Multiobjective integer and mixed-integer nonlinear programming: exact approaches and applications

Daniele Patria

Sapienza, University of Rome

Department of Computer, Control
and Management Engineering

PhD program in Automaic Control,
Bioengineering and Operations Research

Thesis advisor: Prof. Giampaolo Liuzzi
Thesis co-advisor: Prof. Marianna De Santis

Academic year: 2024

SAPIENZA
UNIVERSITÀ DI ROMA

Author's email: daniele.patria@uniroma1.it

Advisor's email: liuzzi@diag.uniroma1.it

Co-advisor's email: marianna.desantis@unifi.it

Thesis presented for the degree of Doctor of Philosofy

Contents

Acknowledgments

The journey that led me to this moment was tough but rewarding. I had the chance to live great experiences but I've also found myself struggling. I met a lot of people which enriched me as a researcher and also as a person. I have to thank the ones which made the effort of walking on this trail worth it. First, I thank professor Gabriele Eichfelder and doctor Leo Warnow for welcoming me in Ilmenau and letting me join them in a challenging research project which has taught me a lot. Second, I thank professor Justo Puerto for accepting me as a visiting PhD student in Seville, letting me get out of my comfort zone research-wise and giving me the opportunity to meet a lot of great people. Then I have to convey my big gratitude to my supervisor professor Marianna De Santis. I not only received a thorough guidance from her but I also had the pleasure of working with a kind and understanding person who has also been able to push me towards my limits. It made me more aware of myself in many ways and I am very grateful for this. Thanks Marianna, working with you was a privilege and a pleasure. Last, I want to thank all the supporting people I had around during this journey: my parents Alessandra and Giorgio, my brother Davide, my girlfriend Federica, my friends and my colleagues. Thanks for the very valuable support you provided when I was in need. I am very thankful for having this opportunity and glad for reaching this moment.

Abstract

Multiobjective optimization comes into play in many real world problems, e.g. finance, production planning, biology, medical imaging, transportation. Some of these applications require to include integer variables in the optimization model, in order to give a proper description of the problem. Thus, multiobjective integer and mixed integer optimization comes into play.

This thesis presents new exact approaches and applications in the field of multiobjective integer optimization. First of all, we present new theoretical results about the ε-constraint algorithm for biobjective nonlinear integer programming problems. It can be shown that, when certain assumptions are satisfied, the algorithm is able to retrieve the complete Pareto frontier.

Then, we leverage this theoretical result for building a biobjective model, used for solving the industrial problem that we named "Best Allocation of Resource Size". The problem consists of determining which set of item sizes or volumes to choose in order to carry out the production of certain products, likewise having certain sizes or volumes. This is done by analyzing the trade-off between the number of items chosen and a cost function, consisting of different cost factors, e.g. purchase cost and stock cost. More specifically, the solutions of the biobjective integer optimization problem are retrieved by means of the ε-constrained method and examined under an industrial point of view.

Then, the results concerning the exactness of the ε-constraint method for biobjective problems are extended to the triobjective case. In particular, an exact method able to retrieve all the nondominated points of triobjetive integer nonlinear optimization problems is devised. Such algorithm, named `TrIntOpt`, is then used to solve linear and nonlinear triobjective problems in order to determine sustainable and nutritionally adequate diet plans for elementary school cantines. More specifically, the model determines a five days lunch menu by taking into account constraints regarding nutritional factors and attractiveness, while minimizing the carbon dioxide, water and nitrogen footprint.

Finally, we also propose an algorithm for finding an approximation of the nondominated set of multiobjective mixed-integer convex quadratic optimization problems. The devised branch-and-bound algorithm, named `DEIA-BB`, solves dual formulations for computing linear outer approximations of subproblems at the nodes of the branch-and-bound tree, where the values of certain integer variables are fixed. The pruning conditions defined and a tailored preprocessing phase allow the nodes to be quickly enumerated. In addition to the enclosure of the nondominated set, `DEIA-BB` also returns a superset of the efficient assignments for the integer variables.

Chapter 1

Introduction

Multi-objective mixed-integer programming (MOMIP) is a crucial area in optimization that addresses problems involving multiple, often conflicting objectives, while also handling variables that can be either continuous or discrete/binary. This type of problem formulation is highly relevant in real-world applications where decision-makers must consider several goals simultaneously, such as e.g. cost, quality, and environmental impact, while making choices constrained by discrete elements, such as the number of items to produce or the on/off status of machines.

In manufacturing industries, logistics, energy, and finance, many decisions involve not just continuous variables (e.g., flow rates or time) but also binary or integer decisions (e.g., whether to build a facility or which route to choose). Solving these problems allows organizations to balance trade-offs among competing objectives, leading to more informed, efficient, and sustainable decisions. As a result, MOMIP plays an important role in optimizing processes across various sectors, where decisions must satisfy complex criteria within certain defined constraints.

As we already mentioned, in multiobjective optimization, the aim is to simultaneously optimize two or more objective functions at the same time. These functions are usually conflicting meaning that it is not possible to find a choice for the decision variables where all the functions take their optimal values at the same time. On the contrary, if this happen we can say that the considered problem is not a multiobjective optimization problem. Hence, solving such class of problems requires finding, rather than a single one, a set of solutions which can in general be infinite. Each of said solutions represents a trade-off choice between the objective functions. The choice of values for the variables leading to one of the solutions is called an efficient point and belongs to the so called decision space. On the other hand, the image of an efficient point under the objective functions is called nondominated point and belongs to the so called image (or criterion) space. A nondominated point is such that there exists no other feasible point for the problem having a better value with respect to one of the objectives without that implying the degradation of at least another objective. Given the very challenging structure of MOMIPs, proper approaches for solving this class of problems have been defined.

A special case for MOMIPs is represented by those problems considering only integer variables. This kind of optimization problems are called multiobjective integer optimization problems (MOIPs). In MOIPs, both the nondominated and efficient sets are represented by isolated points in the image space and decision space respectively.

1.1 Structure of the thesis

This thesis proposes algorithms for solving multiobjective integer nonlinear optimization problems and related applications. We also present a method for computing an approximation of the nondominated set of multiobjective mixed-integer nonlinear optimization problems.

In Section 1.2, we present an overview of the existing literature. In particular, in Section 1.2.1 we review the literature concerning general multiobjective optimization. Then, in Section 1.2.2 we provide an overview of the literature about multiobjective mixed-integer linear problems, and in the following Section 1.2.3 we go through literature covering multiobjective mixed-integer nonlinear optimization problems. Finally, in Section 1.2.4 we provide a review on methods for both linear and nonlinear multiobjective integer optimization problems.

In Chapter 2, we introduce basic concepts and notions for multiobjective optimization problems. In particular, we present a general formulation of multiobjective mixed-integer and multiobjective integer optimization problems. Also, we define the optimality notions for this class of problems, i.e. the definition of (weakly) nondominated points in the image or criterion space and (weakly) efficient points in the decision space.

Chapter 3 describes how, under specific assumptions, the ε-constraint method can be used in order to retrieve the complete Pareto front of biobjective integer optimizations problems. It will be shown that the required assumptions allow the problem to consider nonlinear objectives. Also, a comparison of the devised method with another suitable algorithm is carried out over portfolio optimization instances.

In Chapter 4, we present an application taking advantage of what is shown in the previous chapter. Namely, we formulate a biobjective integer optimization problem for solving the Best Allocation of Resource Sizes (BARS) problem arising in certain industrial scenarios.

Chapter 5 devises a new algorithm for solving triobjective nonlinear integer optimization problems. The devised method extends the ideas presented in [29] and in the second chapter of this thesis. This algorithm, named `TrIntOpt`, is then employed for designing sustainable diet plans by taking into account three different environmental impact factors, namely water, nitrogen and carbon dioxide footprints.

In Chapter 6, we describe a new algorithm for finding an approximation of the nondominated set of multiobjecive mixed-integer optimization problems with strongly convex functions and linear constraints. The branch-and-bound method is proven to require a finite number of iterations in order to retrieve not only an enclosure of the nondominated set, but also a superset of the efficient integer assignments of the problem. In the special case where the problem only has integer variables, the algorithm is able to retrieve all the efficient and nondominated solutions.

Chapter 7, finally, collects the conclusions about this thesis and describes some possible future developments for the presented work.

1.2 Literature review

In this section we provide a review of the existing literature about multiobjetive mixed-integer and integer optimization problems. In particular, we report contributions about multiobjective problems, then we review the main results about multiobjecive mixed-integer (linear and nonlinear) problems and multiobjective purely integer (linear and nonlinear) problems.

1.2.1 Multiobjective optimization

Multiobjective optimization is employed to deal with several real world problems such as drug design, flight scheduling, design of water networks, finance [66, 85, 80, 84]. This wide scope

of application clearly shows its scientific relevance. The objective functions in multiobjective optimization problems are usually conflicting, meaning that no objective can be improved without that implying the deterioration of at least another objective. Therefore, new optimality notions must be defined: a solution in the decision space, namely the space of the decision variables, is called an *efficient point* while its corresponding image from the criterion space (or image space equivalently), namely the images of points in the decision space under the objective functions, is called nondominated point. Every such point represents a trade-off solution among the objectives (criterions equivalently). This concepts will be formally introduced in Chapter 2. Given that, the solution of a multiobjective optimization problem (MOP) is a set of points which can in general be infinite. For a wide and comprehensive introduction on multiobjective optimization we refer to [36].

Since finding an infinite or very high number of points from the solution set of a MOP is not always doable in practice, obtaining an approximation of such set becomes more appealing. However, there are cases for which retrieving an exact description of such set is viable as for the case of problems having linear objective functions and constraints. Indeed, the structure of the nondominated set of a multiobjective linear optimization problem (MOLP) gives the opportunity to efficiently obtain an exact parametric description of the solution set. In 1973 Evans and Steuer [46] proposed a revised version of the simplex algorithm, able to compute the efficient extreme points of a MOLP by employing a necessary and sufficient condition for a point in the decision space to be efficient. In 1991 Armand and Malivert [3] published a paper which also concerns a simplex based algorithm for MOLPs able to retrieve the set of efficient points. By the way, such algorithm is not able to compute the efficient faces of the problem's polyhedron as efficiently as done by the method proposed by Sayin in 1995 [73]. In this work, the author devised an algorithm able to avoid the enumeration of all the efficient extreme points by directly locating the efficient faces instead. More recently, in 2017, Rudloff and coauthors [70] proposed an improved version of the algorithm proposed by Armand and Malivert, aiming at finding only those efficient extreme points involved in the definition of efficient faces. We refer to [61] for an in depth introduction on the topic of multiobjective linear optimization.

