to consider more theoretical comparisons between several management strategies, leading to insights directly useful in practice. We have compared simultaneous decision strategies for the two facilities, either with independent or equal quality, with a sequential decision strategy. Our computational tests have shown that there is not much difference in profit for the optimal solution when we consider that the new facilities are of the same quality as compared to the optimal solution when different qualities are allowed, although this might have been a side-effect of our testing set-up. However, the difference in profit is in some cases remarkable when we consider the optimal solutions generated by the simultaneous and the sequential approaches. On the other hand, we have found that differences in location and quality are often outstanding between the optimal solutions generated by any of the two pairs of approaches (simultaneous versus simultaneous with equal qualities, and simultaneous versus sequential). Consequently, there are many local optimal solutions with a profit very close to the maximum profit which might be quite different in location and/or quality.

It is clear that the simultaneous approach finds better solutions than the sequential approach, but at an extremely high computational cost. As future research we propose studying new procedures for the multifacility problem when different qualities for the new facilities are allowed.

Chapter 7

The cannibalization problem

In many competitive location problems, a chain wishes to open a set of new facilities to provide goods to the customers of a given geographical area where other competing chains offering the same goods are already present. As a result of the entrance of a new facility, the existing facilities of the chain may lose part of their market share to the new facility, a detrimental phenomenon called *cannibalization*. The purpose of this chapter is to study the planar location problem for a single facility when the chain seeks to maximize profit while keeping cannibalization as low as possible [50]. This secondary objective should be taken into account when the chain has to compensate those owners of the chain's facilities that lose market share due to the entrance of the new facility, or when there is an *inner* competition between the facilities of the same chain in addition to the *outer* competition against facilities of other chains. This situation has been studied in the franchise system when the set of candidate locations is finite (see [21, 65]). To our knowledge, this is the first time that the problem is studied in planar location space. A related problem to the one mentioned above is studied in [122], where customers patronize the facility which attracts them most and only decision on location is considered.

This chapter is organized as follows: after introducing the model with the two conflicting objectives in Section 7.1, we present some computational results in Section 7.2 using the Lexicographical-like method introduced in Section 3.2. The discarding tests of the Lexicographical-like method are also compared. In Section 7.3, some economic analyses are carried out, and finally Section 7.4 closes the chapter with some conclusions.

7.1 The model

As in our earlier models, a chain wants to locate a single new facility in a given area of the plane, where there already exist m facilities offering the same goods or product. The first k ($k \geq 1$) of those m ($m > k$) facilities belong to the chain, while the other facilities (numbered $k+1, \ldots, m$) are competitors. The demand, supposed to be inelastic, is concentrated at n demand points, whose locations dp_i and buying power ω_i are known. The location ef_j and quality of the existing facilities is also known. We follow the notation used in Chapter 5.

We consider that the patronizing behavior of customers is probabilistic. Under these assumptions *after* entry of the new facility of quality α at site z, the market share of the new facility will be

$$msa(nf) = \sum_{i=1}^{n} \omega_i \frac{\dfrac{\gamma_i \alpha}{u_i(d_{iz})}}{\dfrac{\gamma_i \alpha}{u_i(d_{iz})} + \displaystyle\sum_{j=1}^{m} \frac{\alpha_{ij}}{u_i(d_{ij})}} \tag{7.1}$$

and that of existing facility ℓ

$$msa(\ell, nf) = \sum_{i=1}^{n} \omega_i \frac{\frac{\alpha_{i\ell}}{u_i(d_{i\ell})}}{\frac{\gamma_i \alpha}{u_i(d_{iz})} + \sum_{j=1}^{m} \frac{\alpha_{ij}}{u_i(d_{ij})}} \tag{7.2}$$

The total market share attracted by the chain is then given by

$$M(nf) = msa(nf) + \sum_{\ell=1}^{k} msa(\ell, nf).$$

7.1.1 First objective: maximization of the profit obtained by the chain

The first objective of our problem is the maximization of the profit obtained by the chain, to be understood as the difference between the revenues obtained from the captured market share minus the operating costs of the new facility (see Chapter 5). Note that we may disregard the operating costs of the existing facilities of the own chain, since these are considered to be constant.

Therefore we consider profit as given by the expression

$$\Pi(nf) = F(M(nf)) - G(nf)$$

where F and G are as in Section 5.1.

In order to be able to exploit a broader spectrum of tools of interval analysis, we also assume both F and G to be differentiable. Weakening this assumption to continuity, the methodology remains valid, except that the δ-monotonicity and δ-pruning tests proposed in Section 3.2.1 will not be available.

In our computational studies we made the following more detailed choices.

- Function F is linear, $F(M(nf)) = c \cdot M(nf)$, where c is the income per unit of goods sold.
- Function G is separable, i.e. $G(nf) = G_1(z) + G_2(\alpha)$, with $G_1(z) = \sum_{i=1}^{n} \Phi_i(d_{iz})$, with $\Phi_i(d_{iz}) = \omega_i / ((d_{iz})^{\phi_{i0}} + \phi_{i1})$, $\phi_{i0}, \phi_{i1} > 0$ and $G_2(\alpha) = e^{\frac{\alpha}{\rho_0} + \rho_1} - e^{\rho_1}$, with $\rho_0 > 0$ and $\rho_1 \in \mathbb{R}$ given values.

A more detailed explanation of these functions, as well as other possible expressions for $G(nf)$ can be found in Section 5.1. Of course, other functions might be more suitable depending on the real problem considered, and for each real application the most appropriate F and G functions should be ascertained.

7.1.2 Second objective: minimization of the cannibalization suffered by the existing chain-owned facilities

Obviously, the maximization of profit is the main objective for the owner of the chain. This considers the competition coming from outside the chain. However, he/she should also take into account that some form of competition also exists within the chain, as expressed by the so-called *cannibalization*. When the new facility enters the market, the existing chain-owned facilities might see a decrease of their own market share. At first glance this can be seen as just a simple transfer of the market share captured by the chain from some of its facilities to another, without side effects. However, if a given existing chain-owned facility loses much of its market share then it may become no longer profitable to the chain. The chain is then forced to either close it down or reduce its operating costs (usually dismissing employees and/or reducing the quality it offers, both things are bad for the image of the chain) or to increase its quality (which means more cost for the chain, and also that the other chain-owned facilities will then suffer the cannibalization from it). To avoid this, we will include as second objective the minimization of the cannibalization suffered by the existing chain-owned facilities.

The market share captured by the existing facility $\ell \in \{1, \ldots, k\}$, *before* the new facility enters the market is given by

$$msb(\ell) = \sum_{i=1}^{n} \omega_i \frac{\frac{\alpha_{i\ell}}{u_i(d_{i\ell})}}{\sum_{j=1}^{m} \frac{\alpha_{ij}}{u_i(d_{ij})}}$$

which is easily seen to be strictly greater than its market share *after* entry given by (7.2). The cannibalization suffered by ℓ is the difference between these market shares

$$Can(\ell, nf) = msb(\ell) - msa(\ell, nf)$$

and our second objective is to minimize the sum of the cannibalizations suffered by all existing members of the chain,

$$\min Can(nf) = \sum_{\ell=1}^{k} Can(\ell, nf).$$

On the other hand, notice that minimizing the sum of individual cannibalizations is equivalent to maximizing the market share captured by all the facilities of the chain *except* the new one after the entrance of the new facility, i.e.,

$$\min Can(nf) \Leftrightarrow \max \sum_{\ell=1}^{k} msa(\ell, nf).$$

7.1.3 The problem

The problem to be solved is then

$$\begin{array}{ll} \max & \Pi(nf) \\ \min & Can(nf) \\ \text{s.t.} & d_{iz} \geq d_i^{\min},\ i = 1, \ldots, n \\ & \alpha \in [q_{\min}, q_{\max}] \\ & z \in R \subset \mathbb{R}^2 \end{array} \tag{7.3}$$

where $d_i^{\min} > 0, q_{\min} > 0$, $q_{\max}$ and R are as in Section 5.1.

This is a biobjective optimization problem which may be expected to be quite hard to solve, since each objective alone already leads to a hard global optimization problem, the first of which has been studied in Chapter 5.

We may write problem (7.3) in the standard form of a biobjective minimization problem:

$$\begin{array}{ll} \min & f_1(x) \\ \min & f_2(x) \\ & x \in S \subset \mathbb{R}^3 \end{array} \tag{7.4}$$

where $x = nf$, $f_1(x) = -\Pi(nf)$, $f_2(x) = Can(nf)$ and S denotes the feasible set of problem (7.3).

7.2 Computational experiments

In this problem one can easily observe that the two objectives do not have the same importance. In fact, the first objective, i.e. the maximization of the profit, is much more important than the second one, i.e. the minimization of the cannibalization.

A number of computational experiments were devoted to study the usefulness of the discarding tests used in the implementation of the δ-lexicographic method presented in Section 3.2.1. All the

computational results presented in this chapter have been obtained on a PC with an Intel Pentium IV 2.33GHz processor and with 1 Gbyte RAM running under Linux operating system. For the implementation we have used the interval arithmetic modules provided in the PROFIL/BIAS library [91] and the automatic differentiation of the C++ toolbox library described in [69]. To this end, we have generated different types of problems, varying the number of demand points ($n = 20$, 40 or 60), the number of existing facilities ($m = 2$, 3, 4 or 5) and the number of those facilities belonging to the chain ($k = 1, \ldots, m-1$). For every type of setting, 10 instances were generated by randomly choosing the parameters uniformly within the following intervals:

- $ef_j, dp_i \in [0, 10]^2$,
- $\omega_i \in [1, 10]$,
- $\gamma_i \in [0.75, 1.25]$,
- $\alpha_{ij} \in [0.5, 5]$,
- $\phi_{i0} = \phi_0 = 2, \phi_{i1} \in [0.5, 2]$, the parameters for $\Phi_i(d_{iz}) = \omega_i \frac{1}{(d_{iz})^{\phi_{i0}+\phi_{i1}}}$ (recall that we have used $G_1(z) = \sum_{i=1}^{n} \Phi_i(d_{iz})$)
- $\rho_0 \in [7, 9], \rho_1 \in [4, 4.5]$, the parameters for $G_2(\alpha) = e^{\frac{\alpha}{\rho_0}+\rho_1} - e^{\rho_1}$,
- $c \in [1, 2]$, the parameter for $F(M(nf)) = c \cdot M(nf)$,
- $b_1, b_2 \in [1, 2]$, parameters for $d_{iz} = \sqrt{b_1(x - dp_{i1})^2 + b_2(y - dp_{i2})^2}$ (see [47])
- $\lambda_i = \lambda = 2$, the parameter for $u_i(d_{iz}) = (d_{iz})^{\lambda_i}$.

The search space for every problem was $z \in [0, 10]^2$, $\alpha \in [0.5, 5]$. We always chose $\delta = 0.1$ (this parameter determines the ratio of the first objective which can be traded off by the second objective, see Section 3.2) and the tolerances used in the algorithm were $\varepsilon = 0.05$ and $\eta = 0.005$.

Based on Algorithm 3.1 we have studied the following five algorithms:

interval GBSSS: in this algorithm we only use the feasibility and the δ-cut-off tests; the pruning test is substituted by a simple bisection perpendicular to the direction of maximum width,

basic: we use the tests in the previous algorithm and also the cut-off test due to the second objective function,

basic + δ-mono: we use the tests in basic and the δ-monotonicity test,

basic + δ-prun: we use the tests in basic and the δ-pruning test,

basic + all: in which we use all the discarding tests, that is, we use as main procedure the one given in Algorithm 3.1.

The results obtained are summarized in Table 7.1. For the *interval GBSSS* algorithm we give the CPU time in seconds (CPU), the number of boxes processed by the algorithm (NB) and the maximum number of boxes stored in the lists ($\mathcal{L}_W^{min} + \mathcal{L}_S^{min} + \mathcal{L}_W^{\delta} + \mathcal{L}_S^{\delta}$) at any time during the execution of the algorithm (ML). The given values are averages over the 10 random instances in each setting (n, m, k). For the rest of the methods the above figures are given in percentages with respect to the corresponding values of the *interval GBSSS* algorithm.

As we can see, even the *interval GBSSS* is able to solve all the problems in a reasonable amount of time. However, the new discarding tests make the process much more efficient. *Basic* reduces 57%, in average, the CPU time needed for solving the problems as compared to *interval GBSSS*. Analogously, the reduction in ML is also very important, going from 34% for problems with $n = 60$ demand points to 66% for problems with the $n = 20$. As for NB, the reduction is even higher, in average 67%. Thus, the cut-off test due to the second objective function is very effective.

The δ-monotonicity test is also very efficient at discarding boxes. Furthermore, the greater the n, the more effective the test. For instance, for $n = 20$, *basic+δ-mono* reduces the CPU time by 29% as compared to *basic*, and for $n = 60$ the reduction is 36%. Something similar happens with NB. For ML the corresponding reductions are, in average, around 24%.

Table 7.1: Comparison of the discarding tests of the δ-lexicographic method

	interval GBSSS			basic			basic+δ-mono			basic+δ-prun			basic+all		
n,m,k	CPU (sec)	NB #	ML #	CPU %	NB %	ML %	CPU %	NB %	ML %	CPU %	NB %	ML %	CPU %	NB %	ML %
20,2,1	61.9	135908	34249	21.3	16.0	35.0	14.5	11.1	25.8	22.6	16.5	21.8	15.2	11.5	16.6
20,3,1	75.4	157678	36473	29.0	23.3	40.3	20.8	16.6	31.4	24.9	19.1	25.2	17.1	13.1	20.1
20,3,2	50.1	115652	27106	30.5	24.5	29.9	19.0	15.0	22.1	36.3	23.8	19.9	21.2	14.9	15.1
20,4,1	95.6	200756	51485	25.4	21.5	29.3	21.0	15.9	24.2	30.4	22.7	18.4	21.2	17.0	14.8
20,4,2	73.3	160133	37378	34.2	31.2	37.7	23.1	20.0	29.6	41.5	30.3	28.6	28.0	19.2	19.2
20,4,3	40.2	90894	20470	21.4	17.0	35.7	14.7	11.0	25.6	26.9	18.1	23.0	16.9	11.6	17.6
20 Av:	66.1	143503	34527	27.4	22.7	34.4	19.5	15.4	26.6	30.6	22.1	22.7	20.3	15.0	17.1
40,2,1	73.6	80656	17282	34.9	27.2	72.0	22.1	18.3	52.2	32.2	24.4	47.1	20.9	16.4	35.8
40,3,1	117.7	129356	28541	35.2	28.8	58.4	24.7	19.9	43.1	29.9	22.8	36.0	22.9	16.5	27.9
40,3,2	66.1	73952	16218	40.5	31.9	74.0	29.5	25.0	59.1	37.7	29.6	46.0	29.7	21.7	39.2
40,4,1	173.5	190930	43246	39.4	34.1	50.9	25.9	22.1	38.5	26.8	19.1	26.5	15.4	12.2	20.6
40,4,2	88.5	90765	18211	39.8	33.7	81.1	27.2	24.4	64.7	30.6	25.3	46.1	20.7	17.5	40.2
40,4,3	54.4	62597	14351	51.3	38.4	82.0	36.2	28.0	61.2	45.4	35.4	48.3	34.0	25.0	37.0
40,5,1	201.2	221074	46203	41.7	35.9	54.7	28.9	24.9	45.2	26.1	19.8	28.5	16.8	13.5	23.7
40,5,2	175.7	195631	42938	45.1	39.0	58.3	31.5	27.4	47.3	35.2	28.7	33.4	23.6	19.3	26.9
40,5,3	97.0	106217	20903	52.5	45.6	81.7	37.9	34.1	65.7	33.1	26.1	43.2	24.6	19.1	36.3
40,5,4	35.1	38001	8012	39.3	32.1	84.9	31.1	23.8	66.4	30.8	23.6	45.4	22.5	17.5	39.0
40 Av:	120.3	132131	28434	41.9	35.2	64.0	29.1	24.8	50.1	31.3	24.3	36.3	21.4	16.8	29.4
60,2,1	90.8	67979	14729	43.2	32.2	75.2	25.1	19.9	49.2	30.3	20.8	43.9	19.3	12.9	31.3
60,3,1	215.4	145660	32887	42.7	33.4	59.8	25.0	20.3	41.6	25.7	18.9	30.7	14.5	11.9	22.9
60,3,2	151.9	100733	22792	53.0	41.0	83.1	31.2	27.5	57.0	35.9	25.3	41.4	20.8	16.7	31.6
60,4,1	267.4	181941	38982	39.8	30.2	55.7	21.5	18.1	41.4	24.3	16.9	30.1	14.1	10.5	22.8
60,4,2	182.5	132017	29114	43.4	34.2	70.5	27.7	22.4	50.0	29.6	21.0	37.8	18.7	13.7	29.0
60,4,3	104.0	72737	16159	45.6	37.5	86.7	33.5	27.6	66.0	35.3	28.6	48.8	23.7	20.3	40.1
60,5,1	560.7	373574	85472	45.2	37.2	48.7	29.7	24.8	38.1	23.6	18.3	22.3	16.8	12.3	18.3
60,5,2	266.4	174606	36499	51.7	44.0	74.8	35.3	29.5	59.1	27.0	21.4	37.3	18.1	14.5	30.1
60,5,3	211.4	142811	30135	52.3	43.1	88.6	36.5	30.8	68.9	29.5	23.9	43.0	20.4	16.4	35.5
60,5,4	100.6	72307	16356	47.8	35.5	84.1	33.0	25.8	64.7	32.6	24.6	45.3	24.3	18.0	36.9
60 Av:	227.8	154673	34085	46.1	37.1	65.7	29.5	24.5	49.0	27.3	20.6	33.3	17.6	13.6	26.2
GloAv:	135.8	132394	29609	42.7	32.8	56.7	28.2	22.3	43.3	28.9	22.3	31.4	19.1	15.1	24.9

The δ-pruning test shows a similar behavior: the greater the n, the more effective. For problems with $n = 20$ demand points *basic+δ-prun* is a bit slower than *basic*; however, it slightly reduces NB, and the reduction of ML attains a considerable 34%. For $n = 40$ *basic+δ-prun* already attains a reduction of 25% in the CPU time, and for $n = 60$ it goes to 61%. Considering all the problems, the reduction in both CPU time and NB as compared to *basic* is 32% and in ML it is even greater, around 45%.

As for *basic+all*, for $n = 20$, its CPU time is 26% better than *basic*, but slightly worse than *basic+δ-mono* (due to the use of the δ-pruning test); however, it slightly reduces NB as compared to *basic+δ-mono*, and the reduction in ML attains a considerable 35%. In average, the reduction in time, is 55%, as compared to *basic*, 32% as compared to *basic+δ-mono* and 34% as compared to *basic+δ-prun*. Notice that the reduction in CPU time obtained with *basic+all* as compared to *basic* is not the sum of the reductions obtained with *basic+δ-mono* and *basic+δ-prun*, but it does attain more than half of that sum. This means that the δ-monotonicity test and the δ-pruning test discard different types of boxes, they use the information of the inclusion of the gradient differently: that is why they are so efficient when used together. Also, the reduction in ML is very important.

As a summary, we can say that all the tests analyzed are very efficient, and reduce the CPU time, NB and ML considerably. The reduction in ML is particularly interesting when working with computers with low available RAM memory.

It is worth noting that for all the algorithms, for given (n, m) values, the algorithm becomes faster as k increases. This is because for a higher k it is better to locate the new facility close to the facilities of the competitor, both for maximizing the profit and, to minimize the cannibalization: the two objectives are less conflicting. Finally, notice that the time increases linearly with n.

7.3 Economic analysis of the model

In this section, we carry out an analysis of some of the economic implications of the model proposed, using an earlier version of the quasi-real example of Section 5.5. The only differences are that in the example used in this section the buying power is mapped into the interval $[1, 10]$, and the parameter $\rho_1 = 4.5$. The problems have been solved with the *basic+all* implementation of the δ-lexicographic method with tolerances $\varepsilon = 0.001$ and $\eta = 0.0005$.

7.3.1 Influence of the threshold δ

Using the proposed methodology, we have studied what happens when either the Large Chain or the Small Chain wants to locate a single new facility, using four different values of δ. The results are given in Table 7.2.

The value between brackets shown after each chain's existing facility is the original market share, before entry of the new facility. The columns marked *Market share* indicate the market shares of each of the chain's facilities after entry of the new facility. The next column gives the profit obtained by the chain. The column marked *Range cannib.* refers to the obtained range for $Can(R^1_\delta)$, i.e. the range of cannibalization over the δ-optimal region on the profit. The lower bound of this interval is just the solution of (P^δ_2), i.e. $\min Can(nf)$ subject to $nf \in R^1_\delta$; the upper bound has been estimated by solving the problem

$$\begin{array}{ll} \max & Can(nf) \\ \text{s.t.} & nf \in R^1_\delta \end{array}$$

The first row ($\delta = 0$) corresponds to optimization of the chain's profit as a single objective. Even in this case *Range cannib.* is a non-degenerate interval, since $\eta > 0$. Observe the remarkable size of the ranges in that column, which means that there are feasible points offering similar values for the first objective but quite different ones for the second. This phenomenon is the main motivation for the δ-lexicographic approach. As δ increases the corresponding δ-optimal regions grow, so the corresponding ranges in cannibalization must also grow. In particular the decrease in lower bounds clearly shows that cannibalization may be considerably reduced by relatively small allowances in the chain's profit, and that the trade-off is quite nonlinear: by a 1% sacrifice in profit for the Large Chain, cannibalization may be reduced by 9% (100*(9.06-8.24)/9.06 = 9.05%), which may be further reduced up to 25% (resp. 39%) by losing no more than 5% (resp. 10%) on the optimal profit. When the Small Chain locates, these cannibalization reductions are even larger: 11%, 28% and 63%, respectively.

Table 7.2: Sensitivity wrt δ of the cannibalization and market share of the chain's facility.

Large Chain locates

δ-optimality for (P_1)	Market share				$\Pi(nf)$	Range cannib.
	L1 (17.15)	L2 (22.31)	L3 (13.85)	New (–)		
0.00	14.68	16.86	12.71	20.38	673.62	[9.06, 9.51]
0.01	14.93	17.29	12.84	18.60	672.53	[8.24, 10.28]
0.05	15.12	18.39	12.99	15.85	668.05	[6.82, 11.16]
0.10	15.47	19.14	13.15	13.19	662.47	[5.55, 11.30]

Small Chain locates

δ-optimality for (P_1)	Market share			$\Pi(nf)$	Range cannib.
	S1 (25.13)	S2 (20.64)	New (–)		
0.00	23.02	18.45	13.96	598.30	[4.29, 4.59]
0.01	23.25	18.68	12.77	597.34	[3.83, 5.04]
0.05	23.61	19.04	10.83	593.36	[3.11, 5.71]
0.10	24.22	19.94	8.16	588.44	[1.60, 6.20]

As δ increases the market share captured by the existing facilities also increases (although at a different rate for each facility), since cannibalization is reduced. Accordingly, the market share captured by the new facility reduces (as well as the chain's profit). Furthermore, the facility capturing the largest market share before the entry of the new facility is not necessarily the one who loses the greatest part (see the case when the Small Chain locates and $\delta = 0$).

Although the Small Chain has only two facilities, they each generate a considerably larger market share, totaling a comparable total market share only slightly below that of the Large Chain's. The best profit strategy of the Small Chain seems to be to open an additional facility which generates a rather small market share. This is due to the central position close to Murcia, of its two existing facilities, already grabbing most of the very large market of this very important demand point. It is therefore better for the Small Chain to open its new facility in a less equipped and cheaper area, close to the west-most facility of the Large Chain, thus obtaining a relatively small market share, but also giving rise to a small cannibalization effect, which may, as pointed out earlier, quite easily be reduced by small sacrifices on profit.

The situation is quite different for the Large Chain. Its best profit solution is to go for the expensive western part of the large demand in Murcia, thereby grabbing a quite large new market share, but also resulting in a corresponding quite high cannibalization, particularly to its main facility (L_2) at the south-east of the city. And this cannibalization is not reduced as easily.

7.3.2 Effect of additional constraints

In this second part of our case study we focused on the changes in the two objectives and in the market share captured by the new facility in case constraints are added to the problem. As before, calculations were done for four different values of δ.

In the first case, we add circular forbidden regions around the existing chain-owned facilities modifying the feasible set R for both objectives. This is an indirect technique to avoid cannibalization, which is quite common, probably because it is easy to implement and to enforce. We study the model when the radii of the forbidden regions are 5, 10 and 15 km (recall that the total width of the feasible region R was around 50 km).

In the second case we directly bound cannibalization: we add constraints which guarantee that the cannibalization suffered by *each* of the existing chain-owned facilities is less than a given percentage *ca* of the market share that it captured before the entry of the new facility. For each chain-owned existing facility ℓ a constraint of the form

$$can(\ell, nf) \leq ca \cdot msb(\ell),$$

should be added or equivalently,

$$msa(\ell, nf) \geq (1 - ca) msb(\ell).$$

The results obtained when the Larger Chain wants to locate a new facility are summarized in Table 7.3. The left-hand part of this table shows the effects of adding forbidden regions of different sizes, the right-hand part concerns several cannibalization thresholds. The two first columns in both parts contain the value of each objective at the final optimal solution to (P_2^δ), while the last column gives the corresponding market share of the new facility.

As could have been expected, for all the δ considered, both the profit, Π and the cannibalization, *Can* decrease as the radii of the forbidden circles around the existing chain-owned facilities increase. However, the decrease is higher for *Can* than for Π, that is, the addition of forbidden regions around the facilities helps more to reduce cannibalization than to worsen the profit of the chain. Therefore, it is an effective protectionist measure. On the other hand, the market share that the new facility loses is very large. Thus, one should employ this kind of measure carefully.

Concerning the direct cannibalization bounding constraints, the reduction of Π is much smaller than for *Can*. Notice that *Can* does not decrease at the same rate as the decrease of the allowed cannibalization. This means that cannibalization is particularly suffered by one (or some) of the members of the chain, not all. And as expected, the smaller the allowed cannibalization, the smaller the market share captured by the new facility.

Table 7.3: Sensitivity of objectives to constraint addition when the Large Chain is locating.

δ-optimality for (P_1)	Circular forbidden regions: radii of constraints	Π	*Can*	Market new	Forbid cannibalization ratio: Cannib. allowed	Π	*Can*	Market new
0.00	–	673.6	9.1	20.4	ca =100%	673.6	9.1	20.4
	$r = 5$ Km	664.9	7.4	17.2	ca =50%	673.6	9.1	20.4
	$r = 10$ Km	660.8	5.7	11.7	ca =20%	670.5	7.7	17.6
	$r = 15$ Km	641.8	2.8	5.5	ca =10%	657.8	4.2	8.8
0.01	–	672.5	8.2	18.6	ca =100%	672.5	8.2	18.6
	$r = 5$ Km	663.8	6.6	15.6	ca =50%	672.5	8.2	18.6
	$r = 10$ Km	659.7	4.9	10.1	ca =20%	669.4	7.2	16.7
	$r = 15$ Km	640.8	2.1	4.2	ca =10%	656.7	4.0	8.4
0.05	–	668.0	6.8	15.9	ca =100%	668.0	6.8	15.9
	$r = 5$ Km	659.4	4.7	9.7	ca =50%	668.0	6.8	15.9
	$r = 10$ Km	655.3	3.7	7.8	ca =20%	664.9	6.0	14.2
	$r = 15$ Km	636.5	0.7	1.3	ca =10%	652.3	3.2	6.8
0.10	–	662.5	5.6	13.2	ca =100%	662.5	5.6	13.2
	$r = 5$ Km	653.9	3.4	7.3	ca =50%	662.5	5.6	13.2
	$r = 10$ Km	649.8	2.9	6.1	ca =20%	659.4	4.7	9.7
	$r = 15$ Km	632.3	0.1	0.3	ca =10%	646.8	2.5	6.6

These results clearly demonstrate that of both protection measure types, the second strategy based on explicit cannibalization bounding is better than the first, based on distance bounds: the reduction of Π is smaller, *Can* keeps decreasing (although at a smaller rate) and the market share that the new facility keeps capturing is remarkably higher.

7.4 Conclusions

We have presented a biobjective facility location and design model on the plane where a chain wants to locate a single new facility in a geographical area in which other facilities (some belonging to the same chain, others to competing chains) exist offering the same product. The main objective is the maximization of the profit obtained by the chain. The secondary (less important) objective is the minimization of the cannibalization suffered by the existing chain-owned facilities.

To cope with this problem, we have used the δ-lexicographic method, which takes the secondary objective into account, and allows a worsening of the first objective to be traded off with an improvement of the second objective. Computational studies show that the method is rather efficient. It obtains efficient solutions which reflect the preferences of the decision maker. The experiments on the discarding tests proved their usefulness: using all the tests, the computational time has been reduced by more than 80%.

Furthermore, the Lexicographical-like method allows the handling of additional constraints, such as other protective measures against cannibalization (forbidden regions surrounding the existing facilities or bounds on the cannibalization each of these are allowed to suffer). Through the economic analysis, we have shown that direct constraints on the cannibalization give better results than locational restrictions near the existing chain-owned facilities. These studies also showed the wide range of information which can be provided by the biobjective model using the Lexicographical-like method.

Chapter 8

Franchisor versus franchisee

In this chapter we study the case of a franchise which wants to enlarge its presence in a given geographical region by opening one new facility. Both the franchisor (the owner of the franchise) and the franchisee (the actual owner of the new facility to be opened) have the same objective: maximizing their own profit. However, the maximization of the profit obtained by the franchisor is in conflict with the maximization of the profit obtained by the franchisee. This suggests using a biobjective model to obtain the efficient solutions for this problem, so that later on the franchisor and the franchisee can agree in both location and design for the new facility, taking the corresponding economic implications of their selection into account.

The chapter is organized as follows: In Section 8.1 we introduce the biobjective model for the franchise problem [56]. The computational experiments are collected in Section 8.2. First, an economic analysis of the model is conducted in Section 8.2.2. The discarding test for the Constraint-like method [54], and for the biobjective B&B method [53] are compared in Section 8.2.3 and 8.2.5, respectively. Experiments on the selection rules of the biobjective B&B method are carried out in Section 8.2.4. We have compared the biobjective B&B method to the Lexicographical-like method in Section 8.2.6, and to the Constraint-like method in Section 8.2.7. The chapter ends with some conclusions in Section 8.3.

8.1 The model

A franchise wants to locate a new single facility in a given area of the plane, where there already exist m facilities offering the same goods or product. The first k ($k \geq 1$) of those m ($m > k$) facilities are part of the franchise. The *demand*, supposed to be *inelastic*, is concentrated at n demand points, whose locations dp_i and *buying power* ω_i are known. The location ef_j and quality of the existing facilities is also known. We will use the same notation as in Chapter 5. Again, we consider that the *patronizing behavior* of customers is *probabilistic*. The location and the quality of the new facility are the variables of the problem.

8.1.1 First objective: maximization of the market share captured by the franchisor

Similar to previous models, the total market share attracted by the franchise is

$$M(nf) = \sum_{i=1}^{n} \omega_i \frac{\frac{\gamma_i \alpha}{u_i(d_{iz})} + \sum_{j=1}^{k} \frac{\alpha_{ij}}{u_i(d_{ij})}}{\frac{\gamma_i \alpha}{u_i(d_{iz})} + \sum_{j=1}^{m} \frac{\alpha_{ij}}{u_i(d_{ij})}}.$$

We assume that the operating costs for the franchisor due to the new facility are fixed, that is, they are independent from the final location and quality chosen. This is usually the case, since the operating

costs for the franchisor are mainly due to the advertising of the franchise's trademark. In the same way, we also disregard the operating costs of the existing facilities that are part of the franchise.