When the objective functions or the constraints of a MOP are nonlinear, due to the many possible hurdles arising in this setting, as already said an adequately precise approximation of the nondominated set becomes the goal. A wide review on the topic of approximation methods for MOPs is given in [71] and a more recent one in [39] with the latter also covering mixed-integer MOPs. One way of obtaining an approximation of the set of solutions of a MOP is computing a subset of the nondominated set. This can be done by using a scalarization method. In this case, parameter-depending single-objective problems are built starting from the original MOP. Varying the parameters results in the retrieval of different nondominated or efficient points of the original problem. We can cite [37, 55, 18] among the works presenting approximating algorithms based on scalarization methods. For some of this algorithms, it can be proved that for any efficient point of the MOP there exists a choice of the parameters for building a scalarized single-objective problem having that efficient point as optimal solution. Another class of methods for MOPs is evolutionary algorithms. The stochastic nature of this methods usually implies that no guarantees on the quality of the obtained approximation can be given. We refer to [86] for a survey on the topic.

1.2.2 Multiobjective mixed-integer linear optimization

Adding integer variables to a MOP makes the task of finding efficient or nondominated points harder. MOPs including integer variables are called Multiobjective Mixed-Integer Optimization problems (MOMIP). Solving MOMIPs require finding integer value assignments for the integer

variables leading to efficient solutions of the problem including also continuous variables. When MOMIPs have linear objective functions and constraints, we are in the setting of Multiobjective Mixed-Integer Linear Optimization Problems (MOMILP). The nondominated set of such problems, just like linear MOPs, has a particular structure which makes computing the exact set of solutions viable. MOMIPs are the first class of multiobjective mixed-integer problems that has been studied. The first work on this topic was proposed by Mavrotas and Diakoulaki in 1998 [64] and then enhanced in 2005 by the same authors [65]. The papers show an algorithm for solving binary MOMILPs following a branch-and-bound procedure. Also Belotti et al. [5] proposed an algorithm employing the branch-and-bound paradigm. The developed method is specifically tailored for biobjective mixed-integer linear optimization problems (BOMILP). The nondominated set of BOMILPs is made of isolated points and line segments. The algorithm makes use of the parametric simplex method and a fathoming rule able to eliminate the nodes of the branching tree guaranteed not to contain nondominated points. It initializes a set of feasible points which is iteratively updated, i.e. dominated points are removed and new points are added to the set, making it evolve toward the nondominated set of the problem and hence detecting all of it. However, branch-and-bound is not the only applicable approach for BOILPs. Indeed, Soylu and Yıldız [77] developed an image space algorithm for BOILPs making use of the so called Tabu constraints, employed for excluding already explored regions of the decision space. The devised method is able to find, at each iteration, a new nondominated line segment or isolated point forming the nondominated set. Then, it terminates when no new nondominated point or line segments are found, returning the complete nondominated set of the BOMILP. For a comprehensive review on the topic of MOLMIPs see [52, Section 4].

1.2.3 Multiobjective mixed-integer nonlinear optimization

Focus on multiobjective mixed-integer nonlinear optimization problems (MOMINLP) only began in the last decade. In 2021, Burachik et al. [17] extend one of the algorithms presented in [18] by the same authors. Said algorithm is based on scalarization, as well as the one devised by Ceyhan and coauthors [22] and the one from Doğan and coauthors [34]. For all the mentioned approaches, though, the single-objective problems obtained by scalarization are still mixed-integer nonlinear problems. Hence, the applicability of these algorithms rely on the availability of solvers for such problems. We refer to [4] for an extensive survey on approaches for solving single-objective mixed integer nonlinear optimization problems. The methods for solving MOMINLPs we have mentioned so far do not leverage the mixed-integer nature of such problems. On the other hand, the work from De Santis et al. presented in [28], which is a branch-and-bound based method for solving convex MOMINLPs, takes into account the integer variables for building a branching tree. Furthermore, the proposed method is able to detect both the efficient and the nondominated set of the problem up to a prescribed precision. The authors underline that the devised algorithm is the first non-scalarization-based deterministic algorithm for convex MOMINLPs and so it opened a new research direction in this field. By the way, this approach also employs scalarization for computing linear outer approximations by solving single-objective nonlinear convex problems. Cabrera-Guerrero et al. [19] proposed an algorithm for biobjective mixed-integer nonlinear convex problems, motivated by an application in the medical field. The method assumes that a set of feasible assignments for the integer variables is available and exploits it to apply a decomposition strategy where sub-problems taking into account said assignments are created. As the authors state, the performance of the algorithm decreases as the set of feasible assignments increases. The knowledge about the specific medical application studied in the paper helps excluding integer assignment values that are unlikely to lead to optimal solutions. In the end, the devised algorithm returns a subset of the nondominated set. In 2022, Diessel [33] also proposed an algorithm for

MOMINLPs with affine linear objective functions and convex constraints. The aim is to devise a method able to obtain an approximation of the nondominated set having a certain quality. This is done by computing an inner approximation of the nondominated set. Eichfelder et al. [41] propose the first solver for multiobjective mixed-integer nonconvex optimization problems with performance guarantee. It is based on a branch-and-bound procedure which is made more efficient by introducing two different types of cuts in the image space. Also, it makes use of polyhedral lower bound sets computed by means of continuous convex relaxations at each node of the branching tree. The method terminates by delivering an enclosure of the nondominated set. The definition of enclosure is given in Chapter 6 of this thesis. In [45] Eichfelder and Warnow devise a decomposition method for multiobjective mixed-integer convex optimization problems (MOMICP) named HyPaD. In particular, it decomposes the MOMICP into several multi-objective continuous convex problems, referred as patches, obtained by fixing the integer variables to certain values. It entirely works in the image space and thus it is able to handle problems with a high number of variables namely a decision space having an high dimension. In the proposed method, coverages of such patches are computed and improved. The algorithm is able to deliver a box enclosure of the nondominated set having a prescribed quality and within a finite number of iterations. In [45], the last two mentioned approaches are compared to each other, along with the algorithm from [28]. The results showed that HyPaD is the fastest among the three approaches on the considered test instances.

1.2.4 Multiobjective integer optimization

When all the variables from a multiobjective optimization problem are constrained to assume integer values, we call the problem a multiobjective integer optimization problem (MOIP). Among the many possible approaches for solving this class of problems we can find the ε-constraint method. It was first introduced by Haimes et al. [51] within a work about system optimization and identification. The method consists in finding weakly efficient and corresponding weakly nondominated solutions of the multiobjective problem by solving a single-objective problem derived from the original one. In particular, in the biobjective case one will solve a problem having one of the two criteria as objective function and where the remaining one will be used to build a constraint. Such constraint will require the chosen criterion to be less than or equal to a certain value which is typically denoted as ε. This can also be applied to problems with a generic number of criteria and so all the objective functions from the original problem but one need to be moved to the constraints. In 2014, Kirlik and Sayin [56] proposed a method, based on the ε-constraint, able to detect all the nondominated points of MOIPs with an arbitrary number of criteria. The devised algorithm employs a two-stage strategy for solving single-objective problems derived from the MOIP, in order to avoid the detection of weakly nondominated points. Furthermore, a partitioning mechanism is employed for searching the $(p-1)$ dimensional boxes in the image space, where p denotes the number of objectives, built by means of the ε-constraints. The numerical results show that the algorithm is rather efficient compared to the ones proposed up to then. In 2020, De Santis et al. [32] proposed an algorithm for biobjective integer optimization problems (BOIP) making use of both a special weighted sum scalarization and constraints on one of the two objective functions, in a ε-constraint manner. The authors prove that the method is able to detect the whole Pareto frontier, namely the nondominated set of a biobjective optimization problem, by solving a number of single-objective integer programs equal to the number of points in the front plus two. The method is also applicable to BOIPs with nonlinear objective functions, as long as there exists an oracle for solving the single-objective integer problems built and as long as there exists a lower bound for the distance of points in the image space. In Chapter 3 we use the notions introduced in [32] to obtain correctness results for the ε-constrained method.

Furthermore, the performance of the algorithm from De Santis and coauthors and the ε-epsilon constraint are compared.

Another common approach for retrieving all the nondominated solutions of MOIPs is decomposing the image space in order to obtain regions of interest where to look for new nondominated points. Boland, Charkhgard and Savelsbergh proposed different methods for solving MOIPs employing this technique. In particular, an algorithm named "the L-Shape Method" [9] is proposed for solving triobjective integer optimization problems (TOIP). The method uses rectangles and L-shaped regions in the projected image space, i.e. the image space taking into account two of the three objective functions, for finding nondominated points. The authors are also able to provide lower and upper bounds for the number of single-objective integer problems to be solved. More details about the algorithm will be given in Chapter 5. Another method for solving MOIPs is presented by the same group of authors in [10]. The so called "Quadrant Shrinking Method" is also tailored for TOIPs and works in the projected bi-dimensional image space, as the L-Shape Method does. The algorithm uses a two-stage scalarization in order to detect new nondominated points and avoiding detecting weakly nondominated ones. In addition, it explores quadrants in the projected image space defined by proper upper bounds in the projected image space. Also for this algorithm, the authors are able to bound the number of single-objective integer problems to be solved in order to find the complete set of nondominated points, that is $3|\mathcal{Y}_N| + 1$ where $\mathcal{Y}_N$ denotes the nondominated set. Both [9] and [10] are designed for linear TOIPs. Klamroth et al. propose in [57] a way of characterizing the search region in multiobjective problems, namely the region of the image space that could still include not-yet-found nondominated points. This idea is employed in [79], where an efficient algorithm for solving MOIPs is devised. The proposed method considers both full-dimensional and projected search zones in the image space. Also, it is able to always start from a feasible solution when performing the exploration of a search zone, speeding up the computation. The aim of the authors is to reduce as much as possible the number and the complexity of the subproblems that are iteratively solved in order to explore the search regions and find nondominated points of the MOIP. This is done by keeping an exact characterization of the search region throughout the algorithm and by implementing strategies to prove the emptiness of some search zones without the need of solving an integer problem. The performances of the algorithm are compared to the approach from [56] and from [11] showing that the devised method is the most efficient among the three. The method proposed by Tamby and Vanderpooten does not require the MOIP to have linear objective functions. In 2024, Dächert and coauthors [26] devise a versatile algorithm for linear MOIPs, employing the ideas from [57] for characterizing the search region, along with efficient update procedures for the search region itself. Also, an upper bound for the number of iterations needed by the proposed method to retrieve the complete set of solutions is given. The proposed algorithm is considered versatile due to its compatibility with different types of scalarization methods. By the way, the authors focus on ε-constraint scalarization method, as it allows to reduce the number of search regions to be explored. The algorithm is implement in `C++` programming language and compared to other methods chosen as representatives of different possible approaches for solving MOIPs and which are implemented in the same programming language, allowing for a fair comparison. Numerical results shows that the devised method is the fastest among the compared ones. We refer to [52, Section 3] for an in-depth review on algorithms for finding the complete set of solutions of multiobjective integer linear optimization problems.