In this way, the profit obtained by the franchisor is an increasing function of the market share captured by the franchise. Thus, maximizing the profit obtained by the franchisor is equivalent to maximizing the market share captured by the franchise. This will be the first objective of our problem.

8.1.2 Second objective: maximization of the profit obtained by the franchisee

The second objective of our problem is the maximization of the profit obtained by the franchisee, to be understood as the difference between the revenues obtained from the market share captured by the new facility minus its operational costs (see [51]). The market share captured by the new facility (by the franchisee) is given by

$$m(nf) = \sum_{i=1}^{n} \omega_i \frac{\dfrac{\gamma_i \alpha}{u_i(d_{iz})}}{\dfrac{\gamma_i \alpha}{u_i(d_{iz})} + \displaystyle\sum_{j=1}^{m} \dfrac{\alpha_{ij}}{u_i(d_{ij})}}$$

and the profit is given by the following expression,

$$\pi(nf) = F(m(nf)) - G(nf)$$

where F and G are as in Section 5.1 (see page 98).

8.1.3 The problem

The problem to be solved is then

$$\begin{array}{ll} \max & M(nf) \\ \max & \pi(nf) \\ \text{s.t.} & d_{iz} \geq d_i^{\min},\ i = 1, \ldots, n \\ & \alpha \in [q_{\min}, q_{\max}] \\ & z \in R \subset \mathbb{R}^2 \end{array} \tag{8.1}$$

where the parameters $d_i^{\min}$, $q_{\min}$, $q_{\max}$ and the set R are as in Section 5.1.

To clarify the biobjective nature of (8.1), consider Figure 8.1. Light gray circles with numbers 1 to 5 denote the forbidden regions around the existing demand points (supposed to be at the center of the forbidden regions, and all with demand 1), the cross $\times$ denotes the location of an existing facility owned by the chain and the $\boxdot$ marker the location of a competitor's facility. The franchisor would like the new facility to be located close to demand point 5 (he/she already captures the market of demand points 1 to 4, and in this way he/she can win a part of the market of demand point 5), whereas the franchisee would like the facility to be located close to the existing chain-owned facility (in this way, he can capture nearly half of the market of demand points 1 to 4, which is much more than he/she can get by locating close to demand point 5). In a color scale we can see an enclosure of the efficient set for this problem (the more red the better for the franchisor, and more blue the better for the franchisee).

In order to have problem (8.1) written in the form of problem (3.1), in what follows we will use the notation: $x = nf$, $f_1(x) = -M(nf)$, $f_2(x) = -\pi(nf)$ and S will denote the feasible set of problem (8.1).

8.2 Computational experiments

All the computational results presented in this section have been obtained on a PC with an Intel Pentium IV 2.33GHz processor and with 1 Gbyte RAM running under Linux operating system. For the implementation we have used the interval arithmetic from the PROFIL/BIAS library [91] and the automatic differentiation of the C++ toolbox library described in [69].

Figure 8.1: Conflicting objectives.

8.2.1 Test Problems

To test the different methods we have used two sets of problems. The first set is used for the economic analysis of the next section, while the second set for the rest of the experiments. In the first set (Set1 hereinafter) three problems were examined: one with 20 demand points, 4 existing facilities from which 3 belongs to the chain, i.e. $(n, m, k) = (20, 4, 3)$, and the other two with $(40, 5, 4)$ and $(60, 5, 4)$. These three problems were generated by randomly choosing the parameters of the problems uniformly within the following intervals:

- $ef_j, dp_i \in [0, 10]^2$,
- $\omega_i \in [1, 10]$,
- $\gamma_i \in [0.75, 1.25]$,
- $\alpha_{ij} \in [0.5, 5]$,
- $\phi_{i0} = \phi_0 = 2, \phi_{i1} \in [0.5, 2]$, the parameters for $\Phi_i(d_{iz}) = \omega_i \frac{1}{(d_{iz})^{\phi_{i0}+\phi_{i1}}}$, and $G_1(z) = \sum_{i=1}^{n} \Phi_i(d_{iz})$,
- $\rho_0 \in [7, 9], \rho_1 \in [4, 4.5]$, the parameters for $G_2(\alpha) = e^{\frac{\alpha}{\rho_0}+\rho_1} - e^{\rho_1}$,
- $c \in [1, 2]$, the parameter for $F(m(nf)) = c \cdot m(nf)$,
- $b_1, b_2 \in [1, 2]$, parameters for $d_{iz} = \sqrt{b_1(z_1 - dp_{i1})^2 + b_2(z_2 - dp_{i2})^2}$ (see [47]).
- $\lambda_i = \lambda = 2$, the parameter for $u_i(d_{iz}) = (d_{iz})^{\lambda_i}$.

The search space for every problem was

$$z \in [0, 10]^2, \quad \alpha \in [0.5, 5].$$

The second set of problems (called Set2) contains sixteen problems, but in this case the number of demand points is 50 or 100, the number of existing facilities is 2, 5 or 10, while the size of the chain is set to 0 or 1 if $m = 2$, to 0, 1 or 2 if $m = 5$ and to 0, 2 or 4 if $m = 10$. For each settings one instance was generated by choosing uniformly the parameters from the same intervals as above.

8.2.2 Economic analysis of the model using the Lexicographical-Like and the Biobjective interval Branch and Bound method

In this subsection, we try to obtain supplementary sensitivity information when modifying some of the parameters in the model or when adding some additional constraints to it. This may be viewed as an exploratory analysis of some of the economic implications of the proposed model. The aim is to acquire a deeper knowledge of the interactions between the different elements of the model that can be adjusted to real applications. We use the instances of Set1 here.

Table 8.1: Study of the variation in the range of one of the objectives in a region of δ-optimality of the other objective.

	(20,4,3)		(40,5,4)		(60,5,4)	
δ	$\pi(R^1_\delta)$	$M(R^2_\delta)$	$\pi(R^1_\delta)$	$M(R^2_\delta)$	$\pi(R^1_\delta)$	$M(R^2_\delta)$
0.01	[13.7, 40.0]	[166.7, 168.6]	[35.1, 42.2]	[358.5, 361.7]	[82.6, 94.7]	[537.3, 540.6]
0.05	[11.8, 43.3]	[165.4, 169.7]	[31.5, 47.6]	[356.9, 362.9]	[78.3, 116.0]	[534.8, 541.9]
0.10	[10.2, 46.7]	[164.3, 171.3]	[28.6, 92.8]	[355.4, 363.5]	[73.4, 127.1]	[532.5, 543.7]
0.15	[8.5, 52.7]	[163.6, 171.7]	[25.9, 101.3]	[354.3, 363.7]	[66.2, 137.7]	[530.6, 545.4]

First, we have studied the variation in the range of one of the objectives in a region of δ-optimality of the other objective for four different values of δ. The results are given in Table 8.1. Each pair of columns refers to one of the instances considered (whose actual settings is given is the first row). In the columns called $\pi(R^1_\delta)$ we give the exact range of the profit π at R^1_δ. The lower bound of this interval is just the solution of (P^δ_2) provided by the δ-lexicographic method; the upper bound has been estimated by solving the problem

$$\begin{array}{ll} \min & \pi(nf) \\ \text{s.t.} & nf \in R^1_\delta \ ' \end{array}$$

i.e. maximizing f_2 over R^1_δ. Notice that to solve this problem we need to obtain the whole region of δ-optimality; for this, we have used the main procedure of the δ-lexicographic method without the cut-off test due to the second objective function. Analogously, $M(R^2_\delta)$ gives the range of the market share at the region of δ-optimality R^2_δ of problem

$$\begin{array}{ll} \max & \pi(nf) \\ \text{s.t.} & nf \in S \ ' \end{array}$$

i.e. $\min_{x \in S} f_2(x)$.

As expected, the ranges increase with δ: e.g. $R^1_\delta \subset R^1_{\delta'}$ when $\delta < \delta'$. However, notice that the $\pi(R^1_\delta)$ ranges increase much more than the $M(R^2_\delta)$ ranges. Also, the width of the $\pi(R^1_\delta)$ ranges are much bigger than the width of the $M(R^2_\delta)$ ranges. This indicates that there are efficient points offering similar values for the first objective but quite different for the second. Thus, in a negotiation between the franchisor and the franchisee to decide the final location, the franchisor has a much more comfortable situation.

In our second analysis we studied how the addition of circular forbidden regions around the existing franchise-owned facilities affects both objectives, depending on the radii of the forbidden regions. The addition of this kind of constraints is a common technique in franchise systems to avoid that the existing facilities lose too much of their market share [50]. We have used the same instances as in the previous study, but now the biobjective interval B&B method is applied to explore the whole efficient set. Table 8.2 gives a summary of the results obtained for three different radii r of the circular forbidden regions.

The columns $\pi(M^{-1})$ give the range for the values of π on those efficient points at which M takes values within $I = [a\%, b\%]$ of its efficiency range $M(S_E)$. The columns $M(\pi^{-1})$ give similar figures, but interchanging both functions. For any given r, once we have obtained the whole efficient set of the constrained problem with the biobjective interval B&B method, this information can be obtained easily for any choice of $I = [a\%, b\%]$. Thus, each pair of 4-row sub-columns of Table 8.2 is obtained after a single B&B run, and shows only part of the full information provided. In fact using the $M(\pi^{-1})$ and $\pi(M^{-1})$ ranges given allows to reconstruct a coarser approximation of the efficient set, as shown in Figure 8.2.

As we can see from Table 8.2, in the chosen instances, the influence of the forbidden regions around the existing chain-owned facilities is quite small. In these instances the efficient set in the unconstrained problems lies far from the existing chain-owned facilities; thus, the addition of the constraints does not

Table 8.2: Sensitivity of the objectives to the size of the forbidden regions.

I	$(20,4,3)$ $\pi(M^{-1})$	$(20,4,3)$ $M(\pi^{-1})$	$(40,5,4)$ $\pi(M^{-1})$	$(40,5,4)$ $M(\pi^{-1})$	$(60,5,4)$ $\pi(M^{-1})$	$(60,5,4)$ $M(\pi^{-1})$
			$r=0$			
[0%, 25%]	[32.0, 43.1]	[167.4, 170.6]	[44.7, 61.8]	[402.0, 415.0]	[86.8, 117.8]	[539.5, 543.8]
[25%, 50%]	[42.8, 44.6]	[170.5, 171.6]	[61.5, 70.3]	[414.8, 421.8]	[117.1, 130.7]	[543.6, 547.4]
[50%, 75%]	[44.3, 46.8]	[171.6, 171.8]	[70.0, 77.9]	[421.6, 427.4]	[129.9, 142.4]	[547.2, 549.1]
[75%, 100%]	[46.5, 47.3]	[171.8, 171.9]	[77.7, 79.6]	[427.1, 428.8]	[141.5, 152.0]	[549.0, 550.3]
			$r=1.0$			
[0%, 25%]	[32.0, 43.1]	[167.4, 170.6]	[44.7, 61.8]	[402.0, 415.0]	[86.8, 116.9]	[540.2, 544.2]
[25%, 50%]	[42.8, 44.6]	[170.5, 171.6]	[61.5, 70.3]	[414.8, 421.8]	[116.1, 129.2]	[544.0, 547.6]
[50%, 75%]	[44.3, 46.8]	[171.6, 171.8]	[70.0, 77.9]	[421.6, 427.4]	[128.4, 140.2]	[547.4, 549.1]
[75%, 100%]	[46.5, 47.3]	[171.8, 171.9]	[77.7, 79.6]	[427.1, 428.8]	[139.4, 149.6]	[549.0, 550.3]
			$r=2$			
[0%, 25%]	[40.2, 43.7]	[167.7, 169.3]	[42.2, 62.1]	[401.5, 415.7]	[86.3, 119.1]	[538.7, 547.4]
[25%, 50%]	[43.4, 44.6]	[169.2, 170.6]	[61.1, 70.8]	[415.1, 423.0]	[117.3, 128.0]	[547.0, 548.8]
[50%, 75%]	[44.3, 46.5]	[170.4, 171.1]	[69.9, 78.1]	[422.2, 428.2]	[126.7, 130.3]	[548.6, 549.3]
[75%, 100%]	[46.2, 46.9]	[171.1, 171.4]	[77.7, 79.3]	[427.6, 428.8]	[129.4, 130.7]	[548.9,550.4]

affect the solution so much. When the constraints do have some influence (specially for the case $r = 2$), it mainly reduces the range of $M(\pi^{-1})$.

Our last experiment dealt with the sensitivity of both objectives when the franchisee has a limited budget, that is, we now have an additional constraint of the form $G(nf) \leq B$, where B is the available budget. We have again used the same instances as before the biobjective interval B&B method. The

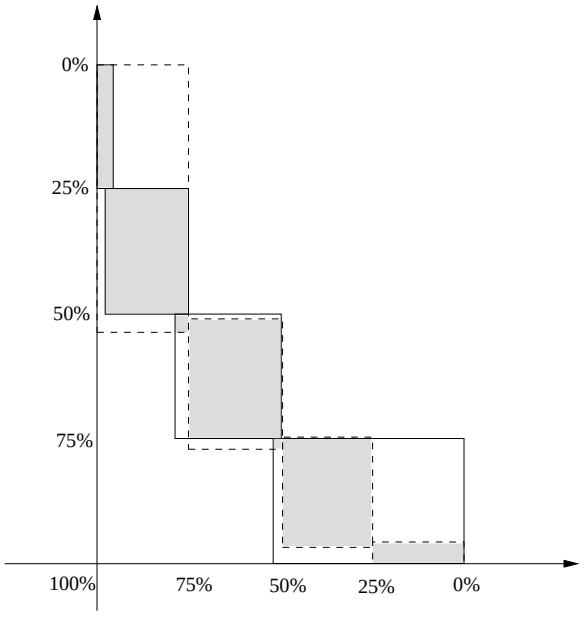


Figure 8.2: Approximate efficient set for the instance $(40,5,4)$, when $r = 2$

Table 8.3: Sensitivity of the objectives to the limited budget

	$(20,4,3)$		$(40,5,4)$		$(60,5,4)$	
I	$\pi(M^{-1})$	$M(\pi^{-1})$	$\pi(M^{-1})$	$M(\pi^{-1})$	$\pi(M^{-1})$	$M(\pi^{-1})$
$[0\%, 25\%]$	[14.5,43.3]	[167.4,171.7]	[42.3,62.3]	[401.6,414.0]	[86.8,120.5]	[537.6,544.3]
$[25\%, 50\%]$	[43.0,44.6]	[171.6,171.7]	[61.9,68.4]	[413.6,422.9]	[119.8,135.1]	[544.1,547.6]
$[50\%, 75\%]$	[44.3,46.8]	–	[68.1,77.8]	[422.6,427.3]	[134.2,143.7]	[547.5,549.1]
$[75\%, 100\%]$	[46.6,47.3]	[171.6,171.8]	[77.6,78.3]	[427.0,427.9]	[142.9,149.2]	[549.0,550.3]
Reduction of $B = 20\%$						
$[0\%, 25\%]$	[14.3,43.3]	[167.4,171.1]	[41.6,58.4]	[404.7,411.1]	[86.8,122.6]	[534.6,544.4]
$[25\%, 50\%]$	[43.0,44.6]	–	[57.8,63.8]	[410.8,421.0]	[121.9,131.4]	[544.1,548.2]
$[50\%, 75\%]$	[44.3,46.8]	–	[63.5,68.3]	[420.7,425.5]	[130.7,136.6]	[548.1,549.1]
$[75\%, 100\%]$	[46.6,47.3]	[171.0,171.8]	[67.7,75.2]	[425.2,426.5]	[135.8,141.6]	[549.0,550.3]
Reduction of $B = 30\%$						
$[0\%, 25\%]$	[16.9,44.1]	[167.6,170.4]	[39.0,53.9]	[402.3,407.5]	[84.9,115.7]	[530.7,543.9]
$[25\%, 50\%]$	[44.0,44.6]	–	[53.3,59.6]	[407.1,418.6]	[114.8,123.2]	[543.5,547.0]
$[50\%, 75\%]$	[44.5,46.8]	–	[59.2,61.4]	[418.2,423.0]	[122.5,125.6]	[546.8,548.3]
$[75\%, 100\%]$	[46.7,47.3]	[170.4,171.1]	[60.7,69.9]	[422.6,424.6]	[124.9,128.3]	[548.1,549.2]
Reduction of $B = 40\%$						
$[0\%, 25\%]$	[18.7,22.4]	[167.4,169.5]	[33.4,47.2]	[398.6,403.4]	[79.5,104.1]	[532.7,543.4]
$[25\%, 50\%]$	[22.0,44.6]	–	[46.5,53.2]	[402.8,415.7]	[103.2,109.5]	[543.0,545.0]
$[50\%, 75\%]$	[44.5,45.7]	–	[52.7,54.3]	[415.3,419.2]	[108.7,110.6]	[544.8,545.5]
$[75\%, 100\%]$	[45.4,47.2]	[169.5,170.4]	[53.8,61.7]	[418.7,421.7]	[110.0,110.7]	[545.3,546.3]
Reduction of $B = 50\%$						
$[0\%, 25\%]$	[19.5,22.5]	[166.3,168.4]	[21.9,39.5]	[395.0,405.6]	[70.2, 89.9]	[533.3,541.0]
$[25\%, 50\%]$	[22.1,43.6]	–	[38.6,43.7]	[397.6,412.4]	[88.9, 92.4]	[540.6,542.1]
$[50\%, 75\%]$	[43.5,44.6]	–	[43.0,44.0]	[412.0,415.1]	[91.6, 93.5]	[541.9,542.2]
$[75\%, 100\%]$	[44.3,46.0]	[168.4,169.5]	[43.3,50.9]	[414.6,416.3]	[92.8, 93.6]	[542.0,542.4]

results are presented in Table 8.3, where *Reduction of* $B = b\%$ refers to the reduction (in percentage) in the budget B as compared to the original budget needed for locating and running the facility in the unconstrained case (which corresponds to the case $r = 0$ of the previous table). The meaning of the columns is the same as in Table 8.2.