Chapter 2

Basic definitions for multiobjective integer optimization problems

In this chapter we define the optimization problems this thesis focuses on. Specifically, in Section 2.1, we describe both the multiobjective mixed-integer optimization problem (MOMIP) and the multiobjective integer optimization problem (MOIP). Then, in Section 2.2, we present the key optimality concepts relevant to multiobjective optimization, which will serve as foundational references throughout the remainder of this thesis.

2.1 Multiobjective mixed integer and integer problems

A MOMIP is defined as

$$\begin{aligned} \min \quad & (f_1(x), \dots, f_m(x))^T \\ \text{s.t.} \quad & x \in \mathcal{X} \subseteq \mathbb{Z}^k \times \mathbb{R}^{n-k}, \end{aligned} \tag{MOMIP}$$

with $k \leq n$, where $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \dots, m$ are continuous functions and $\mathcal{X} \subseteq \mathbb{Z}^k \times \mathbb{R}^{n-k}$ denotes the feasible set of the problem. We refer to $\mathcal{X}$ as *decision space* or *preimage space* and to $f(\mathcal{X}) := \mathcal{Y} \subseteq \mathbb{R}^m$ as *image space* or *criterion space*. In Chapter 6 we propose an algorithm for solving problem (MOMIP) under the assumption that all the objective functions to be strongly convex quadratic functions. Furthermore, we will take into account a feasible set defined by means of linear inequalities, namely $\mathcal{X} = \{x \in \mathbb{Z}^k \times \mathbb{R}^{n-k} : Ax \leq b\}$.

In case $k = n$, namely all the decision variables are constrained to take integer values, we are in the context of multiobjective integer programming. We can define a MOIP as follows:

$$\begin{aligned} \min \quad & (f_1(x), \dots, f_m(x))^T \\ \text{s.t.} \quad & x \in \mathcal{X} \cap \mathbb{Z}^n, \end{aligned} \tag{MOIP}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$, $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \dots, m$ are continuous functions and $\mathcal{X} \cap \mathbb{Z}^n$ denotes the feasible set. For (MOIP) also, we refer to $\mathcal{X} \cap \mathbb{Z}^n$ as *decision space* or *preimage space* and to $f(\mathcal{X} \cap \mathbb{Z}^n) := \mathcal{Y}$ as *image space* or *criterion space*.

2.2 Optimality notions for multiobjective problems

A multiobjective optimization problem usually considers conflicting objective functions, meaning that there exists no feasible point minimizing all the objectives at the same time. This leads to the fact that there is no such thing as a unique optimal vector of values $f^* = (f_1^*, \dots, f_m^*)$ for the problem. Hence, the solution of a multiobjective optimization problem is a set of points. We now give the definitions of *efficient point* and *nondominated point* along with the ones of *weakly efficient point* and *weakly nondominated point*. Since these notions are not strictly related to a specific class of multiobjective optimization problems, e.g. MOMIPs or MOIPs, we will consider the following generic multiobjective optimization problem (MOP) as a reference:

$$\begin{aligned} \min \quad & (f_1(x), \dots, f_m(x))^T \\ \text{s.t.} \quad & x \in \mathcal{X} \subseteq \mathbb{R}^n. \end{aligned} \tag{MOP}$$

Definition 2.2.1. *Let $\bar{x} \in \mathcal{X}$ be a feasible point of (MOP). We call $\bar{x}$ an efficient point for (MOP) if there exists no $x \in \mathcal{X}$ such that $f_i(x) \leq f_i(\bar{x})$ for $i = 1, \dots, m$, with $f(\bar{x}) \neq f(x)$. We call the image of an efficient point $f(\bar{x}) \in \mathcal{Y}$ a nondominated point for (MOP).*

Definition 2.2.2. *Let $\bar{x} \in \mathcal{X}$ be a feasible point of (MOP). We call $\bar{x}$ a weakly efficient point for (MOP) if there exists no $x \in \mathcal{X}$ such that $f_i(x) < f_i(\bar{x})$ for $i = 1, \dots, m$. We call the image of a weakly efficient point $f(\bar{x}) \in \mathcal{Y}$ a weakly nondominated point for (MOP).*

We refer to the set of efficient points and weakly efficient points as $\mathcal{E} \subseteq \mathbb{R}^n$ and $\mathcal{E}_w \subseteq \mathbb{R}^n$ respectively. We call them *efficient set* and *weakly efficient set*. We refer to the set of nondominated points and weakly nondominated points as $\mathcal{Y}_N \subseteq \mathbb{R}^m$ and $\mathcal{Y}_{wN} \subseteq \mathbb{R}^m$ respectively. We now define the notion of dominance among points in the image space of (MOP).

Definition 2.2.3. *Let $y^1, y^2 \in \mathcal{Y}$. We say that y^1 dominates y^2 if $y_i^1 < y_i^2$ for $i = 1, \dots, m$.*

Definition 2.2.4. *Let $y^1, y^2 \in \mathcal{Y}$. We say that y^1 weakly dominates y^2 if $y_i^1 \leq y_i^2$ for $i = 1, \dots, m$, with $y^1 \neq y^2$.*

We also introduce the definitions of ε-efficient point and ε-nondominated point since it will be recalled in Chapter 6.

Definition 2.2.5. *Let $\bar{x} \in \mathcal{X}$ be a feasible point of (MOP) and let $\varepsilon > 0$. We call $\bar{x}$ an ε-efficient point for (MOP) if there exists no $x \in \mathcal{X}$ such that $f_i(x) \leq f(\bar{x}) - \varepsilon e$ for $i = 1, \dots, m$, with $f(x) \neq f(\bar{x}) - \varepsilon e$. We call the image of an ε-efficient point $f(\bar{x}) \in \mathcal{Y}$ an ε-nondominated point.*

We refer to the set of ε-efficient points as $\mathcal{E}_\varepsilon \subseteq \mathbb{R}^n$ and we call it *ε-efficient set*. We refer to the set of ε-nondominated points as $\mathcal{Y}_{\varepsilon N} \subseteq \mathbb{R}^m$. Furthermore, we introduce the definition of stable set.

Definition 2.2.6. *$N \subseteq \mathbb{R}^m$ is a stable set if there exists no couple of points $y^1, y^2 \in N$ such that $y^1 = y^2$ or y^1 dominates y^2.*

Figures 2.1 and 2.2 help visualizing the definitions given in this chapter. Figure 2.1 depicts the image set of a (MOMIP). The image space is the union of the four outlined sets. The nondominated set of this problem is represented by the red lines shown in the figure. Also, we can see a feasible point $y \in f(\mathcal{X})$ being dominated by a point $\bar{y} \in \mathcal{Y}_N$. Figure 2.2 depicts the image set of a (MOIP). The image set is represented by isolated points. The nondominated set $\mathcal{Y}_N$ is made of the points highlighted in red. The feasible point $y \in f(\mathcal{X} \cap \mathbb{Z}^n)$ is dominated by the point $\bar{y} \in \mathcal{Y}_N$.

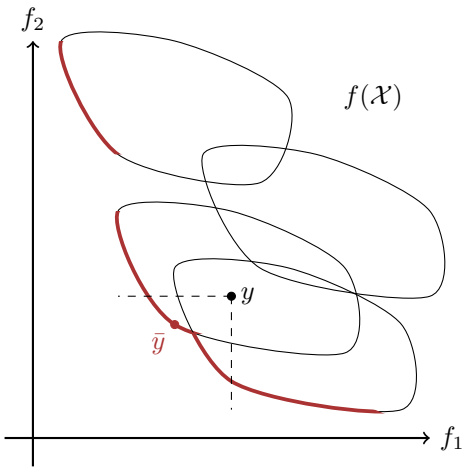

Figure 2.1: Example of the nondominated set of a (MOMIP) with $m = 2$ objective functions.

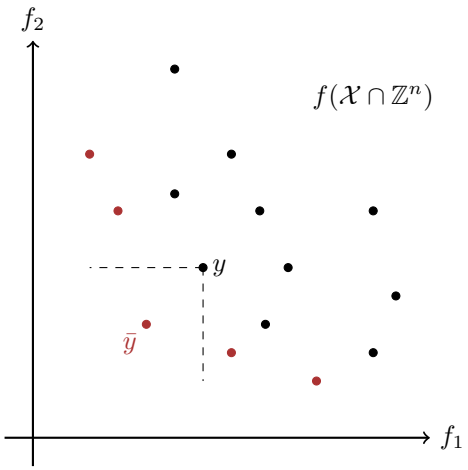

Figure 2.2: Example of the nondominated set of a (MOIP) with $m = 2$ objective functions.

Chapter 3

Exactness of the ε-constraint method for biobjectve nonlinear integer programming *

In this chapter, we focus on biobjective nonlinear integer programming problems of the following form

$$\begin{array}{ll} \min & (f_1(x), f_2(x))^T \\ \text{s.t.} & x \in \mathcal{X} \cap \mathbb{Z}^n, \end{array} \tag{BOIP}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ and $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ are continuous functions. We aim to solve (BOIP) exactly in the sense that we aim to detect its complete set of non-dominated points, denoted as $\mathcal{Y}_N$. More specifically, we will extend the ideas developed in [32] to define an exact criterion space algorithm based on the ε-constraint scalarization technique. The ε-constraint method produces single-objective subproblems adding further constraints to the original feasible set. Namely, given (BOIP), the ε-constraint method minimizes single-objective optimization problems of the following form:

$$\begin{array}{ll} \min & f_2(x) \\ \text{s.t.} & f_1(x) \leq \varepsilon \\ & x \in \mathcal{X} \cap \mathbb{Z}^n. \end{array} \tag{P_ε}$$

The parameter ε varies between $\min\{f_1(x) \mid x \in \mathcal{X} \cap \mathbb{Z}^n\}$ and $f_1(\hat{x}) - \delta$ with $\hat{x} \in \operatorname{argmin}\{f_2(x) \mid x \in \mathcal{X} \cap \mathbb{Z}^n\}$. Thereby, δ is a positive step size which influences the decrease of the upper bounds ε. As it will be clarified in the next section, the role of the step size δ is crucial to detect the complete non-dominated set of a (BOIP). The role of f_1 and f_2 in the definition of Problem (P_ε) can be interchanged. We refer to [23, 38, 59] for some examples on contributions on the ε-constraint method.