The influence of the budget constraint is now clearly noticeable. For example, in the problem with setting $(20, 4, 3)$, the two first quarter ranges for $M(\pi^{-1})$ are quite different when the budget cut increases from $B = 30\%$ to 40%. See also in the problem $(40, 5, 4)$, the ranges $M(\pi^{-1})$ for $I = [50\%, 75\%]$ and $[75\%, 100\%]$ when we change from $B = 10\%$ to 20% or the changes in the $M(\pi^{-1})$ ranges in the example with settings $(60, 5, 4)$.

Observe also from Table 8.3 that some of the $\pi(M^{-1})$ ranges for the $(20, 4, 3)$ problem are empty (they are represented by a '–'). This is because in that problem, the efficient set is disconnected (similar to figure 8.3): there are no efficient points at which π takes values within interval I.

8.2.3 Computational results of the Constraint-Like Method

In this section we study the performance of the Constraint-like method described in Algorithm 3.2, as well as the efficiency of the different discarding tests. Here we used the problems of Set2, and we set $\delta = 0.01$ (the parameter for the regions of δ-optimality) and the tolerances used in the algorithm were $\varepsilon = 0.01$ and $\eta = 0.005$.

First, we have studied the usefulness of the different discarding tests. To this end, we have solved all the problems with the following algorithms:

simple : in this algorithm we only use the δ-cut-off and the feasibility tests. In the later test, we use the corresponding natural interval extensions as the inclusion functions of the constraints to check the feasibility. The pruning test is substituted by a simple bisection of the box under consideration perpendicular to the direction of maximum width.

basic : we use the tests in *simple*, but now, in the feasibility test, for the constraint on f_2 we use the centered form as inclusion function.

basic + mono : we use the tests in *basic* and the δ-monotonicity test.

basic + mco : we use the tests in *basic* and the multiobjective cut-off test.

basic + prunf_2 : we use the tests in *basic* and the pruning test applied to f_2.

basic + prun$f_1 f_2$: we use the tests in *basic* and the pruning test applied to f_1 and f_2.

basic + mco + prun$f_1 f_2$: we use the tests in *basic* + *prun*$f_1 f_2$ and the multiobjective cut-off test.

basic + all : in which we use all the discarding tests, that is, we use as main procedure the one given in Algorithm 3.3.

The results obtained are given in Table 8.4. For the *basic* algorithm we computed the CPU time in seconds (Time), the effort of the algorithm (Effort, to be understood as the number of function evaluations plus three times the number of gradient evaluations), the maximum number of boxes stored in the lists ($\mathcal{L}_W^{min} + \mathcal{L}_S^{cur} + \mathcal{L}_W^{\delta} + \mathcal{L}_{Next}$) at any time during the execution of the algorithm (ML), the number of boxes in $\mathcal{L}_S$ (FB) and the volume of the boxes in the solution list $\mathcal{L}_S$ (Vol). For the rest of the algorithms, we computed the relative values of each of those indices as compared to the values for *basic*, in percentage. The values in Table 8.4 summarize those results, and give the corresponding values (in average) when we consider the sixteen problems altogether (Average), and the average of the relative values of each problem (Av. of %), respectively.

Whereas *basic* is able to solve all the problems, *simple* runs out of memory in thirteen of the sixteen problems (that is why we have not written any value in the average line; on the other hand, in the other

Table 8.4: Comparison of the different discarding tests of the Constraint-Like Method

Algorithm		Time	Effort	ML	FB	Vol
basic	Average:	336.6	1710024	13791	62043	2.4e-2
simple	Average:	–	–	–	–	–
	Av. of %:	461 %	755 %	1563 %	7360 %	4307 %
basic+	Average:	101.8 %	99.1 %	85.1 %	101.9 %	104.2 %
mono	Av. of %:	100.8 %	99.3 %	87.9 %	98.0 %	93.6 %
basic+	Average:	86.0 %	61.8 %	66.1 %	44.8 %	50.0 %
mco	Av. of %:	82.6 %	64.1 %	66.2 %	38.3 %	49.9 %
basic+	Average:	94.5 %	93.4 %	55.2 %	65.4 %	100.0 %
prunf_2	Av. of %:	92.8 %	86.9 %	54.7 %	64.4 %	103.9 %
basic+	Average:	79.3 %	70.9 %	67.6 %	52.2 %	83.3 %
prun$f_1 f_2$	Av. of %:	90.9 %	85.0 %	72.1 %	65.3 %	97.3 %
basic+mco	Average:	75.7 %	55.7 %	43.6 %	34.0 %	54.2 %
+prun$f_1 f_2$	Av. of %:	71.9 %	53.0 %	44.3 %	28.3 %	61.5 %
basic+	Average:	80.4 %	60.4 %	44.9 %	34.7 %	58.3 %
all	Av. of %:	75.5 %	59.1 %	46.9 %	27.6 %	53.9 %

line we compute the mean of the averages of the three problems for which *simple* finished). This clearly shows that the overestimation produced by the natural interval extension used in the feasibility test causes serious difficulties to the algorithm, which may be overcome with the use of the centered form. *Basic* solves all the problems in a reasonable amount of time (less than 6 minutes) and the volume of the outer approximating set of the efficient set is, in average, 0.024, that is, 0.0053% of the volume of the searching region.

The δ-monotonicity test does not seem to be useful for the type of problems under consideration, since it does not significantly reduce any of the parameters under study. On the contrary, the multiobjective cut-off test is very effective: it reduces the CPU time by more than 15%, the corresponding reductions in effort and maximum number of boxes stored are around 35%, and the number of boxes in $\mathcal{L}_S$ and their volume is reduced by almost half.

The pruning test applied to f_2 is also quite effective, specially in the reduction of storage (ML and FB). However, it is better to apply the pruning to both f_1 and f_2, since all the parameters under study (except ML) improve with regard to *basic* + *prun*f_2.

When the multiobjective cut-off test and the pruning test applied to f_1 and f_2 are used together, the algorithm obtains the best results. The CPU time is reduced by more than 25%, the effort and the volume of the solution boxes by nearly half, the maximum number of boxes stored is reduced by more than 55% and the number of final boxes by more than 70%. Although the reductions obtained when used jointly are not the sum of the individual reductions obtained with each discarding test alone, the results clearly show that when they are used together, the performance of the algorithm is much better. This is so because the type of information that they use is different, and thus, they discard different types of boxes.

If in addition to the previous tests we also use the δ-monotonicity test (i.e., the algorithm *basic+all*) then all the parameters under study either remain in similar values or slightly worsen, again confirming that the δ-monotonicity test is not useful for the type of problems under consideration.

8.2.4 Usefulness of the selection rules in the biobjective B&B method

The first part of the computational experiments devoted to the biobjective B&B method studies the efficiency of the selection rules presented in Section 3.4.1. We use the instances of Set2, and the tolerances used for the biobjective interval B&B method were $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 0.05$ with termination rule TR1, i.e. $(\mathrm{w}_{rel}(\boldsymbol{f}_1(\boldsymbol{x})) < \varepsilon_1$ and $\mathrm{w}_{rel}(\boldsymbol{f}_2(\boldsymbol{x})) < \varepsilon_2)$ or $\mathrm{w}(\boldsymbol{x}) < \varepsilon_3$.

All the problems were solved four times by the interval B&B method, using each selection rule in turn. The results are shown in Table 8.5. The columns *SRi* refer to the results obtained when using selection rule 'i' (see Section 3.4.1). In rule 3, we have set $\lambda = 0.5$. In the fourth selection rule, we have used $p = 10$ different values of λ. We first give results regarding CPU time. In column SR2 we give CPU times in seconds observed when using SR2, while for the other selection rules we give their relative times as compared to SR2. Next, we give the maximum number of boxes stored in memory at any time by the algorithm (ML), following the same structure as for the CPU time.

CPU times obtained by rules SR2 and SR4 are very similar, and they are both better than rule SR3. SR3 is only better on those problems where the efficient set is very small (see Table 8.7 for the range of the objectives and the solution). SR1 is the worst rule in global average, although in the average of percentages it is similar to SR3. As for the maximum number of boxes stored in memory at any time by the algorithm, we see that SR1, SR2 and SR4 are very similar, while SR3 is slightly worse. The best selection rule for these problems is different than for other problems (in [56] SR1 was the best rule). The explanation for this last fact seems to be that the width of the range of $f_1(= M)$ is usually greater than the corresponding range of $f_2(= \pi)$ (see Table 8.7) for these problems (it was the other way around for the problems in [56]). Thus the cut-off test will more easily discard boxes when using SR2 than when using SR1 (or any other rule) for the current problems.

This shows that rule SR2 is the best for handling these competitive location problems. In the rest of the experiments in this chapter, we have always used rule SR2.

Table 8.5: Efficiency of the selection rules.

Problem	SR2		SR1		SR3		SR4	
Type	T	ML	T%	ML%	T%	ML%	T%	ML%
50,2,0	20.3	4748	110.3	129.8	108.4	114.2	100.0	101.3
50,2,1	47.2	5886	92.2	144.0	122.9	156.0	102.1	108.2
50,5,0	52.5	9200	110.1	132.5	120.8	128.1	102.3	104.5
50,5,1	49.3	8301	115.4	136.0	126.6	144.6	99.0	104.1
50,5,2	135.3	12495	113.2	166.9	140.7	160.1	105.1	118.0
50,10,0	141.7	14510	135.1	127.4	117.3	125.6	104.0	104.0
50,10,2	98.4	38716	155.4	82.8	125.6	97.5	104.5	99.0
50,10,4	93.9	40468	144.6	65.2	162.1	96.3	118.8	102.7
100,2,0	15.7	3207	122.3	116.8	101.9	108.8	101.3	103.6
100,2,1	61.7	5948	105.5	104.1	127.6	120.8	108.4	102.4
100,5,0	101.5	22862	123.9	109.8	102.3	105.0	102.0	102.2
100,5,1	51.2	14636	99.8	82.5	96.1	100.3	100.4	101.2
100,5,2	117.8	26935	100.0	71.6	96.6	102.0	99.8	98.6
100,10,0	61.6	16400	103.7	107.4	97.1	102.7	100.5	102.5
100,10,2	175.8	45669	98.5	100.9	95.3	106.0	99.8	101.6
100,10,4	293.1	42691	123.3	90.7	105.2	101.5	106.6	103.6
Glo. Av.:	94.8	19542	118.1	97.5	114.4	108.3	104.2	102.6
Av. of %:			115.8	110.5	115.4	116.8	103.4	103.6

8.2.5 Comparison of the discarding tests of the biobjective B&B method

Now, we study the usefulness of the different discarding tests of the biobjective B&B method presented in Section 3.4. To this end, we have solved all the problems in Set2 with the following algorithms:

basic: The biobjective interval Branch and Bound method when only the feasibility and multiobjective cut-off tests are used.

basic + mono: This is *basic* with the multiobjective monotonicity test.

basic + mono + gen: In addition to the tests in *basic+mono* we also use the generalized multiobjective monotonicity test.

basic + prun: we use *basic* with the multiobjective pruning test.

basic + mono + prun: In addition to the tests in *basic+mono* we also use the multiobjective pruning test.

basic + all: We use all the discarding tests.

For all the above algorithms we have used SR2 and TR2 with tolerances $\varepsilon_1 = \varepsilon_2 = 0.1$.

Using the much stronger termination rule TR2 (see Section 3.4.6.1), the computation of the efficient set becomes more difficult, and optimization requires more memory. Therefore, a technical modification was done to the method to avoid swapping. Those boxes stored in the solution list which most probably will never be removed by the cut-off test, were saved from the solution list to a file. In particular, a box $\boldsymbol{x}$ is saved from the solution list if $\underline{f}_2(\boldsymbol{x}) < 1.1 * \tilde{f}_2$, where $(\tilde{f}_1, \tilde{f}_2)$ is the last provisional nondominated point entered in $\mathcal{L}_{PNS}$. When the working list empties, the solution boxes are read from the file, and a final cut-off test is done by all the points in $\mathcal{L}_{PNS}$. This programming trick is very useful when the efficient set (and so the solution set) is large, thus the required memory can decrease considerably.

The results obtained are summarized in Table 8.6. For the *basic* algorithm we computed the CPU time in seconds (Time), the effort of the algorithm (to be understood as the number of function evaluations plus three times the number of gradient evaluations, Effort), the maximum number of boxes stored in the working list $\mathcal{L}_{\mathcal{W}}$ at any time during the execution of the algorithm (ML), the number of boxes in the solution list $\mathcal{L}_{\mathcal{S}}$ (FB), the volume of the boxes in $\mathcal{L}_{\mathcal{S}}$ (Volume) and the area that they cover when

Table 8.6: Efficiency of the different discarding tests.

Algorithm	Value	Time	Effort	ML	FB	Volume	Area
basic	mean	1834.6	967190	20998	62919	0.0039	0.89
basic+	Glob. %	90.2 %	78.3 %	64.5 %	84.5 %	92.3 %	94.4 %
mono	Av. of %	61.3 %	60.9 %	62.3 %	52.7 %	45.5 %	78.1 %
basic+	Glob. %	30.5 %	60.6 %	64.5 %	42.6 %	59.0 %	57.3 %
mono+gen	Av. of %	34.6 %	50.1 %	62.2 %	32.1 %	28.9 %	51.2 %
basic+	Glob. %	69.5 %	114.2 %	45.1 %	85.0 %	105.1 %	103.4 %
prun	Av. of %	75.4 %	98.4 %	53.3 %	82.2 %	110.8 %	101.5 %
basic+	Glob. %	52.4 %	92.0 %	31.6 %	71.8 %	94.9 %	97.8 %
mono+prun	Av. of %:	40.2 %	64.5 %	35.0 %	44.8 %	47.3 %	83.9 %
basic+	Glob. %	31.9 %	67.2 %	31.6 %	41.4 %	61.5 %	60.7 %
all	Av. of %	29.8 %	49.5 %	35.0 %	29.6 %	31.0 %	55.9 %

projected onto the bidimensional location space (Area). For the rest of the methods the above figures are given in percentages with respect to the corresponding values of the *basic* algorithm. The values in Table 8.6 summarize those results, and give the corresponding values when we consider the sixteen problems altogether (Glob. %), and the average of the relative values of each problem (Av. of %), respectively.

As we can see, *basic* solves all the problems in a reasonable amount of time (an average of approximately 30 minutes) and the volume of the outer approximating set of the efficient set is, in average, 0.0039, that is, 0.00086% of the volume of the search region. However, the use of the new discarding tests presented in Section 3.4.3 can make the algorithm much more efficient.

The multiobjective monotonicity test reduces the CPU time by 9.8% when we consider all the problems together, and the average reduction in the sixteen problems attains a considerable rate of 38.7% (in some problems it is around 85%, whereas in others it is nearly negligible). In all the sixteen cases the CPU time is reduced. The corresponding average reductions in effort and maximum number of boxes stored are around 40%, and the number of boxes in $\mathcal{L}_{\mathcal{S}}$ and their volume is reduced to nearly half. The area is reduced less, around 22%.

Since in order to apply the generalized monotonicity test we need to compute everything that is necessary to apply the monotonicity test, there is no reason not to apply both tests together. When doing it, the CPU time used by the algorithm when considering all the problems together is just 30.5% of the time used by *basic*, that is, a reduction of 69.5%. Now the CPU time of all the problems is reduced considerably (the reduction varying from 97.9% to 29.3%), with an average of 65.4%. We can also see a considerable reduction as compared to *basic+mono* in the number of boxes in the solution list, volume and area and a smaller reduction in effort. Thus, we can see that the generalized multiobjective monotonicity test works very differently from the multiobjective monotonicity test, and they discard different type of boxes. On the other hand, ML has similar values to those of *basic+mono*.

As for the multiobjective pruning test, it is also quite effective in reducing CPU time. When considering all the problems together, the algorithm uses 69.5% of the time of *basic*, and the corresponding average time in the sixteen problems is 75.4%. However, as with the multiobjective monotonicity test, in some problems the reduction is quite large, whereas in other problems it is nearly negligible (in fact, in three problems there is an increase in the CPU time). The reduction in the number of boxes in the solution list is also important, around 15%. However, Effort, Volume and Area remain similar or slightly worsen as compared to the corresponding values in *basic*. It is in the maximum number of boxes stored at any time during the algorithm where the multiobjective pruning test proves to be most effective, with reductions of around 50% as compared to *basic* (and this reduction holds for all the problems).

If we use the multiobjective monotonicity test in addition to the multiobjective pruning test, then the reduction in time is nearly the sum of the reductions obtained independently by each discarding test, which proves that those tests work on different types of boxes. ML and FB are also better than

when each test is used alone, although the reductions are not as high as that of time. On the other hand, the values of Effort, Volume and Area are better than those of *basic+prun*, but worse than those of *basic+mono*.

Finally, when the three new discarding tests are used, the reductions in CPU time, Effort, FB, Volume and Area are similar to those of *basic+mono+gen*, but memory requirements are nearly half, similar to those of *basic+mono+prun* (see the values of ML). Thus, *basic+all* is the best algorithm for handling the biobjective problems.

8.2.6 Biobjective B&B versus Lexicographical-Like Method

Let us now try to compare the information obtained by the biobjective interval Branch and Bound method introduced in Section 3.4 and the modification of the δ-lexicographic method described in Section 3.2. First observe that these methods are *not* directly comparable, since they do very different things: whereas the former obtains the *whole* efficient set, the latter just gives a *small part* of it. That is why we highlight the different kind of information that they provide.

Figure 8.3 shows the solution boxes (projected in the location space) and their image in criterion space obtained by each algorithm in one of the instances. As we can see, the nondominated set in criterion space can have a disconnected shape. Accordingly, the efficient set may also be disconnected. This clearly illustrates the difficulty of finding the complete efficient set for this kind of location problems. Also notice that (most of) the region $R^2_{\theta(\delta)}$ offered by the modified δ-lexicographic method is included in the solution set offered by the biobjective interval B&B method (which, as we recall, contains the complete efficient set); in fact, in all our instances, it was a proper subset of it. This confirms that by choosing an appropriate θ, the points in $R^2_{\theta(\delta)}$ are either efficient or lie very close to efficient points.

In our computational study we have used the problems in Set2. In this study we used only the multiobjective cut-off test as elimination rule. The tolerances used for the biobjective B&B method in

Table 8.7: Study of the information provided by the modified δ-lexicographic method and the biobjective interval B&B method.

Problem Type	CPU time		size of solution boxes z-space		size of solution boxes α-space		width of range at $\cup_{\boldsymbol{y}\in\mathcal{L}_S}\boldsymbol{y}$ f_1		width of range at $\cup_{\boldsymbol{y}\in\mathcal{L}_S}\boldsymbol{y}$ f_2	
(n,m,k)	Lexic.	B&B	Lexic.	B&B	Lexic.	B&B	Lexic.	B&B	Lexic.	B&B
50,2,0	10.8	20.1	0.19	0.38	0.18	0.53	3.36	10.30	0.85	2.63
50,2,1	4.7	32.6	0.04	0.59	0.13	0.95	1.32	15.85	0.74	8.99
50,5,0	10.3	45.8	0.05	0.41	0.11	0.91	1.41	15.97	0.47	8.62
50,5,1	9.2	42.5	0.05	0.27	0.18	1.16	1.70	16.32	0.54	7.25
50,5,2	7.9	90.8	0.00	0.99	0.02	1.02	0.41	16.05	0.39	18.59
50,10,0	25.6	104.4	0.07	0.45	0.09	1.62	1.24	23.15	0.54	15.38
50,10,2	13.3	94.3	0.00	0.05	0.00	2.67	0.09	20.44	0.06	11.46
50,10,4	6.6	90.6	0.00	0.03	0.00	3.16	0.04	21.77	0.01	15.13
100,2,0	18.3	15.4	0.22	0.48	0.11	0.07	4.54	5.68	2.06	14.46
100,2,1	14.4	53.3	0.00	1.69	0.01	0.07	0.69	26.35	1.36	63.50
100,5,0	36.6	101.2	0.02	0.71	0.04	0.07	2.23	11.62	0.62	10.19
100,5,1	26.5	51.2	0.00	0.09	0.04	0.18	1.03	7.41	0.65	5.37
100,5,2	50.0	117.9	0.05	0.24	0.05	0.11	1.92	3.71	0.40	2.21
100,10,0	66.0	61.7	0.12	0.32	0.07	0.07	2.47	3.44	0.81	2.44
100,10,2	56.8	175.9	0.00	0.10	0.21	1.09	3.34	19.03	0.26	5.50
100,10,4	41.2	278.8	0.00	0.59	0.16	1.12	1.50	21.10	0.11	29.18
Average	24.9	94.8	0.05	0.46	0.09	0.93	1.70	14.89	0.62	13.81

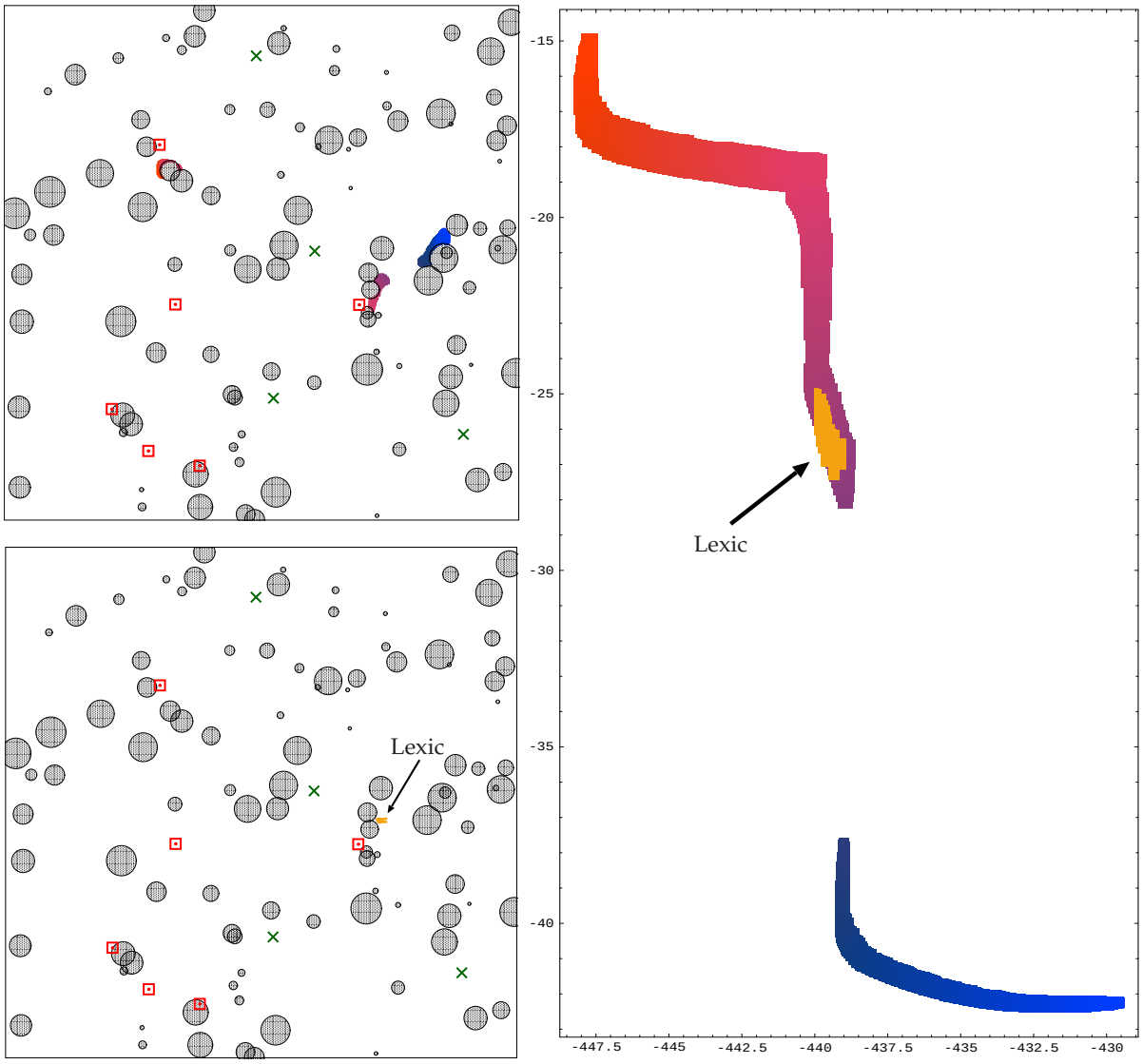


Figure 8.3: Solution of an instance by the methods. The light gray circles correspond to the forbidden areas surrounding the demand points; the crosses $\times$ and the markers $\boxdot$ represent the locations of the existing facilities. The top-left figure is the projection on the location space of the efficient set (in a color scale) obtained by the interval B&B; the bottom-left figure is the projection on the location space of the solution boxes (in orange) offered by the modified δ-lexicographic method (they are indicated by an arrow); in the right picture, we have the corresponding images in the solution space.

TR1 were $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 0.05$, and for the modified δ-lexicographic method $\delta = 0.01$ and $\theta = 0.001$. The results are summarized in Table 8.7. For each algorithm (*Lexic* refers to the modified δ-lexicographic method and *B&B* to the interval B&B method) we give the CPU time (in seconds), the area covered by the solution boxes when they are projected to the location space (i.e., the area of $\cup_{\boldsymbol{y}\in\mathcal{L}_S} proy_z(\boldsymbol{y})$), the width of the interval containing the possible values of α (obtained by projecting the solution boxes to the α-space, i.e., the width of $\cup_{\boldsymbol{y}\in\mathcal{L}_S} proy_\alpha(\boldsymbol{y})$), and the width of the range of f_1 and f_2 at the solution set offered by each algorithm.

As was expected, in general, Lexic is faster than B&B. We can also observe that CPU-time increases

with n for the Lexic and for the B&B method as well. For Lexic, and for a fixed pair (n, m), CPU time generally decreases as k increases; this is so, because usually the region of δ-optimality R^1_δ becomes smaller as k increases, as we can see from the columns giving the size of the solution boxes and the width of the range.

As for the area covered by the solution boxes, notice that there is no connection between the number of demand points (n) and the z-space for Lexic. The α-space may decrease with the increase of n, but not significantly. On the other hand, the area in z-space is more or less constant for B&B, whereas, similar to Lexic, in α-space it decreases as n increases. In all cases, we can see that the area covered by the solution boxes, both in z-space and α-space, is much smaller (from 2 to 10 times) for Lexic than for B&B, which clearly shows that Lexic only finds a small part of the efficient set.

Something similar can be said about the width of the ranges of both functions at the solution boxes offered by the algorithms. Again, the range for Lexic is from 2 to 20 times smaller than for B&B for f_1, and the difference is even higher for f_2. With Lexic the ranges increase with n; for a fixed (n, m) the ranges decrease as k increases. With B&B, the range of f_1 is more or less constant for all the settings, whereas in the range of f_2 there are larger differences (from 2.2 to 63.5).

8.2.7 Biobjective B&B versus Constraint-Like Method

Here, we have compared the best implementation of the Constraint-Like Method (CLM) introduced in Section 3.3 with the best biobjective interval Branch and Bound method presented in Section 8.2.5 (basic+all). The results obtained are summarized in Table 8.8.

For CLM we give the values obtained when the tolerances used are $\delta = 0.0001$, $\varepsilon = 0.01$ and $\eta = 0.004$ (see Section 3.3), whereas for *basic+all* we give the relative values of each of those indices as compare to the other algorithm, using $\varepsilon_1 = \varepsilon_2 = 0.1$.

Table 8.8: Comparison with the constraint-like method

Algorithm	value	Time	Effort	ML	FB	Volume	Area
CLM	Mean:	1864.1	3407872	7802	95783	5.1e-03	8.4e-01
basic+	Glob. %:	22.0	15.3	82.1	20.2	33.3	46.4
all	Av. of %:	25.5	18.8	105.4	17.2	23.5	60.5

As we can see, for this type of problems, the biobjective interval Branch and Bound method clearly outperforms the Constraint-Like Method. The implementation of *basic+all* is much faster (it needs around 75% less time than CLM), its effort is much smaller (around 85% less) and the number of boxes in the solution list is also much smaller (around 80% less). In addition to those reductions, the interval Branch and Bound method also gives tighter outer approximations of the efficient set: both the volume and the area are considerably reduced. When considering all the problems together, the volume of the outer approximation with *basic+all* is just one third of that of CLM, and the area is less than half. Only the memory requirements are not dramatically reduced. Considering all the problems together, the reduction is 17.9%, although for some particular problems there is an increase in ML. In fact, the average of the relative values when we consider the sixteen problems is slightly greater than 100%.

8.3 Conclusions

We have shown how to tackle a particular biobjective location problem with difficult nonlinear objective functions. In the particular three-dimensional setting used here the computational burden remained very acceptable. How the methods will behave in higher dimension remains to be studied. It is to be expected that the efficient set will grow in size, however, with a corresponding growth in computation time and memory usage. This also holds for multiple objective settings.

The economic analysis of the model gives an indication of the wealth of information obtained by the Lexicographical-like method. For the particular application in competitive location used here this kind of analysis should be very useful, e.g. in a negotiation between the franchisor and the franchisee, in which the former might want to convince the latter through some payment to compensate for a lower income in order for the whole chain to have a larger profit. Having the complete efficient set, and its corresponding two-dimensional graph in the criterion space, may help to reach an agreement, since each can know by how much one of them improves while the other loses. Further exploitation of this type of information is currently under study.