This chapter is organized as follows. In Subsection 3.1, we recall the γ-positivity assumption introduced in [32] and we establish our main result. Under specific assumptions, we prove that the ε-constraint method is able to detect the complete Pareto front of a (BOIP) after having addressed a finite number of single-objective problems. In section 3.2, we report our numerical experience. We compare the performance of FPA* [32], which is an improved version of FPA, with the ε-constraint method devised on portfolio optimization problems.

*The content of this chapter is part of the paper [30].

3.1 The γ-positivity assumption and the exactness of the ε-constraint method

The scheme of the ε-constraint method applied to (BOIP) is reported in Algorithm 1. As already mentioned, at every iteration k of the algorithm, the following single-objective integer subproblem needs to be addressed

$$\begin{array}{ll} \min & f_2(x) \\ \text{s.t.} & f_1(x) \leq \varepsilon^k \\ & x \in \mathcal{X} \cap \mathbb{Z}^n, \end{array} \tag{P_ε^k}$$

with $\varepsilon^k \in \mathbb{R}$ properly set. Recall that by [36, Proposition 4.3] any optimal solution of (P_ε^k) is a weakly efficient solution of (BOIP). Moreover, by [36, Theorem 4.5], for any efficient point x^* of (BOIP) there exists an upper bound $\varepsilon^k \in \mathbb{R}$ such that x^* is an optimal solution of (P_ε^k). However, we cannot solve (P_ε^k) for an infinite number of upper bounds ε^k. Thus, the question is whether and how we can find a finite number of upper bounds for which we have to solve (P_ε^k) such that we can still find all non-dominated points of (BOIP). In the following we discuss assumptions and an algorithm which guarantees such a finiteness result.

As a first assumption, we ask for the availability of a solver for (P_ε^k).

Assumption 3.1.1. *There exists an oracle that either returns an optimal solution of* (P_ε^k) *or certifies its infeasibility for any choice of* $\varepsilon^k \in \mathbb{R}$.

Note that there exists a number of solvers able to deal with single-objective nonlinear integer programming problems such as, e.g., `BARON` [72] or `SCIP` [49], so that Assumption 3.1.1 holds for many classes of (BOIP). In our computational experience, we will consider BOIPs having quadratic objective functions and a polyhedral set $\mathcal{X}$ and we will use `GUROBI` [50] as a solver.

Algorithm 1 Scheme of the ε-constraint method.

Input: (BOIP), $\delta > 0$, $k = 1$;
Output: the Pareto front $\mathcal{Y}_N$ of (BOIP);

Compute $x^* \in \operatorname{argmin}_{x \in \mathcal{X} \cap \mathbb{Z}^n} f_1(x)$
Compute $\hat{x} \in \operatorname{argmin}_{x \in \mathcal{X} \cap \mathbb{Z}^n} f_2(x)$
Set $\mathcal{M} = \{f(\hat{x})\}$
Set $\varepsilon^1 = f_1(\hat{x}) - \delta$
while $\varepsilon^k \geq f_1(x^*)$ **do**
    Compute $x^k \in \operatorname{argmin}_{x \in \mathcal{X}^k \cap \mathbb{Z}^n} f_2(x)$, with $\mathcal{X}^k = \mathcal{X} \cap \{x \in \mathbb{R}^n : f_1(x) \leq \varepsilon^k\}$
    Set $\mathcal{M} = \{f(x^k)\} \cup \mathcal{M}$
    Set $\varepsilon^{k+1} = f_1(x^k) - \delta$
    Set $k = k + 1$
end while
Apply a filtering procedure to $\mathcal{M}$ to obtain $\mathcal{Y}_N$
Return $\mathcal{Y}_N$

In the definition of FPA (and FPA*) in [32] some basic assumptions on Problem (BOIP) had to be made. Here we make the same assumptions, reported in the following.

Assumption 3.1.2 (Existence of the ideal point)**.** *We assume that the ideal objective values* $f_i^{\mathrm{id}} := \min_{\mathcal{X} \cap \mathbb{Z}^n} f_i(x)$, $i = 1, 2$, *and thus the ideal point* $f^{\mathrm{id}} := (f_1^{\mathrm{id}}, f_2^{\mathrm{id}}) \in \mathbb{R}^2$, *exists.*

The crucial assumption that we make in order to prove the exactness of the ε-constraint method is the so called *positive gap value* assumption. We need to assume that a positive

value exists that underestimates the distance between the image of two integer feasible points of (BOIP), componentwise.

Definition 3.1.3 (Positive γ-function)**.** *Let $\gamma > 0$. A function $g : \mathcal{X} \to \mathbb{R}$ is a positive γ-function over $\mathcal{X} \cap \mathbb{Z}^n$ if it holds $|g(x) - g(z)| \geq \gamma$ for all $x, z \in \mathcal{X} \cap \mathbb{Z}^n$ with $g(x) \neq g(z)$.*

Assumption 3.1.4. *The functions $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, 2$ in Problem (BOIP) are positive γ-functions as in Definition 3.1.3 for some $\gamma > 0$.*

Assumptions 3.1.2 and 3.1.4 imply that the non-dominated set $\mathcal{Y}_N$ of (BOIP) is finite (see Proposition 2.7 in [32]). Thus, we know that there exists a finite number of upper bounds ε^k for which we have to solve (P_ε^k) to find the complete Pareto front. The question is how to find this parameter set, and Algorithm 1 proposes an answer for that. Note that there exist a number of classes of functions that easily satisfies Assumption 3.1.1 and Assumption 3.1.4, such as linear or quadratic functions defined over $\mathbb{Q}^n$, or polynomials with rational coefficients over $\mathbb{Z}^n$ as long as they have no roots in $\mathbb{Z}^n$. Table 1, in Section 4.3 in [32], shows some classes of functions for which Assumption 3.1.4 holds and reports how to compute γ. The following example shows a case where only Assumption 3.1.2 holds, while Assumption 3.1.4 does not.

Example 3.1.5. *Let $\mathcal{X} \cap \mathbb{Z} = \{x \in \mathbb{Z} \mid x \geq 0\}$, $f_1(x) = \arctan(x)$ and $f_2(x) = (x-1)^2$. We have that $f^{\mathrm{id}} = (0, 0)$, so that Assumption 3.1.2 is verified. However, it is not possible to find $\gamma > 0, \gamma \in \mathbb{R}$ such that $|\arctan(x) - \arctan(y)| \geq \gamma$ for all $x, y \in \mathcal{X} \cap \mathbb{Z}$ with $\arctan(x) \neq \arctan(y)$. Therefore, Assumption 3.1.4 does not hold.*

We further want to underline that it is common to assume $\mathcal{X}$ to be bounded so that $\mathcal{X} \cap \mathbb{Z}^n$ and $f(\mathcal{X} \cap \mathbb{Z}^n)$ are finite sets. In this case, Assumption 3.1.4 trivially holds.

Remark 3.1.6. *In the context of the ε-constrained method, Assumption 3.1.4 can be relaxed, requiring that only the function that is moved to the constraints (that is f_1 in (P_ε^k)) is a positive γ-function. Indeed, in all the proofs presented here, we need the positive γ-function property only for f_1. Moreover, since there are no two different non-dominated points with the same first component, the proof in [32, Proposition 2.7] can be adapted to show that the non-dominated set $\mathcal{Y}_N$ of (BOIP) is still finite.*

Let Assumption 3.1.4 hold for f_1 and f_2 with $\gamma > 0$. Let $\delta > 0$ be the input parameter for Algorithm 1. Two different scenarios occur:

- $\delta > \gamma$: in this case the ε-constraint method could miss some points of the Pareto front $\mathcal{Y}_N$, since the step size δ may be wider than the minimum distance between two non-dominated points;
- $\delta \leq \gamma$: the ε-constraint algorithm is able to detect the complete Pareto front $\mathcal{Y}_N$, as shown in the following.

As already mentioned, according to [36, Proposition 4.3], any optimal solution of (P_ε^k) is a weakly efficient solution of (BOIP). For $\varepsilon^0 := f_1(\hat{x})$ the point $\hat{x}$ is an optimal solution of (P_ε^0) by construction and thus weakly efficient. Note that we cannot prove that a solution of (P_ε^k) is efficient, since a point $\tilde{x} \in \mathcal{X}^k \cap \mathbb{Z}^n$ may exist such that $f_1(\tilde{x}) < f_1(x^k)$ and $f_2(\tilde{x}) = f_2(x^k)$. Hence, before the filtering step, the set

$$\mathcal{M} = \{f(x^{k-1}), f(x^{k-2}), \ldots, f(x^1), f(\hat{x})\}$$

from Algorithm 1 may contain points which are just weakly non-dominated without being non-dominated. We will show in the next lemmata that it holds $\mathcal{Y}_N \subseteq \mathcal{M}$. The filtering step

excludes those points $z \in \mathbb{R}^2$ from $\mathcal{M}$ for which there exists some $y \in \mathcal{M}$ with $y_i \leq z_i$, $i = 1, 2$ and $y \neq z$. Such a filtering procedure is cheap thanks to the fact that the points are already sorted w.r.t. increasing first component and all points in $\mathcal{M}$ are already weakly non-dominated. In practice, we scroll the list $\mathcal{M}$ and, in case the distance with respect to the first component of two subsequent points is less than or equal to 10^{-5}, we remove the first point. For finite sets the domination holds, i.e., $f(\mathcal{X} \cap \mathbb{Z}^n) \subseteq \mathcal{Y}_N + \mathbb{R}^2_+$. Hence, we can use the same argumentation as in [32, Proposition 2.7] and get that for any weakly non-dominated point z in $\mathcal{M}$ which is not also non-dominated, a point $y \in \mathcal{Y}_N \subseteq \mathcal{M}$ exists with $y \leq z$, $y \neq z$. Thus, any point $z \in \mathcal{M}$ with $z \notin \mathcal{Y}_N$ is in fact filtered out. It remains to show that $\mathcal{Y}_N \subseteq \mathcal{M}$.

First, we prove that the ε-constraint method can find, at every iteration, a not-yet detected weakly efficient solution of (BOIP). We also show that no non-dominated point is missed in-between, where in-between refers to the sorting w.r.t. the first component, i.e. $f_1(x^{k-1}), f_1(x^{k-2}), \ldots, f_1(x^1), f_1(\hat{x})$. For the following, recall that $x^0 := \hat{x}$ is weakly efficient and is the point detected at 'iteration' $k = 0$ to $\varepsilon^0 := f_1(\hat{x})$.

Lemma 3.1.7. *Let Assumption 3.1.4 hold with $\gamma > 0$. Assume that $\delta \leq \gamma$ in Algorithm 1, $k \geq 1$, and that the point x^{k-1} is the point detected at iteration $k-1$. Then, at step k, Algorithm 1 finds a weakly non-dominated point $f(x^k)$ and no non-dominated point y with $y \neq f(x^k)$, $y \neq f(x^{k-1})$ and with $f_1(x^k) \leq y_1 \leq f_1(x^{k-1})$ exists.*

Proof. Let $x^k \in \operatorname{argmin}_{x \in \mathcal{X}^k \cap \mathbb{Z}^n} f_2(x)$ where $\mathcal{X}^k = \mathcal{X} \cap \{x \in \mathbb{R}^n : f_1(x) \leq \varepsilon^k\}$ and with $\varepsilon^k = f_1(x^{k-1}) - \delta$. By [36, Proposition 4.3] the point x^k is, as well as x^{k-1}, a weakly efficient solution of (BOIP). Assume that there exists an efficient point $\tilde{x} \in \mathcal{X} \cap \mathbb{Z}^n$ with $f_1(x^k) \leq f_1(\tilde{x}) \leq f_1(x^{k-1})$. By Assumption 3.1.4 we have either $f_1(\tilde{x}) = f_1(x^{k-1})$ or $f_1(\tilde{x}) \leq f_1(x^{k-1}) - \gamma$.

First, let $f_1(\tilde{x}) = f_1(x^{k-1})$. As x^{k-1} was obtained by solving $(\mathrm{P}_\varepsilon^{k-1})$ and as $\tilde{x}$ is feasible for this problem we have $f_2(\tilde{x}) \geq f_2(x^{k-1})$. As $\tilde{x}$ is efficient we derive $f(\tilde{x}) = f(x^{k-1})$. Second, let $f_1(\tilde{x}) \leq f_1(x^{k-1}) - \gamma$. As $-\gamma \leq -\delta$ it follows $f_1(\tilde{x}) \leq f_1(x^{k-1}) - \delta$. Then $\tilde{x}$ is feasible for $(\mathrm{P}_\varepsilon^k)$ and as a consequence $f_2(x^k) \leq f_2(\tilde{x})$. As $\tilde{x}$ is efficient and we have assumed that $f_1(x^k) \leq f_1(\tilde{x})$ we derive $f(x^k) = f(\tilde{x})$. □

As a consequence of Lemma 3.1.7, we have that $y \in \mathcal{M}$ for any non-dominated point $y \in \mathcal{Y}_N$ with $y_1 \in [f_1(x^{k-1}), f_1(\hat{x})]$. By the definition of $\hat{x}$ there is no $y \in \mathcal{Y}_N$ with $y_1 > f_1(\hat{x})$. Next we show that we miss no non-dominated point y with $y_1 < f_1(x^{k-1})$ by stopping the while loop based on $\varepsilon^k = f_1(x^{k-1}) - \delta < f_1(x^*)$, or in case we do not start the while loop at all, based on $\varepsilon^1 = f_1(\hat{x}) - \delta < f_1(x^*)$.

Lemma 3.1.8. *Let Assumption 3.1.4 hold with $\gamma > 0$. Assume that $\delta \leq \gamma$ in Algorithm 1 and that the algorithm stopped at some iteration k. Then, there is no non-dominated point y with $y_1 < f_1(x^{k-1})$ in case $k \geq 2$, and with $y_1 < f_1(\hat{x})$ in case $k = 1$.*

Proof. First, we consider the case $k \geq 2$. Then the algorithm stopped due to $\varepsilon^k = f_1(x^{k-1}) - \delta < f_1(x^*)$. Assume y is a non-dominated point with $y_1 < f_1(x^{k-1})$. By Assumption 3.1.4 we have $y_1 \leq f_1(x^{k-1}) - \gamma$ and hence $y_1 \leq f_1(x^{k-1}) - \delta < f_1(x^*)$ which is a contradiction to the definition of x^*. Next, let $k = 1$, i.e., the while loop did not start at all due to $\varepsilon^1 = f_1(\hat{x}) - \delta < f_1(x^*)$. Assume y is a non-dominated point with $y_1 < f_1(\hat{x})$. Then we obtain with Assumption 3.1.4, as before, that $y_1 \leq f_1(\hat{x}) - \gamma \leq f_1(\hat{x}) - \delta < f_1(x^*)$, which is again a contradiction to the definition of x^*. □

By Lemma 3.1.7 and Lemma 3.1.8 we have $\mathcal{Y}_N \subseteq \mathcal{M}$. However, there may exist weakly non-dominated points that cannot be found by our algorithm, as the following example shows:

Example 3.1.9. *Let $\mathcal{X} \cap \mathbb{Z} = \{1, 2, 3, 4\}$, $f_1(x) = 5 - x$ and $f_2(x) = 1$ for all $x \in \mathcal{X} \cap \mathbb{Z}$. Thus we have $\gamma = 1$. All feasible points x are weakly efficient, but only $x = 4$ is efficient. We apply Algorithm 1 with $\delta = 0.5 \leq \gamma$. Let $x^* = 4$, $\hat{x} = 1$ and thus $\varepsilon^0 = f_1(\hat{x}) = 4$. For $\varepsilon^1 = 3.5$ we may compute $x^1 = 4$ and the algorithm would stop without finding the remaining weakly efficient solutions.*

On the other hand, it may happen that the ε-constraint method needs to compute all weakly non-dominated points before detecting the complete Pareto front.

Example 3.1.10. *Let $\mathcal{X} = \{x \in \mathbb{R}^2 \mid 100x_2 \geq -x_1 + 100,\ 0 \leq x_1 \leq 100,\ 0 \leq x_2 \leq 1\}$ and let $f_1(x) = x_1$ and $f_2(x) = x_2$. Note that $\mathcal{Y}_N = \{(0, 1)^T, (100, 0)^T\}$. After having detected the non-dominated point $(100, 0)^T$, the ε-constraint method may need to address 100 more single-objective integer linear programming problems. Indeed, it may find the 99 weakly non-dominated points $(y_1, 1)^T$ with $y_1 = 99, 98, 97, \ldots, 2, 1$ before detecting $(0, 1)^T$ and with this additional point the complete Pareto front.*

It is important to note that the choice of δ has no impact on the number of iterations needed by the algorithm to stop. This means that using $\delta^1 \leq \gamma$ or $\delta^2 \leq \gamma$ with $0 < \delta^1 < \delta^2 \leq \gamma$ leads to the same result. So there is no need to find the largest possible value for γ but any γ for which Assumption 3.1.4 is satisfied will be enough. It is just important that δ is not chosen larger than any possible γ.

Lemma 3.1.11. *Let Assumption 3.1.4 hold with $\gamma > 0$. Assume that $0 < \delta^1 < \delta^2 \leq \gamma$ and that the point x^{k-1} is the point obtained by solving ($\mathrm{P}_\varepsilon^{k-1}$). Then a point $\bar{x}$ is an optimal solution of (P_ε^k) with $\varepsilon^k = f_1(x^{k-1}) - \delta_1$ if and only if $\bar{x}$ is an optimal solution of (P_ε^k) with $\varepsilon^k = f_1(x^{k-1}) - \delta_2$. Moreover, for any optimal solution $\bar{x}$ of (P_ε^k) with $\varepsilon^k = f_1(x^{k-1}) - \delta$ for some $\delta \in (0, \gamma]$ it holds $f_1(\bar{x}) \leq f_1(x^{k-1}) - \gamma$.*

Proof. First, let $\bar{x}$ be an optimal solution of (P_ε^k) with $\varepsilon^k = f_1(x^{k-1}) - \delta_2$. Then $\bar{x}$ is feasible for the problem with $\varepsilon^k = f_1(x^{k-1}) - \delta_1$. Assume $\bar{x}$ is not optimal for that problem. Then there exists x' feasible for the problem with $\varepsilon^k = f_1(x^{k-1}) - \delta_1$ with $f_2(x') < f_2(\bar{x})$. As $f_1(x') \leq f_1(x^{k-1}) - \delta_1$ we have $f_1(x') < f_1(x^{k-1})$ and we get by Assumption 3.1.4 $f_1(x') \leq f_1(x^{k-1}) - \gamma$. Thus $f_1(x') \leq f_1(x^{k-1}) - \delta_2$ and x' is feasible for (P_ε^k) with $\varepsilon^k = f_1(x^{k-1}) - \delta_2$. Thus $f_2(x') \geq f_2(\bar{x})$ which is a contradiction.

Next, let $\bar{x}$ be an optimal solution of (P_ε^k) with $\varepsilon^k = f_1(x^{k-1}) - \delta_1$. Thus $f_1(\bar{x}) < f_1(x^{k-1})$ and we get by Assumption 3.1.4 $f_1(\bar{x}) \leq f_1(x^{k-1}) - \gamma$. Thus $f_1(\bar{x}) \leq f_1(x^{k-1}) - \delta_2$ and $\bar{x}$ is feasible for (P_ε^k) with $\varepsilon^k = f_1(x^{k-1}) - \delta_2$. As the feasible set for (P_ε^k) for δ_2 is a subset of the feasible set for δ_1, we obtain that $\bar{x}$ is also optimal for the problem with δ_2. This also shows that any optimal solution $\bar{x}$ of (P_ε^k) with $\varepsilon^k = f_1(x^{k-1}) - \delta$ and $\delta \in (0, \gamma]$ satisfies $f_1(\bar{x}) \leq f_1(x^{k-1}) - \gamma$. □

Based on the previous lemmata we are able to prove the following result.

Theorem 3.1.12. *Let Assumptions 3.1.1, 3.1.2 and 3.1.4 hold. Let $\delta \leq \gamma$. Algorithm 1 finds the complete Pareto front $\mathcal{Y}_N$ of (BOIP) after having addressed a finite number of single-objective integer programs.*

Proof. By Lemma 3.1.7 and Lemma 3.1.8 we have $\mathcal{Y}_N \subseteq \mathcal{M}$ and after the filtering step, as discussed above, we obtain exactly the set $\mathcal{Y}_N$. Thanks to Assumption 3.1.4 we have that the `while` loop will take at most $m' = \lfloor (f_1(\hat{x}) - f_1(x^*))/\delta \rfloor$ iterations. Based on Lemma 3.1.11 it makes no difference within the `while` loop how $\delta \in (0, \gamma]$ is chosen and thus the upper bound is $m = \lfloor (f_1(\hat{x}) - f_1(x^*))/\gamma \rfloor$ iterations. Then, considering the two single-objective integer programs tackled at the beginning of Algorithm 1 for the computation of x^* and $\hat{x}$, the total number of single objective integer programs addressed by Algorithm 1 is $m + 2$. □

Note that the generated set $\mathcal{M}$ contains weakly non-dominated points only and in each step of the `while` loop in Algorithm 1 a not-yet detected weakly non-dominated point of (BOIP) is found. For that reason, the `while` loop will take at most as many iterations as the number of weakly non-dominated points. However, we have no guarantee that the set of weakly non-dominated points of (BOIP) is finite. But using the same arguments as in the proof of [32, Proposition 2.7.] we have that the number of weakly non-dominated points y with $f_1(x^*) \leq y_1 \leq f_1(\hat{x})$ is finite and that the number of iterations is finite. This is another possibility to prove Theorem 3.1.12 above. Moreover, in case there are no weakly non-dominated points which are not at the same time non-dominated, we need to solve exactly $|\mathcal{Y}_N| + 1$ single-objective integer programs.