The computational studies showed that the new discarding tests for the biobjective Branch and Bound method and for the Constrained-Like Method are very efficient, and can reduce the required time by 70% and 25%, respectively. In the last study we have also seen that the biobjective B&B outperforms the Constrained-Like Method for this kind of location problems.

The most important feature of the methods used in this chapter is that they can be applied to any nonlinear biobjective optimization problem of the form (3.1). The only requirement is the feasible set be written through a set of analytical constraints and have inclusion functions for the functions defining the problem (but this second requirement holds for nearly any function written analytically). Thus, the potentiality of use of these methods is huge and varied. This shows that interval analysis based algorithms are also efficient methods to cope with multiobjective global optimization problems.

Symbol Index

Indices and other constants

n	number of demand points
m	number of existing facilities
k	number of existing facilities which belong to the locating chain (the first k of the m existing facilities are assumed to be in this category)
i	index of a demand point, $i = 1, \ldots, n$
j	index of an existing facility $j = 1, \ldots, m$

Variables

z	location of the new facility when a single facility problem is considered: $z = (z_1, z_2)$
z_l	location of a new facility when a two-facility problem is considered: $z_l = (z_{l1}, z_{l2}), l = 1, 2$
α	quality of the new facility when a single facility problem is considered
α_l	quality of a new facility when a two-facility problem is considered, $l = 1, 2$
nf	variables of the new facility/facilities. If a single facility is to be located $nf = (z, \alpha)$, if two $nf = (nf_1, nf_2), nf_l = (z_l, \alpha_l), l = 1, 2$

Data

dp_i	location of demand point i, $i = 1, \ldots, n$
ef_j	location of existing facility j, $j = 1, \ldots, m$
α_{ij}	quality of facility j as perceived by demand point i
ω_i	buying power of demand point i
γ_i	weight for the quality of nf as perceived by demand point i
c	income per unit of good sold
ϕ_{i0}, ϕ_{i1}	parameters of the location cost function $\Phi_i(d_{iz})$
ρ_0, ρ_1	parameters of the quality cost function $G_1(\alpha)$
d_i^{min}	minimum distance from demand point i at which the new facilities can be located
q_{min}	minimum allowed quality for the new facilities
q_{max}	maximum allowed quality for the new facilities

Functions

$M(nf)$	market share of the chain
$F(M)$	income of the chain
$\Phi_i(d_{iz})$	cost function related to the closeness of the facility to demand point i
$G_1(z)$	cost of locating a facility at z
$G_2(\alpha)$	cost of achieving a quality α
$G(nf)$	cost function of the new facility, $G(nf) = G_1(z) + G_2(\alpha)$
$\Pi(nf)$	profit of the chain, $\Pi(nf) = F(M(nf)) - G(nf)$

d_{ij}	distance between demand point i and facility j
$u_i(\cdot)$	a non-negative non-decreasing function to describe the distance decay
d_{iz}	distance between demand point i and new facility nf
d_{iz_l}	distance between demand point i and new facility nf_l

Summary

This thesis has addressed two main goals: to contribute to Reliable Global Optimization with new methods and new accelerating devices, and to contribute to Competitive Facility Location with new realistic models.

First, we have developed new accelerating devices for the interval Branch and Bound method. In order to choose the inclusion function that best works within the interval B&B method solving a given type of problem, the empirical convergence speed has been introduced and a comprehensive computational study has been presented for a large set of test problems. Using an adaptive multi-inclusion B&B algorithm, it has been demonstrated for a class of facility location problems that the suggestion of the empirical convergence speed is adequate even when accelerating devices (available for the used inclusion functions) are applied.

Another kind of accelerating devices, so-called pruning techniques or discarding tests, have also been developed. These accelerating devices discard (parts of) boxes, which cannot contain the global optimum by using the derivative information in a smart way. Two such techniques were developed. Both of them construct a linear lower bounding function of the objective function, and discard those parts of the box where the lower bounding function is above the best upper bound of the minimum. The first technique generalizes the one-dimensional pruning method to multi-dimensional problems by converting all but one of the variables to a constant interval, thus viewing the problem as a one-dimensional one. In this method, support functions were also used to achieve even better results. It has been shown in the computational experiments, conducted on a large set of test functions, that in average, the obtained speedup in time is about 1.5. Our second technique, the Baumann Tent Pruning-Dividing Method builds the linear lower bounding function directly for the multi-dimensional function. This allows us to have a larger pruneable *region*, although its non-interval shape makes it not readily applicable. The computation of the best pruneable *box* requires a smart technique: by using the Baumann point an efficient pruning device have been constructed. For a large set of test functions, the achieved acceleration is shown, which is in average about 1.5.

Besides the new accelerating devices mentioned above, we have also made some modifications on some existing ones. The three tests that we have modified have been extended to make them applicable not only to feasible boxes, but to undetermined boxes as well. In the tests which use second derivative information, namely, the projected one-dimensional interval Newton method and the projected one-dimensional non-convexity test, the evaluation of the whole Hessian matrix is not required, hence saving a lot of effort.

The devices discussed above were applied in the interval Branch and Bound method when solving our single facility planar location and design problem. The constructed model closely approximates reality. It assumes that costumers patronize facilities proportionally to the attraction they feel for them, where the attraction depends not only on the distance to the facility but also on the perceived quality of the facility. The objective is the maximization of profit, that is, the running costs of the new facility are taken into account. We have solved a large set of problems, and have shown that the interval B&B method is effective on these problems. We have also shown that the applied discarding tests can reduce the computational time by 80% altogether, and that the reduction due to the modifications of the existing elimination rules is 27-39% in average. In order to have a deeper understanding of the constructed model, an extensive sensitivity analysis has been conducted (with the help of the interval method) on a quasi-real problem, checking the importance of the different parameters of the model.

We have generalized the single facility planar location and design model to the case when two facilities are to be located at the same time. We have examined two main approaches: the sequential approach, which first determines the location and design of one facility, and then, taking it as an existing facility, determines the location and design of the other facility; and the simultaneous approach, which determines the location and quality of both new facilities at the same time. The results of the two approaches have been compared on some test problems, and although the simultaneous approach is significantly slower than the sequential one, it provides much better results.

The location model was extended to consider more than one objective. First, we have formulated a biobjective model where the chain not only wants to maximize its profit, but also to minimize the cannibalization suffered by the existing chain-owned facilities. Second, the case of a franchise has been studied, in which both the franchisor and the franchisee want to maximize their profit. In this last model, the operational costs charge only the franchisee, while the profit of the franchisor is a given percentage of the income of all the franchisees.

In order to deal with the biobjective problems we have developed three new methods. The Lexicographical-like method is able to find weakly efficient points from any part of the weakly efficient set, and it is specially suitable for problems in which one of the objectives is much more important than the other. Therefore, this method was used to solve the cannibalization problem, for which an economic analysis was done as well. On a set of test problems we have also checked the usefulness of the different discarding tests designed for the method. In average, 80% of the computational time can be saved using all the discarding tests as compared to the basic method.

The Constrained-like method has been developed in order to find an enclosure of the whole efficient set. We have applied this method to some franchise problems, and it proved to be quite effective. We have tested the performance of the discarding tests developed for that algorithm, and it was shown that with the best configuration we can reduce the computational time by about 25%.

Finally, a biobjective Branch and Bound method was developed. It deals with the two objectives directly (without reducing the problem to a single-objective one), and finds also a tight enclosure of the efficient set. This method uses the same structure as the interval B&B method, but in this case, boxes are discarded only if they are dominated by some feasible point. New discarding tests were developed, which proved to be very efficient: using all of them, computational time is reduced by 70% in average as compared to the basic method. Since all the results were obtained on the same set of franchise problems used with the Constrained-like method, we have compared the performance of the two algorithms. The biobjective interval B&B method was four times faster for that type of problems, and found a tighter enclosure of the efficient set. It has also been shown that the biobjective interval B&B method has good convergence properties.

As we have seen, interval methods are successfully applicable to the above problems. In the future, we plan to widen the range of such problems. On the one hand, we aim to construct new accelerating devices for the existing methods, and to develop new reliable methods. On the other hand, we plan to solve other, more difficult competitive facility location problems. For instance, we are going to examine the Leader-Follower problem, where it is assumed that after the Leader company locates a new facility, a competing company (the Follower) will also build a new facility. Thus, the Leader must optimize its profit taking into account the reaction of the Follower. The objective of the Leader does not have an analytical form, therefore, existing interval methods are not able to deal with it. Hence, it is a future challenge for us to develop a reliable method which can cope with this problem.

Bibliography

[1] D. Achabal, W. L. Gorr, and V. Mahajan. Multiloc: A multiple store location model. *Journal of Retailing*, 58:5–25, 1982.

[2] P. J. Agrell, B. J. Lence, and A. Stam. An interactive multicriteria decision model for multipurpose reservoir management: The shellmouth reservoir. *Journal of Multi-Criteria Decision Analysis*, 7:61–86, 1998.

[3] G. Alefeld and J. Herzberger. *Introduction to Interval Computations*. Academic Press, New York, 1983.

[4] E. Baumann. Optimal centered forms. *BIT*, 28:80–87, 1988.

[5] M. Berz and G. Hoffstätter. Computation and application of taylor polynomials with interval remainder bounds. *Reliable Computing*, 4:83–97, 1998.

[6] J. Brimberg and R. F. Love. *Facility Location: A Survey of Applications and Methods*, chapter Estimating Distances, pages 9–32. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, Berlin, 1995.

[7] M. Bräuer, W. Hofschuster, and W. Krämer. Steigungsarithmetiken in C-XSC. Preprint, Universität Wuppertal, 2001.

[8] J. C. Burkill. Functions of intervals. *Proceedings of the London Mathematical Society*, 22:375–446, 1924.

[9] E. Carrizosa, E. Conde, and D. Romero-Morales. *Advances in multiple objective and goal programming 1996*, chapter Location of a Semiobnoxious Facility: A biobjective approach, pages 338–346. Springer, 1997.

[10] E. Carrizosa and F. Plastria. A characterization of efficient points in constrained location problmes with regional demand. *Operatios Research Letters*, 19:129–134, 1996.

[11] L. G. Casado, I. García, and T. Csendes. A new multisection technique in interval methods for global optimization. *Computing*, 65:263–269, 2000.

[12] L. G. Casado, I. García, J. A. Martínez, and Y. D. Sergeyev. New interval analysis support functions using gradient information in a global minimization algorithm. *Journal of Global Optimization*, 25:345–362, 2003.

[13] Centro Regional De Estadística de Murcia, Dirección General de Economía, Planificación y Estadística. *Anuario estadístico de la Región de Murcia 2003*, 2003.

[14] V. Chankong and Y. Y. Haimes. *Multiobjective Decision Making Theory and Methodology*. Elsevier Science Publishing Co., New York, 1983.

[15] J. L. Cohon. *Multiobjective Programming and Planning*. Academic Press, Inc., New York, 1978.

[16] R. Colomé and D. Serra. Consumer choice and optimal location models: Formulations and heuristics. *Papers in Regional Science*, 80:439–464, 2001.

[17] J. L. D. Comba and J. Stolfi. Affine arithmetic and its applications to computer graphics. In *Anais do VI Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens (SIBGRAPI'93)*, pages 9–18, Recife (Brazil), Oct. 1993.

[18] T. Csendes. Numerical experiences with a new generalized subinterval selection criterion for interval global optimization. *Reliable Computing*, 9:109–125, 2003.

[19] T. Csendes and D. Ratz. Subdivision direction selection in interval methods for global optimization. *SIAM Journal on Numerical Analysis*, 34:922–938, 1997.

[20] J. Current, H. Min, and D. Shilling. Multiobjective analysis of facility location decisions. *European Journal of Operational Research*, 49:295–307, 1990.

[21] J. R. Current and J. E. Storbeck. A multiobjective approach to design franchise outlet networks. *Journal of the Operational Research Society*, 45:71–81, 1994.

[22] L. H. de Figueiredo and J. Stolfi. Adaptive enumeration of implicit surfaces with affine arithmetic. *Computer Graphics Forum*, 15:287–296, Dec. 1996.

[23] L. H. de Figueiredo and J. Stolfi. *Self-Validated Numerical Methods and Applications*. Brazilian Mathematics Colloquium monographs. IMPA/CNPq, Rio de Janeiro, Brazil, 1997.

[24] L. C. W. Dixon and G. P. Szegö, editors. *Towards Global Optimization*. North-Holland Publishing Company, Amsterdam, 1975.

[25] L. C. W. Dixon and G. P. Szegö, editors. *Towards Global Optimization 2*. North-Holland Publishing Company, Amsterdam, 1978.

[26] T. Drezner. Optimal continuous location of a retail facility, facility attractiveness, and market share: an interactive model. *Journal of Retailing*, 70:49–64, 1994.

[27] T. Drezner. Location of multiple retail facilities with limited budget constraints in continuous space. *Journal of Retailing and Consumer Services*, 5:173–184, 1998.

[28] T. Drezner and Z. Drezner. Validating the gravity-based competitive location model using inferred attractiveness. *Annals of Operations Research*, 111:227–237, 2002.

[29] T. Drezner and Z. Drezner. Finding the optimal solution to the Huff based competitive location model. *Computational Management Science*, 1:193–208, 2004.

[30] T. Drezner, Z. Drezner, and S. Salhi. Solving the multiple competitive facilities location problem. *European Journal of Operational Research*, 142:138–151, 2002.

[31] T. Drezner and H. A. Eiselt. *Facility Location: Applications and Theory*, chapter Consumers in Competitive Location Models, pages 151–178. Springer-Verlag, Berlin, 2002.

[32] Z. Drezner, editor. *Facility Location: a Survey of Applications and Methods*. Springer, Berlin, 1995.

[33] Z. Drezner and H. W. Hamacher, editors. *Facility Location: Applications and Theory*. Springer-Verlag, Berlin, 2002.

[34] Z. Drezner and A. Suzuki. The big triangle small triangle method for the solution of non-convex facility location problems. *Operations Research*, 52:128–135, 2004.

[35] M. Ehrgott. *Multicriteria Optimization*. Springer, Berlin, 2nd edition, 2005.

[36] M. Ehrgott, K. Klamroth, and S. Schwehm. An MCDM approach to portofolio optimization. *European Journal of Operational Research*, 155:752–770, 2004.

[37] M. Ehrgott and D. M. Ryan. Constructing robust crew schedules with bicriteria optimization. *Journal of Multi-Criteria Decision Analysis*, 11:139–150, 2002.

[38] M. Ehrgott and M. M. Wiecek. *Multiple Criteria Decision Analysis: State of the Art Surveys*, chapter Multiobjective Programming, pages 667–722. Kluwer, 2005.

[39] H. A. Eiselt and G. Laporte. Facility location: a survey and application methods. In Z. Drezner, editor, *Objectives in location problems*, pages 151–180. Springer, 1995.

[40] H. A. Eiselt and G. Laporte. *Facility Location: A Survey of Applications and Methods*, chapter Objectives in Location Problmes, pages 151–180. Springer Serier in Operations Research and Financial Engineering. Springer-Verlag, Berlin, 1995.

[41] H. A. Eiselt and G. Laporte. Sequential location problems. *European Journal of Operational Research*, 96:217–231, 1996.

[42] H. A. Eiselt and G. Laporte. Demand allocation functions. *Location Science*, 6:175–187, 1998.

[43] H. A. Eiselt, G. Laporte, and J. F. Thisse. Competitive location models: A framework and bibliography. *Transportation Science*, 27:44–54, 1993.

[44] E. Erkut. Inequality measures for location problems. *Location Science*, 1:199–217, 1993.

[45] E. Erkut and S. Newman. Analytical models for locating undesirable facilities. *European Journal of Operational Research*, 40:275–291, 1989.

[46] J. Fernández, P. Fernández, and B. Pelegrín. A continuous location model for siting a non-noxious undesirable facility within a geographical region. *European Journal of Operational Research*, 121:259–274, 2000.