3.2 Numerical results

In our computational experiments, we consider biobjective nonlinear integer instances arising from portfolio selection problems. Let $\mu \in \mathbb{R}^n$ be the expected return and $Q \in \mathbb{R}^{n \times n}$ be the covariance matrix with respect to a specific set of assets. We consider the following model

$$\begin{array}{ll} \min & (-\mu^T x,\ x^T Q x)^T \\ \text{s.t.} & a^T x \leq b \\ & x \geq 0 \\ & x \in \mathbb{Z}^n, \end{array} \tag{3.1}$$

where the elements of $a \in \mathbb{R}^n$ are the prices of the financial securities, $b \in \mathbb{R}$ is the budget of the investor and the non-negativity constraint rules out short sales. The decision variable $x_i \in \mathbb{Z},\ i = 1, \ldots, n$ stands for the amount of unit of a certain asset the investor is buying. Note that for this model Assumption 3.1.1 is satisfied, as the single-objective integer subproblem built from problem (3.1) is an integer quadratic problem that can be addressed by e.g. `GUROBI` [50].

As benchmark data set, we used historical real-data capital market indices from the Eurostoxx50 index that were used in [20, 21] and are publicy available at `https://www.francescocesarone.com/data-sets`. This data set was used for solving a *Limited Asset Markowitz (LAM)* model. For each of the 48 stocks the authors obtained 264 weekly price data, adjusted for dividends, from Eurostoxx50 for the period from March 2003 to March 2008. Stocks with more than two consecutive missing values were disregarded. Logarithmic weekly returns, expected returns and covariance matrices were computed based on the period March 2003 to March 2007. By choosing stocks at random from the 48 available ones, we built portfolio optimization instances of different sizes with $\mu \in \mathbb{Q}^n$ and $Q \in \mathbb{Q}^{n \times n}$. We decided to generate 60 different instances by considering $n = 5, 10, 25, 30$ stocks. For every n, 15 different instances have been generated. Hence, we got the covariances matrices, the expected returns and the prices for every combination by picking the proper information from the files provided. As in [16], for every instance, we set $b = 10 \sum_{i=1}^{n} a_i$, representing the budget of the investor. We compare the ε-constraint method with FPA*. This is an improved version of FPA which uses the so called *custom weighted-sum* scalarization. It is able to detect the complete Pareto front after having solved $|\mathcal{Y}_N| + 2$ integer programs (see [32] for further details). In order to run FPA* and the exact version of the ε-constraint method, we need a proper value $\gamma > 0$ so that the γ-positivity assumption is satisfied. Therefore, we had to pre-process the data. We trimmed the number of decimal digits to four and multiplied the entries by 10^3. As a consequence, we obtained $\gamma \geq 0.1$. Note that, since the entries of Q and μ are in $\mathbb{Q}$, the value γ can be defined as $1/r$, where $r \in \mathbb{N}$ is the least common multiple of the denominators of the rational coefficients (see [32, Proposition 4.14]).

Both in the implementation of FPA* and of the ε-constraint method, we considered the linear objective $-\mu^T x$ as the function defining the additional constraint in the single-objective

integer subproblems. Consequently, both FPA* and the ε-constraint method have to deal with a sequence of single-objective convex quadratic integer problems with linear constraints. In our Python implementation of the two algorithms, we used the MIQP solver of `GUROBI` [50]. All experiments have been executed on an Intel Core i5-6300U CPU running at 2.40 GHz and all running times were measured in cpu seconds.

In Table 3.1 and Table 3.2 we report, for each instance, the CPU time and the number of iterations ($\text{it}_{\text{FPA*}}$ and it_{eps}) needed by FPA* and the ε-constraint method to detect the non-dominated set. Note that the total number of single-objective subproblems solved by FPA* and ε-constraint method is $\text{it}_{\text{FPA*}} + 2$ and $\text{it}_{\text{eps}} + 2$, respectively. FPA* and the ε-constraint method have very similar performances, as the number of subproblems addressed by the two algorithms resulted to be very close in practice. However, the number of subproblems solved by the ε-constraint method can be larger than $|\mathcal{Y}_N| + 2$, which is the number of subproblems solved by FPA*. This is due to the fact that the ε-constraint method might also detects some weakly-nondominated points.

Table 3.1: Results on instances with $n = 5$ and $n = 10$ variables

Instance	$n = 5$				$n = 10$			
	`FPA*`	$\text{it}_{\text{FPA*}}$	ε-`const`	it_{eps}	`FPA*`	$\text{it}_{\text{FPA*}}$	ε-`const`	it_{eps}
p1	24.99	7381	24.60	7381	238.65	34890	234.75	34915
p2	6.69	1465	6.55	1467	78.24	10395	76.97	10390
p3	34.13	6103	32.89	6105	60.35	6047	60.55	6053
p4	1.89	402	1.81	402	47.17	6972	46.36	6978
p5	6.02	1208	5.86	1208	182.29	32894	180.58	32953
p6	3.09	709	3.03	709	50.24	6268	49.40	6272
p7	8.77	1676	8.56	1677	17.28	2372	17.17	2375
p8	4.61	845	4.20	846	230.24	23700	220.50	23714
p9	1.73	412	1.59	413	102.46	13544	96.96	13548
p10	2.09	428	1.95	430	93.25	8469	88.34	8471
p11	7.79	1547	7.29	1550	103.55	14987	98.22	14989
p12	4.44	837	4.13	840	69.71	7779	66.24	7784
p13	40.04	7257	37.41	7262	57.35	5057	54.45	5058
p14	5.45	1258	5.06	1259	305.46	26330	291.58	26339
p15	10.73	1869	10.00	1874	48.47	4990	46.50	4995

We further compare FPA* and the ε-constraint algorithm using performance profiles as proposed by Dolan and Moré [35]. Given a set of solvers $\mathcal{S}$ and a set of problems $\mathcal{P}$, the performance of a solver $s \in \mathcal{S}$ on problem $p \in \mathcal{P}$ is compared against the best performance obtained by any solver in $\mathcal{S}$ on the same problem. The performance ratio is defined as $r_{p,s} = t_{p,s} / \min\{t_{p,s'} \mid s' \in \mathcal{S}\}$, where $t_{p,s}$ is the measure we want to compare, and we consider a cumulative distribution function $\rho_s(\tau) = |\{p \in \mathcal{P} \mid r_{p,s} \leq \tau\}|/|\mathcal{P}|$. The performance profile for $s \in S$ is the plot of the function ρ_s. We report in Figure 3.1 the performance profiles of FPA* and the ε-constraint algorithm with respect to the CPU time considering all the 60 instances. Note that the value τ needed to have both $\rho_{\varepsilon\text{-const}}(\tau) = 1$ and $\rho_{FPA^*}(\tau) = 1$ is very small ($\tau = 1.116$), confirming that the two algorithm share very similar performance.

Table 3.2: Results on instances with $n = 25$ and $n = 30$ variables

Instance	$n = 25$				$n = 30$			
	FPA*	$\text{it}_{\text{FPA*}}$	ε-const	it_{eps}	FPA*	$\text{it}_{\text{FPA*}}$	ε-const	it_{eps}
p1	1300.58	64896	1265.80	64978	10068.92	168679	10137.36	168963
p2	3111.57	91538	3145.79	91595	3684.09	57207	3682.69	57209
p3	974.07	77143	989.09	77136	5381.17	85015	5416.53	85033
p4	543.68	35835	549.17	35850	9712.74	134844	9906.55	134681
p5	807.29	46351	821.02	46353	5226.97	87750	5311.91	87784
p6	737.88	36444	747.85	36438	7284.13	111329	7465.47	111349
p7	1723.60	126491	1733.17	126556	7036.66	131154	7121.33	131180
p8	954.39	39077	948.63	39085	6923.09	112315	6908.47	112349
p9	899.33	38413	884.50	38411	9984.14	139199	9973.19	139205
p10	631.12	28747	619.97	28755	7683.15	120145	7682.36	120152
p11	2326.07	84459	2305.43	84429	7489.53	121898	7480.73	121920
p12	1882.48	66347	1864.43	66359	4452.29	68145	4396.06	68150
p13	2234.69	105386	2201.69	105393	7281.69	128790	7164.96	128912
p14	1551.69	88263	1539.97	88210	4510.21	67305	4500.24	67342
p15	2224.76	112773	2182.22	112685	5657.23	87619	5647.94	87588

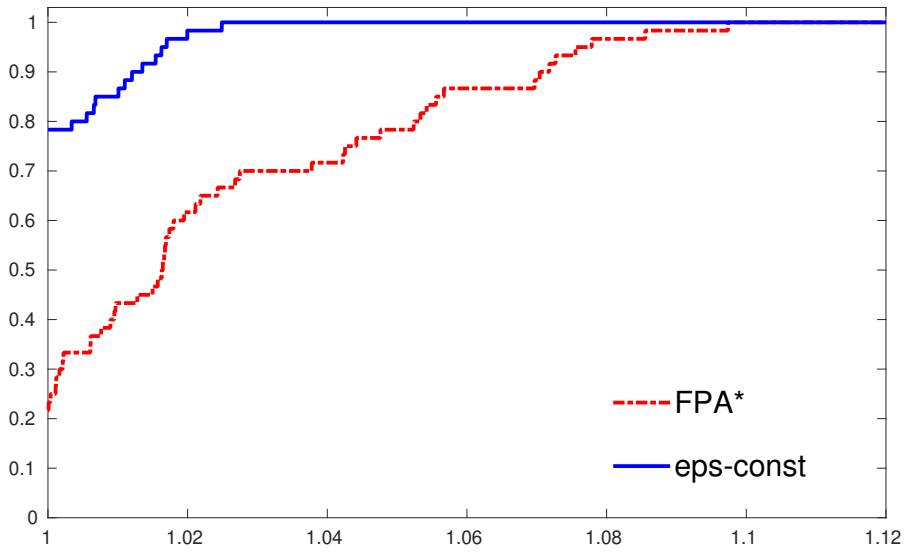

Figure 3.1: Comparison between FPA* and the ε-constraint method on all the 60 instances.

Chapter 4

Raw material size commonality decision tool: a biobjective integer programming model *

In manufacturing companies, the selection of input elements of production is recognized as an opportunity to decrease the cost complexity of several processes [81, 82]. These input elements are considered both raw materials and components that a company purchases as process inputs. The major benefits of the decrease of complexity related to the input elements are risk-pooling [7], lead-time uncertainty reduction [62], reduction of component inventories and simplification of processes and logistics while responding to variations in product demand [24, 78]. One strategy to reduce the complexity of input materials is to address the bill of materials (BOM) in what is known as the component commonality problem. Although authors use various terms and interpretations, component commonality typically refers to the ability to use a single component across multiple end products. For example, references [47, 53, 75] define the issue as utilizing the same version of a component across multiple products, while [8, 53] describe it as the degree to which the variations of a product family can be manufactured using the same components. The underlying concept directly relates to modularity, that is the potential to increase the differentiation of final products and enhance the output of the production process by standardizing and thereby reducing the variety of input components. It is important to note that while companies can modify the BOMs in component commonality, in many cases this is not feasible. Such scenarios include situations where customers have significant contractual power, products are subject to stringent regulations (e.g., in the pharmaceutical industry), or the materials are distinctive to the product and vital for market presence (e.g., in the fashion industry). In these instances, the input materials are dictated by the customers, the market, or regulatory bodies, leaving companies with no option to alter the BOMs. Therefore, in cases where a company has either already conducted a component commonality analysis or cannot modify the BOMs, the input elements of production might still exhibit some commonalities, such as raw materials or components used in more than one product. This research explores a further dimension of commonality - the commonality in the sizes of raw material, which is crucial when there is no opportunity to utilize leftover raw materials from the production process. Thus, we refer to input elements that are raw materials, and "raw material size" refers to the dimensions or volume of the raw material as it is initially purchased or procured before it is processed or used in produc-

*The content of this chapter is part of the submitted paper [25].

tion. This can include length, width, thickness, weight, or any other relevant measurement that defines how large or small the raw material is. The size of the raw material is critical in planning how it can be efficiently cut or shaped to meet the production requirements with minimal waste. For example, in clothing manufacturing, once the cutting problem is solved - determining the most efficient way to cut raw materials into specified sizes and shapes with minimal waste [74] - all remaining material becomes scrap. Another example is in some pharmaceutical productions, where any product left in an open pack must necessarily be discarded due to safety and good manufacturing practices. Our scenario is therefore characterized by 4 conditions: common raw materials among multiple products; raw materials purchased in various sizes; every batch of production is allocated to one raw material size; waste is generated in each batch of production considering the difference between the quantity required and the quantity of the purchased size. To better explain the case, imagine a company that manufactures two products. The company either analyzes Bill of Materials (BOM1 and BOM2) to identify a common raw material, or it is permitted by the market, customers, or regulations to use a common raw material for both products. Each production requires a specific amount of raw material, and the company cannot reuse the remaining material from production 1 in production 2. The difference in quantity between the material input size and the amount used in production constitutes scrap, thus representing a cost to minimize. This scenario underscores the significance of raw material sizes as a key driver for companies. On one hand, it aims to minimize waste and complexity; on the other, it ensures that each production batch corresponds to at least one available raw material size. We refer to this challenge as the Best Allocation of Resource Sizes (BARS) problem. The inputs of the BARS problem include the available resource sizes and the production demands for these resources. The objectives of the problem are twofold: firstly, reducing complexity and associated costs encourages minimizing the number of resource sizes within the company; secondly, reducing waste promotes increasing the number of resource sizes, ideally to one size for each quantity requested in production. Consequently, the problem just described leads back to an issue of decision-making, as a multi-objective optimization problem. The BARS problem is modeled as a biobjective nonlinear integer program, considering both the reduction of the number of the resource sizes and the total costs, and it is addressed with an adapted version of the ε-constraint algorithm described in Chapter 3. The remainder of the chapter is organized as follows. Section 4.1 describes the BARS problem. Section 4.2 defines the analytical formulation of the BARS problem. Section 4.3 describes three case studies, and it shows the results of the optimization. Section 4.4 draws conclusions and describes opportunities for future research.

4.1 Problem description

The problem examines how a manufacturing industry should produce formats (i.e. finished products) decreasing the number of different input items (i.e. raw materials sizes). The difference in size of the item and format produces scraps. Scraps cannot be reused within the process and therefore represent a waste. The problem aims to ensure the production of a range of formats by reducing the number of purchased items entering the process. To minimize scraps, it would be advisable to purchase items with similar dimensions to the formats to be produced, as adapting an item to the required format generates waste. However, this may result in an increased number of different items to be purchased, which negatively impacts costs and complexity management. Reducing the number of items simplifies the management of orders and stocks, while increasing the scrap level. By ordering larger quantities of a single item, discounts on purchases and risk-pooling effect can be obtained, i.e., the mechanism by which individual risks are shared and reduced collectively. Also, machine set-ups could be lowered. Since all formats reporting

positive demand must be manufactured, the items must be chosen to allow the entire production of all formats. This implies that the dimensions, or the weights, of the items purchased, must be greater than or equal to those of the formats. The purchase of all items may not be made at once, as shown below. To fully comprehend the impact of the management of different items, all these factors must be considered together, as well as approached independently to identify single effects. In our context, the smaller the number of items used, the higher the cost. This means that the problem we are dealing with is a biobjective optimization problem, namely a problem where we have two conflicting objective functions. The cost function we consider is made of several terms and we formalize each term in the following. The problem proposes a biobjective minimization approach that considers the reduction of the number of items and the costs. Several possible objective functions are formalized, one for each cost component. Importantly, the cost presented is not intended to be exhaustive. The model easily allows additional cost-terms to be included or existing cost-terms to be removed, to address a common problem in different industries. Notably, the problem does not consider energy costs and the ordering cost as invariant to the number of input components. The energy cost does not increase with increasing scrap because there are no reworks. So, the energy utilization is considered invariant. Also, the ordering cost is invariant because only one supplier is considered. Currently, the following cost-terms are included:

- purchase cost, which arises from buying items and it is impacted by opportunities to negotiate discounts with suppliers as the quantity ordered increases;
- scrap cost, representing the amount of discarded materials during production because of excess;
- storage cost, considering the quantities stored in the warehouse includes the costs of storage, transport, and labor;
- safety stock cost, kept to meet fluctuations in demand for formats;
- set-up cost for transitioning between item variants in manufacturing;
- total cost, which aggregates the above-mentioned costs;

The solution was not evaluated with respect to invariant costs. The presented methodology was tested across three case studies, each representing different industries with distinct cost structures. These industries include the cardboard packaging industry, the luxury leather goods industry, and the pharmaceutical industry. As an example, Figure 4.1 illustrates the items introduced into the production process to produce cardboard materials for packaging.

4.1.1 Assumptions

The following elements are known and time-invariant:

- purchase price of the item and the discount available beyond a predefined quantity threshold;
- supply lead time;
- service level to be provided;
- scrap cost is proportional to the quantity of discarded material for a given item and format. It is assumed that the discarded quantities cannot be recycled, reworked or reintroduced into the production process and are therefore considered waste.

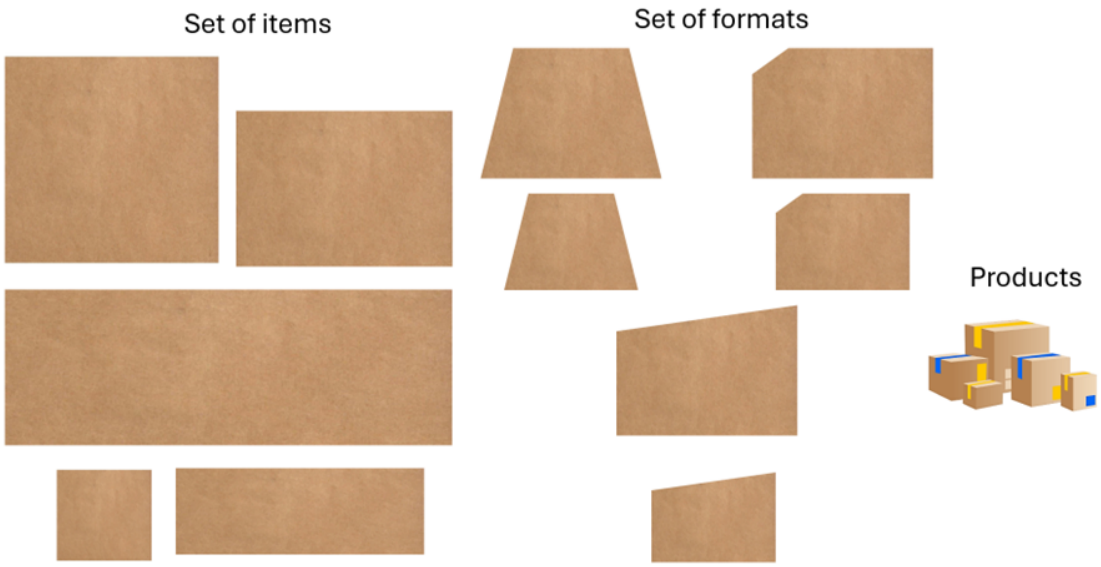

Figure 4.1: Example of items processed to obtain formats and thus end products in cardboard packaging industry.

- storage cost depends on the amount of the item. The storage capacity always meets the required holding level;
- set-up cost is independent of the item. When an item enters the production process to be converted into multiple formats, all these formats must be completed, requiring a number of machine set-ups equal to the number of items fed into the process. The demand for formats is variable and randomly generated in a set range of values. The discussion ignores the selling price of the product. It is assumed that the consumption of goods is highly price-sensitive, therefore prices are fixed and invariant to other model features.

4.2 Formulating the biobjective optimization model

Given a number n of items and a number m of formats our decision problem is defined considering the variables described in Table 4.1, where all the parameters used are also reported.