[47] J. Fernández, P. Fernández, and B. Pelegrín. Estimating actual distances by norm functions: A comparison between the $l_{k,p,\theta}$-norm and the $l_{b_1,b_2,\theta}$-norm and a study about the selection of the data set. *Computers and Operations Research*, 29:609–623, 2002.

[48] J. Fernández and B. Pelegrín. Sensitivity analysis in continuous location models via interval analysis. *Studies in Locational Analysis*, 14:121–136, 2000.

[49] J. Fernández and B. Pelegrín. Using interval analysis for solving planar single-facility location problems: new discarding tests. *Journal of Global Optimization*, 19:61–81, 2001.

[50] J. Fernández, B. Pelegrín, F. Plastria, and B. Tóth. Planar location and design of a new facility with inner and outer competition: An interval lexicographical-like solution procedure. *Networks and Spatial Economics*, 7:19–44, 2007.

[51] J. Fernández, B. Pelegrín, F. Plastria, and B. Tóth. Solving a Huff-like competitive location and design model for profit maximization in the plane. *European Journal of Operational Research*, 1274–1287:179, 2007.

[52] J. Fernández, B. Pelegrín, B. Tóth, and F. Plastria. *Avances en localización de servicios y sus aplicaciones*, chapter Localización competitiva en el plano con decisiones en diseño, pages 109–138. Servicio de Publicaciones de la UMU, 2004.

[53] J. Fernández and B. Tóth. Obtaining the efficient set of nonlinear biobjective optimization problems via interval branch-and-bound methods. *Computational Optimization and Applications*. Accepted for publication.

[54] J. Fernández and B. Tóth. Obtaining an outer approximation of the efficient set of nonlinear biobjective problems. *Journal of Global Optimization*, 38:315–331, 2007.

[55] J. Fernández, B. Tóth, L. Cánovas, and B. Pelegrín. Decomposition of a polygon with holes into convex polygons. Submitted for publication.

[56] J. Fernández, B. Tóth, F. Plastria, and B. Pelegrín. Reconciling franchisor and franchisee: A planar biobjective competitive location and design model. In A. Seeger, editor, *Recent Advances in Optimization*, Lectures Notes in Economics and Mathematical Systems 563, pages 375–398. Springer-Verlag, 2006.

[57] J. Figueira, S. Greco, and M. Ehrgott, editors. *Multiple Criteria Decision Analysis: State of the Art Surveys*. Kluwer, New York, 2005.

[58] P. C. Fishburn. Lexicographic orders, utilities and decision rules: A survey. *Management Science*, 20:1442–1471, 1974.

[59] R. W. Floyd. Algorithm 97: Shortest Path. *Communications of the ACM*, 5:345–370, 1962.

[60] R. L. Francis, T. J. Lowe, and A. Tamir. *Facility Location: Application and Theory*, chapter Demand Point Aggregation for Location Models, pages 207–232. Springer, 2002.

[61] R. L. Francis, L. F. McGinnis, and J. A. White. *Facility Layout Location: An Analytical Approach*. Prentice Hall, N. J., 2nd edition edition, 1992.

[62] J. J. Gabszewicz and J. F. Thisse. *Handbook of Game Theory with Economic Applications*, chapter Location, pages 281–304. Elsevier Science Publishers, 1992.

[63] T. Gal and T. Hanne. On the development and future aspects of vector optimization and MCDM. A tutorial. In J. Climaco, editor, *Multicriteria analysis. Proc. of the XIth Int. Conf. on MCDM*, pages 130–145, Berlin, 1997. Springer-Verlag.

[64] O. Gay. Libaa: Une librarie c++ d' affine arithmetic. Technical report, École Polytechnique Fédérale de Lausanne, 2003.

[65] A. Ghosh and C. S. Craig. FRANSYS: a franchise distribution system location model. *Journal of Retailing*, 67:466–495, 1991.

[66] A. Griewank and G. Corliss, editors. *Automatic differentiation of algorithms: Theory, implementation and application*. SIAM, Philadelphia, 1991.

[67] H. Hamacher and S. Nickel. Multicriteria planar location problems. *European Journal of Operational Research*, 94:66–86, 1996.

[68] H. W. Hamacher and S. Nickel. Classification of location models. *Location Science*, 6:229–242, 1998.

[69] R. Hammer, M. Hocks, U. Kulisch, and D. Ratz. *C++ Toolbox for Verified Computing I: Basic Numerical Problems: Theory, Algorithms, and Programs*. Springer-Verlag, Berlin, 1995.

[70] E. Hansen. The centered form. In E. Hansen, editor, *Topics In Interval Analysis*, pages 102–106. Oxford University Press, 1969.

[71] E. Hansen. Global optimization using interval analysis – the one-dimensional case. *Journal of Optimization Theory and Applications*, 29:331–344, 1979.

[72] E. Hansen and R. Smith. Interval arithmetic in matrix computations, part II. *SIAM Journal of Numerical Analysis*, 4:1–9, 1967.

[73] E. Hansen and G. W. Walster. *Global Optimization Using Interval Analysis*. Marcel Dekker, second revised and expanded edition, 2004.

[74] P. Hansen and B. Jaumard. *Handbook of Global Optimization*, chapter Lipschitz optimization, pages 407–494. Kluwer, Dordrecht, 1995.

[75] P. Hansen, B. Jaumard, and H. Tuy. *Facility Location: A Survey of Applications and Methods*, chapter Global Optimization in Location, pages 43–68. Springer Serier in Operations Research and Financial Engineering. Springer-Verlag, Berlin, 1995.

[76] P. Hansen, D. Peeters, D. Richard, and J.-F. Thisse. The minisum and minimax location problems revisited. *Operations Research*, 33:1251–1265, 1985.

[77] P. Hansen, D. Peeters, and J.-F. Thisse. On the location of an obnoxious facility. *Sistemi Urbani*, 3:299–317, 1981.

[78] P. Hansen, J. Thisse, and R. Wendell. Efficient points on a network. *Networks*, 16:357–368, 1986.

[79] P. Hansen and J.-F. Thisse. The generalized Weber-Rawls problem. In *Operational research '81 (Hamburg, 1981)*, pages 569–577. North-Holland, Amsterdam, 1981.

[80] T. Henriksen and K. Madsen. Use of a depth-first strategy in parallel Global Optimization. Technical report, Institute for Numerical Analysis, Technical University of Denmark, 1992.

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

[82] W. Hofschuster and W. Krämer. C-XSC 2.0: A C++ library for extended scientific computing. In R. Alt, A. Frommer, B. Kearfott, and W. Luther, editors, *Numerical Software with Result Verification*, volume 2991 of *Springer Lecture Notes in Computer Science*, pages 15–35. Springer-Verlag, Heidelberg, 2004.

[83] D. L. Huff. Defining and estimating a trading area. *Journal of Marketing*, 28:34–38, 1964.

[84] D. L. Huff. A programmed solution for approximating an optimum retail location. *Land Economics*, 42:293–303, 1966.

[85] K. Ichida and Y. Fujii. Multicriterion optimization using interval analysis. *Computing*, 44:47–57, 1990.

[86] A. K. Jain and V. Mahajan. *Research in Marketing*, chapter Evaluating the competitive environment in retailing using multiplicative competitive interactive models, pages 217–235. JAI Press, 1979.

[87] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer-Verlag, London, 2001.

[88] R. B. Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer, Dordrecht, 1996.

[89] R. B. Kearfott. On proving existence of feasible points in equality constrained optimization problems. *Mathematical Programming*, 83:89–100, 1998.

[90] R. B. Kearfott. Improved and simplified validation of feasible points - inequality and equality constrained problems. Submitted to Mathematical Programming. Available at `http://interval.louisiana.edu/preprints.html`, 2006.

[91] O. Knüppel. PROFIL/BIAS - a fast interval library. *Computing*, 53:277–287, 1994.

[92] R. Krawczyk. Newton-algorithmen zur bestimmung von nullstellen mit fehlerschranken. *Computing*, 4:187–201, 1969.

[93] R. Krawczyk and K. Nickel. The centered form in interval arithmetics: Quadratic convergence and inclusion isotonicity. *Computing*, 28:117–137, 1982.

[94] K. H. Küfer, A. Scherrer, M. Monz, F. Alonso, H. Trinkaus, T. Bortfeld, and C. Thieke. Intensity-modulated radiotherapy - a large scale multi-criteria programming problem. *OR Spectrum*, 25:223–249, 2003.

[95] A. H. Land and A. G. Doig. An automatic method for solving discrete programming problems. *Econometrica*, 28:497–520, 1960.

[96] R. F. Love, J. G. Morris, and G. O. Wesolowsky. *Facilities Location: Models and Methods*. North-Holland, New York, 1988.

[97] M. C. Markót, J. Fernández, L. G. Casado, and T. Csendes. New interval methods for constrained global optimization. *Mathematical Programming Series A*, 106:287–318, 2006.

[98] J. A. Martínez, L. G. Casado, I. García, Y. D. Sergeyev, and B. Tóth. On an efficient use of gradient information for accelerating interval global optimization algorithms. *Numerical Algorithms*, 37:61–69, 2004.

[99] J. A. Martínez, L. G. Casado, I. García, and B. Tóth. AMIGO: Advanced multidimensional interval analysis global optimization algorithm. In *Frontiers in global optimization*, volume 74 of *Nonconvex Optimization and Its Applications*, pages 313–326. Kluwer, Dordrecht, 2004.

[100] R. G. McGarvey and T. M. Cavalier. Constrained location of competitive facilities in the plane. *Computers and Operations Research*, 32:359–378, 2005.

[101] J. A. Mesa, J. Puerto, and A. Tamir. Improved algorithms for several network location problems with equality measures. *Discrete Applied Mathematics*, 130:437–448, 2003.

[102] F. Messine. Extensions of affine arithmetic: application to unconstrained global optimization. *Journal of Universal Computer Science*, 8:992–1015, 2002.

[103] K. S. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer, Boston, 1998.

[104] P. B. Mirchandani and R. L. Francis, editors. *Discrete Location Theory*. Wiley-Interscience, 1990.

[105] R. E. Moore. Automatic error analysis in digital computation. Technical report, Space Div. Report LMSD84821, Lockheed Missiles and Space Co., 1959.

[106] R. E. Moore. *Interval Arithmetic and Automatic Error Analysis in Digital Computing*. PhD thesis, Department of Mathematics, Stanford University, Stanford, California, November 1962. Published as Applied Mathematics and Statistics Laboratories Technical Report No. 25.

[107] R. E. Moore. *Interval Analysis*. Prentice-Hall, New Jersey, 1966.

[108] R. E. Moore and C. T. Yang. Interval analysis I. Technical report, Space Div. Report LMSD285875, Lockheed Missiles and Space Co., 1959.

[109] M. Nakanishi and L. G. Cooper. Parameter estimate for multiplicative interactive choice model: Least square approach. *Journal of Marketing Research*, 11:303–311, 1974.

[110] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, 1990.

[111] A. Neumaier. *Test functions*. World Wide Web, http://www.mat.univie.ac.at/∼vpk/math/funcs.html, 1999.

[112] A. Neumaier. *Acta Numerica 2004*, chapter Complete Search in Continuous Global Optimization and Constraint Satisfaction, pages 271–369. Cambridge University Press, 2004.

[113] S. Nickel, J. Puerto, and A. M. Rodríguez-Chía. *Multiple Criteria Decision Analysis: State of the Art Surveys*, chapter MCDM Location Problems. Springer, Berlin, 2005.

[114] P. H. Peeters and F. Plastria. Discretization results for the Huff and Pareto-Huff competitive location models on networks. *Top*, 6:247–260, 1992.

[115] B. Pelegrín and L. Cánovas. A new assignment rule to improve seed points algorithms for the continuous *k*-center problem. *European Journal of Operational Research*, 104:266–374, 1998.

[116] B. Pelegrín and F. R. Fernández. Determination of efficient points in multiple objective location problems. *Naval Research Logistics*, 35:697–705, 1988.

[117] B. Pelegrín and F. R. Fernández. Determination of efficient solutions for point objective locational decision problems. *Annals of Operations Research*, 18:93–102, 1989.

[118] B. Pelegrín, J. Fernández, and B. Tóth. The 1-center problem in the plane with independent random weights. *Computers and Operations Research*, 2008. DOI 10.1016/j.cor.2006.05.003.

[119] B. Pelegrín, P. Fernández, R. Suárez, and M. D. García. Single facility location on a network under mill and delivered pricing. *IMA Journal of Management Mathematics*, 17:373–385, 2006.

[120] F. Plastria. GBSSS: The generalized big square small square method for planar single-facility location. *European Journal of Operational Research*, 62:163–174, 1992.

[121] F. Plastria. Static competitive facility location: An overview of optimisation approaches. *European Journal of Operational Research*, 129:461–470, 2001.

[122] F. Plastria. Avoiding cannibalization and/or competitor reaction in planar single facility location. *Journal of the Operations Research Society of Japan*, 48:148–157, 2005.

[123] F. Plastria and E. Carrizosa. Optimal location and design of a competitive facility. *Mathematical Programming*, 100:247–265, 2004.

[124] H. Ratschek and J. Rokne. *Computer Methods for the Range of Functions*. Ellis Horwood, Chichester, 1984.

[125] H. Ratschek and J. Rokne. *New Computer Methods for Global Optimization*. Ellis Horwood, Chichester, 1988.

[126] D. Ratz. *Automatic Slope Computation and its Application in Nonsmooth Global Optimization*. Shaker Verlag, Aachen, Germany, 1998.

[127] D. Ratz. A nonsmooth global optimization technique using slopes — the one-dimensional case. *Journal of Global Optimization*, 14:365–393, 1999.

[128] D. Ratz and T. Csendes. On the selection of subdivision directions in interval branch-and-bound methods for global optimization. *Journal of Global Optimization*, 7:183–207, 1995.

[129] J. L. Redondo, J. Fernández, I. García, and P. M. Ortigosa. Solving the multiple competitive facilities location and design problem on the plane. Submitted for publication, 2006.

[130] F. N. Ris. *Interval Analysis and Applications to Linear Algebra*. Ph. D. Thesis, Oxford, 1972.

[131] S. Ruzika and M. M. Wiecek. Approximation methods in multiobjective programming. *Journal of Optimization Theory and Applications*, 126:473–501, 2005.

[132] S. Sayin. Measuring the quality of discrete representations of efficient sets in multiple objective mathemathical programming. *Mathematical Programming*, 87:543–560, 2000.

[133] M. J. Schniederjans and E. Hollcroft. A multi-criteria modeling approach to jury selection. *Socio-Economic Planning Sciences*, 39:81–102, 2005.

[134] D. Serra and C. ReVelle. *Facility Location: A Survey of Applications and Methods*, chapter Competitive Location in Discrete Space, pages 367–386. Springer, 1995.

[135] J. Silverman, R. E. Steuer, and A. W. Whisman. A multi-period, multiple criteria optimization system for manpower planning. *European Journal of Operational Research*, 34:160–170, 1988.

[136] A. J. V. Skriver and K. A. Andersen. The bicriterion semi-obnoxious location (bsl) problem solved by an ε-approximation. *European Journal of Operational Research*, 146:517–528, 2003.

[137] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Applications*. John Wiley & Sons, Inc., 1986.

[138] R. Suarez-Vega, D. R. Santos-Peñate, and P. Dorta-González. Discretization and resolution of the (r/x_p) medianoid problem involving quality criteria. *Top*, 12:111–133, 2004.

[139] T. Sunaga. Theory of an interval algebra and its application to numerical analysis. Technical report, Gaukutsu Bunken Fukeyu-kai, Tokyo, 1958.

[140] H. Tuy, F. Al-Khayyal, and F. Zhou. A d.c. optimization method for single facility location problems. *Journal of Global Optimization*, 7:209–227, 1995.

[141] B. Tóth and L. G. Casado. Multi-dimensional pruning from the Baumann point in an interval global optimization algorithm. *Journal of Global Optimization*, 38:215–236, 2007.

[142] B. Tóth and T. Csendes. Empirical investigation of the convergence speed of inclusion functions. *Reliable Computing*, 11:253–273, 2005.

[143] B. Tóth, J. Fernández, and T. Csendes. Empirical convergence speed of inclusion functions for facility location problems. *Journal of Computational and Applied Mathematics*, 199:384–389, 2007.

[144] B. Tóth, J. Fernández, B. Pelegrín, and F. Plastria. Sequential versus simultaneous approach in the location and design of two new facilities using planar Huff-like models. *Computers and Operations Research*. Accepted for publication.

[145] B. Tóth, J. Fernández, B. Pelegrín, and F. Plastria. Test problems for [144]. available at `http://www.um.es/geloca/gio/testsproblems2fac.zip`.

[146] B. Tóth, F. Plastria, J. Fernández, and B. Pelegrín. Sensitivity analysis of the optimal solutions to Huff-type competitive location and design problems. Submitted for publication.

[147] A. Törn and A. Žilinskas. *Global Optimization*, volume 3350. Springer-Verlag, Berlin, Germany, 1989.

[148] T. Vinkó, J. L. Lagouanelle, and T. Csendes. A new inclusion function for optimization: Kite — the one dimensional case. *Journal of Global Optimization*, 30:435–456, 2004.

[149] Visual G.I.S. Engineering. Basauri 17 (La Florida) 28023 Madrid, 2003.

[150] G. Walster, E. Hansen, and S. Sengupta. Test results for global optimization algorithm. In P. Boggs, R. Byrd, and R. Schnabel, editors, *SIAM Numerical Optimization 1984*, pages 272–287. SIAM, 1985.

[151] A. Weber. *Über den Standort der Industrien 1*. Teil: Reine theorie des standordes, Tübingen, Germany, 1909.

[152] E. Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal*, 43:355–386, 1937.

[153] D. J. White. A bibliography on the applications of mathematical programming multiple-objective methods. *Journal of the Operational Research Society*, 41:669–691, 1990.

[154] P. Xu. A hybrid global optimization method: The multidimensional case. *Journal of Computational and Applied Mathematics*, 155:423–446, 2003.

[155] R. C. Young. The algebra of many-valued quantities. *Mathematische Annalen*, 104:260–290, 1931.

[156] P. L. Yu. *Multiple-criteria decision making concepts, techniques and extensions*. Plenum Press, New York, 1985.

[157] M. Zeleny. *Multiple Criteria Decision Making*. McGraw-Hill, Inc., 1982.

Appendix A

Table A.1: The logarithm of the empirical convergence parameter c for the test functions for some interval sequences.

Problem name	Set name	$\log_{10} c$				
		Natural	Centered	Baumann	Slope For.	Slope Rev.
BR	B&B	-0.904	-0.898	-1.478	-4.635	-4.347
	Opt	-22.132	0.009	2.117	-2.010	-2.112
	Rand	-2.521	0.704	-20.899	-1.341	-1.249
	Up	6.035	-1.560	-1.958	-1.969	-2.183
	All	-2.075	-0.050	-6.166	-3.227	-3.029
GP	B&B	4.708	6.684	6.513	6.404	6.559
	Opt	4.363	7.626	7.419	3.178	3.015
	Rand	5.185	6.464	-8.748	5.696	6.125
	Up	2.680	4.644	3.683	3.509	2.998
Griew5	B&B	-17.372	-0.855	-16.402	-1.042	-1.045
L10	B&B	-19.563	0.032	-0.176	-0.507	-0.536
	Opt	-21.820	-0.103	-0.284	-0.683	-0.637
	Rand	0.497	1.240	-15.008	0.597	0.586
	All	-8.827	0.678	-8.254	0.103	0.092
L11	B&B	-19.349	0.152	-0.040	-0.398	-0.376
L12	B&B	-19.286	0.221	0.045	-0.293	-0.303
L13	B&B	-8.347	1.539	1.418	1.001	1.004
	Opt	-10.498	1.713	1.576	1.144	1.199
	Rand	1.055	2.784	-4.352	3.594	3.543
	All	-2.277	2.397	-2.380	2.822	2.797
L14	B&B	-12.439	1.546	1.407	0.992	1.016
	Opt	-15.293	1.396	1.166	0.761	0.789
	Rand	1.320	3.394	-10.031	3.146	3.159
	Up	7.545	0.837	0.732	-0.902	-0.921
	All	-6.901	2.720	-4.653	2.128	2.147
L15	B&B	-12.395	1.567	1.452	0.999	1.021
	Opt	-15.155	1.427	1.292	0.815	0.859
	Rand	1.242	3.730	-9.712	2.638	2.655
	All	-4.829	2.804	-4.926	1.956	1.979
L16	B&B	-12.926	1.527	1.388	0.996	0.993
	Opt	-16.102	1.403	1.235	0.807	0.829
	Rand	1.197	3.445	-8.837	2.901	2.900

Table A.1: *(continued)* The logarithm of the empirical convergence parameter c for the test functions for some interval sequences.

Problem	Set	$\log_{10} c$				
name	name	Natural	Centered	Baumann	Slope For.	Slope Rev.
	Up	-17.245	1.690	1.400	0.382	0.359
	All	-8.213	2.540	-3.280	1.855	1.855
L18	B&B	-12.827	1.581	1.453	1.022	1.029
L3	B&B	2.078	3.305	3.202	1.218	1.238
	Opt	1.970	-0.261	2.150	-0.633	-0.545
	Rand	-0.406	2.802	-10.990	2.313	2.416
	Up	1.783	3.251	3.195	2.814	2.769
	All	1.707	3.141	1.244	1.295	1.327
L5	B&B	1.951	2.602	2.546	1.330	1.337
	Opt	2.028	3.539	4.723	-0.539	-0.585
	Rand	1.722	2.845	-12.358	1.973	2.402
	Up	2.379	1.879	1.971	1.681	1.723
	All	1.614	2.531	-2.892	1.063	1.208
L8	B&B	-19.890	-0.161	-0.320	-0.647	-0.646
	Opt	-21.929	-0.461	-0.650	-0.922	-0.912
	Rand	0.361	1.039	-18.057	1.192	1.197
	Up	-3.459	-0.049	-0.333	-1.091	-0.614
	All	-8.916	0.703	-9.959	0.415	0.415
L9	B&B	-19.687	-0.073	-0.253	-0.579	-0.582
	Opt	-21.940	-0.259	-0.439	-0.804	-0.805
	Rand	-2.261	0.943	-15.611	0.330	0.337
	All	-9.588	0.515	-9.344	-0.036	-0.034
Ratz4	B&B	-0.338	2.318	0.818	-0.309	1.492
	Opt	-0.410	0.943	0.468	-3.752	-3.854
	Rand	-7.741	-0.797	-15.403	-2.589	-2.654
	Up	0.140	0.602	0.522	0.052	0.041
	All	-6.165	-0.436	-10.184	-2.719	-2.771
RB	B&B	-14.180	2.496	2.082	2.140	2.147
	Opt	-23.624	2.671	2.567	2.180	2.259
	Rand	-15.186	2.864	-18.302	0.774	0.773
	Up	-29.934	2.055	2.022	1.290	1.213
	All	-18.107	3.409	-5.726	1.875	1.883
S5	B&B	-0.953	1.484	1.182	-0.026	0.411
	Opt	-1.047	-1.141	0.789	0.420	0.420
	Rand	-2.279	-1.689	-18.933	-22.630	-22.614
	Up	1.172	2.677	2.232	2.407	2.420
	All	-1.863	-0.176	-7.875	-11.064	-10.963
Sch218	B&B	-0.025	-2.615	-3.145	-3.834	-3.849
	Opt	-0.517	-0.272	-0.318	-1.623	-1.634
	Rand	-12.748	-0.668	-20.612	-3.162	-3.168
	Up	-0.954	-0.446	-0.439	-0.935	-0.835
	All	-3.681	-2.025	-7.967	-3.569	-3.581
Sch21	B&B	0.869	1.850	1.482	1.435	2.006

Table A.1: *(continued)* The logarithm of the empirical convergence parameter c for the test functions for some interval sequences.

Problem	Set	$\log_{10} c$				
name	name	Natural	Centered	Baumann	Slope For.	Slope Rev.
	Opt	0.454	1.857	1.685	1.313	1.879
	Rand	1.578	3.275	-15.431	1.988	3.644
	Up	3.184	-2.018	-2.027	-3.094	-2.750
	All	1.150	3.103	-0.655	2.498	3.432
Sch25	B&B	-8.036	0.877	0.548	0.531	0.532
	Opt	-29.914	0.974	0.799	0.602	0.601
	Rand	-26.431	0.460	-26.354	-0.339	-0.318
	Up	-21.594	0.927	0.822	0.489	0.577
	All	-18.537	0.736	-9.189	0.116	0.117
Sch31	B&B	0.088	1.232	0.990	0.795	0.868
	Opt	-0.004	1.174	0.998	0.711	0.802
	Rand	-8.985	1.852	-19.672	0.385	0.404
	Up	-72.693	-0.170	0.128	-0.165	-0.313
	All	-6.891	1.756	-10.592	0.759	0.809
Sch32	B&B	-7.611	0.948	0.570	0.588	0.559
	Opt	-17.520	0.834	0.600	0.394	0.359
	Rand	-11.403	0.421	-17.429	0.241	0.238
	Up	-23.546	-0.758	-0.410	-1.380	-1.385
	All	-13.510	1.617	-3.991	1.196	1.177
Sch37_{10}	B&B	-29.902	0.081	-1.781	-0.979	-0.870
Sch37_{5}	B&B	-29.877	-0.054	-1.925	-1.171	-1.026
	Opt	-29.887	-0.221	-2.239	-1.260	-0.756
	Rand	-29.900	-5.581	-20.629	-5.973	-5.944
	Up	-29.896	-0.941	-1.374	-1.818	-1.988
	All	-29.895	2.532	-2.707	1.548	1.558
SHCB	B&B	1.261	2.735	1.699	-0.017	-0.175
	Opt	1.078	-0.242	2.793	-2.258	-2.365
	Rand	1.312	1.868	-16.069	1.181	1.050
	Up	-0.333	-1.022	-1.274	-1.863	-1.683
	All	1.986	2.969	-1.954	0.779	0.646
THCB	B&B	1.663	1.684	1.429	1.317	1.263
	Opt	0.939	1.361	1.117	0.968	1.040
	Rand	2.075	1.401	-14.115	0.204	-0.117
	Up	0.624	0.081	-0.504	-0.918	-0.783
	All	2.613	2.941	-2.968	1.954	1.815
All	B&B	1.418	3.790	3.526	2.962	3.038
	Opt	-11.417	0.877	1.115	-0.240	-0.186
	Rand	-3.986	1.655	-15.472	-0.220	-0.091
	Up	-17.725	-0.027	-0.255	-0.808	-0.766
	All	-1.558	3.267	-1.153	2.060	2.165

Appendix B

Existing facilities

Name	Location		Qualities
j	x_1 coord.	x_2 coord.	α_j
E1	3.11	4.05	3.1
E2	5.33	6.19	3.5
E3	8.67	3.33	3.15
C1	5.33	5.71	4.0
C2	4.89	5.48	3.25

Demand points

Name	Location		Buying power	Aggregated		Aggreg.
i	x_1 coord.	x_2 coord.	w_i	w_i	ϕ_{i1}	ϕ_{i1}
Abanilla	6.67	0.00	0.339	0.339	0.806	0.806
Albudeite	0.00	5.00	0.076	0.076	0.527	0.527
Alcantarilla	3.33	6.43	1.966	1.966	1.301	1.301
Alguazas	2.89	4.29	0.409	0.409	0.849	0.849
Archena	1.56	2.62	0.737	0.864	0.998	1.042
Algaida	2.00	2.86	0.127		0.611	
Beniel	7.33	4.05	0.503	0.503	0.899	0.899
Campos del Río	0.67	4.52	0.115	0.115	0.594	0.594
Ceutí	2.22	3.57	0.443	0.443	0.868	0.868
Fortuna	5.11	0.71	0.404	0.404	0.846	0.846
Librilla	0.67	8.57	0.225	0.225	0.718	0.718
Lorquí	2.44	3.33	0.330	0.330	0.800	0.800
Molina	3.33	4.29	2.282	2.720	1.354	1.419
Ribera	3.33	5.00	0.090		0.552	
Romeral	4.00	2.86	0.281		0.765	
Torrealta	3.33	4.76	0.067		0.507	
Murcia	5.11	5.95	10.000	21.226	2.000	2.433
Albatalia	4.67	5.48	0.123		0.605	
Alberca	4.67	7.14	0.557		0.925	
Algezares	5.33	6.90	0.252		0.742	
Aljucer	4.89	6.19	0.395		0.841	
Alquerias	6.67	4.76	0.272		0.758	
Arboleja	4.67	5.71	0.123		0.606	
Beniajan	6.00	6.19	0.505		0.900	
Cabezo de Torres	5.11	4.76	0.586		0.938	
Casillas	5.56	5.24	0.179		0.673	
Cobatillas	6.22	4.52	0.106		0.580	

Demand points

Name	Location		Buying power	Aggregated		Aggreg.
i	x_1 coord.	x_2 coord.	w_i	w_i	ϕ_{i1}	ϕ_{i1}
Corvera	4.44	10.00	0.111		0.588	
Churra	4.89	4.76	0.223		0.717	
Dolores	5.56	5.95	0.269		0.755	
Era Alta	4.67	6.19	0.154		0.645	
Esparragal	5.78	4.52	0.191		0.686	
Garres	5.78	6.43	0.297		0.777	
Guadalupe	4.00	5.48	0.226		0.720	
Javali Nuevo	3.11	5.95	0.188		0.683	
Javali Viejo	3.33	5.71	0.113		0.590	
Llano de Brujas	6.22	5.24	0.260		0.748	
Monteagudo	5.56	5.00	0.210		0.704	
Nonduermas	3.78	6.67	0.127		0.611	
Ñora	3.78	5.71	0.199		0.694	
Palmar	4.44	6.90	0.961		1.073	
Puebla de Soto	3.56	6.43	0.086		0.546	
Puente Tocinos	5.56	5.71	0.752		1.004	
Puntal	4.67	5.00	0.233		0.725	
Raal	6.67	4.52	0.299		0.779	
Ramos	6.67	5.71	0.146		0.636	
Raya	4.44	5.95	0.124		0.607	
Rincon de Seca	4.67	5.95	0.121		0.602	
San Benito	5.33	6.19	0.499		0.897	
San Gines	4.44	6.43	0.110		0.586	
San Jose de la Vega	5.78	5.95	0.180		0.674	
Sangonera la Seca	2.44	6.90	0.246		0.737	
Sangonera la Verde	3.11	7.38	0.457		0.875	
Santa Cruz	6.44	4.76	0.121		0.602	
Santiago y Zaraiche	4.89	5.48	0.201		0.696	
Santo Angel	4.67	7.62	0.271		0.757	
Torreagüera	6.44	5.95	0.345		0.810	
Zarandona	5.11	5.48	0.320		0.793	
Zeneta	7.33	5.24	0.090		0.552	
Santomera	6.44	3.81	0.681	0.681	0.977	0.977
Torres de Cotillas	2.89	4.76	0.938	0.938	1.066	1.066
Villanueva	1.33	2.14	0.089	0.089	0.551	0.551
Benferri	8.22	1.67	0.064	0.064	0.500	0.500
Bigastro	9.56	3.81	0.295	0.295	0.775	0.775
Jacarilla	10.00	3.81	0.090	0.090	0.553	0.553
Orihuela	8.44	3.10	2.978	3.428	1.454	1.509
Arneva	8.67	3.81	0.091		0.555	
Desamparados	8.00	3.57	0.095		0.562	
Aparecida	7.11	3.33	0.106		0.579	
Murada	8.00	0.71	0.157		0.649	
Redován	9.33	2.38	0.326	0.326	0.797	0.797