The constraints required by the model are as follows:

- formats' demand fulfilment:

$$\sum_{i=1}^{n} d_{ij} a_{ij} = D_j \quad j = 1, \ldots, m;$$

- logical relationship between d_{ij} and x_i:

$$\sum_{j=1}^{m} d_{ij} - D x_i \leq 0 \quad i = 1, \ldots, n,$$

where $D = \sum_{j=1}^{m} D_j$ is the total demand;

Table 4.1: Variables and parameters

Decision variables	$i = 1, \dots, n;\ j = 1, \dots, m$
$x_i \in \{0, 1\}$	$x_i = \begin{cases} 1 & \text{if item } i \text{ is chosen,} \\ 0 & \text{otherwise} \end{cases}$
$y_{ij} \in \{0, 1\}$	$y_{ij} = \begin{cases} 1 & \text{if item } i \text{ is used to fulfill the demand of format } j, \\ 0 & \text{otherwise} \end{cases}$
$d_{ij} \in \mathbb{Z}$	amount of demand of format j satisfied by purchasing item i
Parameters	
c_i	purchase cost of a unit of item i
dis_i	discount factor for item i
s_{ij}	amount of item i wasted if format j is assigned to item i
S_i	unit scrap cost for item i
a_{ij}	$a_{ij} = \begin{cases} 1 & \text{if format } j \text{ can be assigned to item } i, \\ 0 & \text{otherwise} \end{cases}$
D_j	demand of format j
CUM_i	unit storage cost of item i
sc_i	set-up cost of item i
$\bar{D}_i$	quantity discount parameter
$CUMS_i$	unit safety storage cost of item i
k	service level used for the computation of the safety stock level
σ_j^2	demand variance of format j
TP	supply lead time

- logical relationship between y_{ij} and d_{ij}:

$$d_{ij} - y_{ij} \geq 0, \quad i = 1, \dots, n, \quad j = 1, \dots, m,$$
$$d_{ij} - D_j y_{ij} \leq 0, \quad i = 1, \dots, n, \quad j = 1, \dots, m;$$

- non-negativity of d_{ij}:

$$d_{ij} \geq 0, \quad i = 1, \dots, n, \quad j = 1, \dots, m.$$

4.2.1 Objective functions

As already mentioned, in our biobjective model we consider as objective functions the minimization of the number of items to be used and the minimization of a cost function. The number of items used is obviously modeled as the sum:

$$f_1(x, y) := \sum_{i=1}^{n} x_i \tag{f_1}$$

For what concerns the cost function, we consider the sum of five different cost-terms

$$f_2(x, y, d) := t_p(d) + t_s(d) + t_{su}(x) + t_{stc}(d) + t_{ssc}(y) \tag{f_2}$$

each presented in the following. Note that each cost term can also be considered separately as a second objective function along with (f_1). We are going to analyze this possibility in our numerical tests (see Section 4.3).

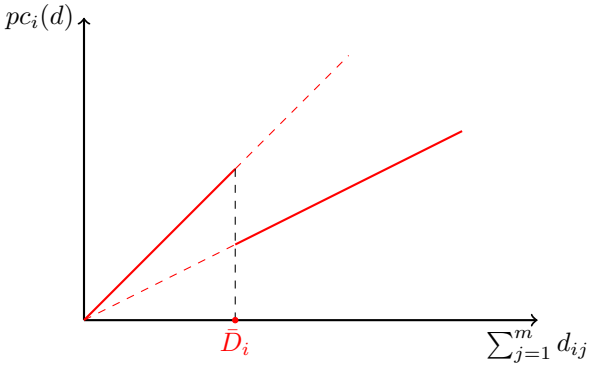

Figure 4.2: The purchase cost $pc_i(d)$ for item i

The purchase cost term with a quantity discount option

The *purchase cost* we consider includes a quantity discount and is modeled by means of the piece-wise linear function, reported below:

$$t_p(d) = \sum_{i=1}^{n} pc_i(d),$$

where

$$pc_i(d) = \begin{cases} c_i \sum_{j=1}^{m} d_{ij} & \text{if } \sum_{j=1}^{m} d_{ij} \leq \bar{D}_i \\ dis_i\, c_i \sum_{j=1}^{m} d_{ij} & \text{if } \sum_{j=1}^{m} d_{ij} > \bar{D}_i \end{cases}$$

As shown in Figure 4.2, the purchase cost of item i is entirely reduced by a discount factor dis_i. In this case, the discount is applied to the price of all items, because the quantity has exceeded the threshold. It is also possible to model a discount on the purchase cost such that it is applied only on the units exceeding the threshold $\bar{D}_i$. In this case the purchase cost is shown in Figure 4.3 and described by the function $\bar{pc}(d)$:

$$\bar{pc}_i(d) = \begin{cases} c_i \sum_{j=1}^{m} d_{ij} & \text{if } \sum_{j=1}^{m} d_{ij} \leq \bar{D}_i \\ c_i(1 - dis_i)\bar{D}_i + dis_i c_i \sum_{j=1}^{m} d_{ij} & \text{if } \sum_{j=1}^{m} d_{ij} > \bar{D}_i \end{cases}$$

The scrap, set-up, storage and stock cost terms

Given that item i has a unitary scrap cost of S_i and that s_{ij} is the units of item i being discarded after the production of format j, the *scrap cost* is modeled as the following linear function in the variables d_{ij}:

$$t_s(d) = \sum_{i=1}^{n} S_i \sum_{j=1}^{m} s_{ij} d_{ij}.$$

Parameter sc_i represents the set-up cost for item i and it is faced by a production plant only if item i is chosen. The *set-up cost* function is thus reported below:

$$t_{su}(x) = \sum_{i=1}^{n} sc_i x_i.$$

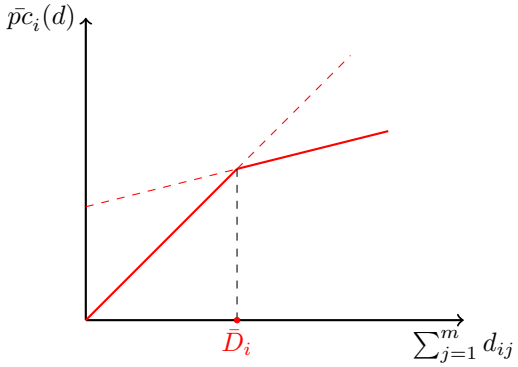

Figure 4.3: The purchase cost $\bar{pc}_i(d)$ for item i

Note that the set-up cost does not conflict with $f_1(x)$, so we always use it in combination with other cost-terms.

In order to represent the stock cost of item i, we sum its unitary storage cost CUM_i over the number of times that item i is purchased. We then obtain the *stock cost* as a linear function in the variables d_{ij}:

$$t_{stc}(d) = \sum_{i=1}^{n} CUM_i \sum_{j=1}^{m} d_{ij}.$$

The safety-stock cost term and a reformulation via second-order cone programming

The total safety-stock cost is described by the function below:

$$t_{ssc}(y) := k \sum_{i=1}^{n} CUMS_i \sqrt{\sum_{j=1}^{m} TP\,\sigma_j^2\, y_{ij}}.$$

In order to numerically deal with this term, we introduce in the model additional variables $z_i \in \mathbb{R}$ and suitable second-order cone constraints so that $t_{ssc}(y)$ can be reformulated as a function of z:

$$t_{ssc}(y) = t_{ssc}(z) := k \sum_{i=1}^{n} CUMS_i\, z_i,$$

with

$$z_i^2 = \sum_{j=1}^{m} TP\,\sigma_j^2\, y_{ij}^2, \quad i = 1, \ldots, n,$$

where we use the fact that $y_{ij} = y_{ij}^2$. Note that the above constraints can be seen as

$$z_i^2 = \|Q_i y_i\|^2,$$

being Q_i the diagonal matrix defined as

$$Q_i = \begin{pmatrix} \sqrt{TP\,\sigma_1^2} & 0 & \cdots & 0 \\ 0 & \sqrt{TP\,\sigma_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{TP\,\sigma_m^2} \end{pmatrix}$$

and $y_i := (y_{i1}, y_{i2}, \ldots, y_{im})^T$.

4.2.2 Complete formulation

Summarizing what has been presented in this section, we can write the biobjective model considered as follows:

$$\begin{aligned}
\min \quad & (f_1(x), f_2(x,y,d))^T \\
\text{subject to} \quad & \sum_{i=1}^{n} d_{ij} a_{ij} = D_j && j = 1,\ldots,m; && \text{(4.1a)}\\
& \sum_{j=1}^{m} d_{ij} - D x_i \leq 0 && i = 1,\ldots,n; && \text{(4.1b)}\\
& d_{ij} - y_{ij} \geq 0 && i = 1,\ldots,n, \quad j = 1,\ldots,m; && \text{(4.1c)}\\
& d_{ij} - D_j y_{ij} \leq 0, && i = 1,\ldots,n, \quad j = 1,\ldots,m; && \text{(4.1d)}\\
& d_{ij} \geq 0, && i = 1,\ldots,n, \quad j = 1,\ldots,m; && \text{(4.1e)}\\
& x_i \in \{0,1\}, && i = 1,\ldots,n; && \text{(4.1f)}\\
& y_{ij} \in \{0,1\}, && i = 1,\ldots,n, \quad j = 1,\ldots,m; && \text{(4.1g)}\\
& d_{ij} \in \mathbb{Z}, && i = 1,\ldots,n, \quad j = 1,\ldots,m; && \text{(4.1h)}
\end{aligned}$$

Note that we do not include the z variables in this model, as they are used as additional variables to handle the square root in the safety-stock cost.

Considering that the total cost function contains nonlinear terms, the model we end up with belongs to the class of biobjective nonlinear integer problems (BOIP). Therefore, we apply the ε-constraint algorithm version discussed in Chapter 3, exploiting the fact that function $f_1(x)$ is a positive γ function with $\gamma = 1$, for retrieving all the nondominated solutions of the biobjective problem.

4.3 Numerical results and discussion

The model described and Algorithm 1 have been implemented in Python 3.10 using Gurobi 9.5.1 as a MIP solver. The code has been executed on a laptop mounting an Intel® Core™ i5-6300U quad-core processor running at 2.40GHz. We analyze three case studies of the model proposed in Section 4.2.1. Every case study is built starting from real world data. For every case study, we consider the model as described in Section 4.2.2. In addition to the biobjective model having as second objective function the sum of all the cost-terms, we consider three additional settings where single cost-terms, namely the purchase cost, the scrap cost and the stock cost, are considered as alternative second objective functions. For each case study, we report three couples of plots (see Figure 4.6, Figure 4.9 and Figure 4.12). Each couple is related to a specific second objective function. On the left, the non-dominated set of the biobjective model considered is depicted by the points highlighted in orange. We also report in blue, those points obtained minimizing the second objective functions, for those number of items not leading to non-dominated points. Note that this only applies to case study 2 (see Figure 4.9). Given a point in the decision space leading to an image point in the left side graph, the corresponding value of the total cost is computed. That, along with the value of the first objective function, namely the number of activated items, is represented on the right side graph.

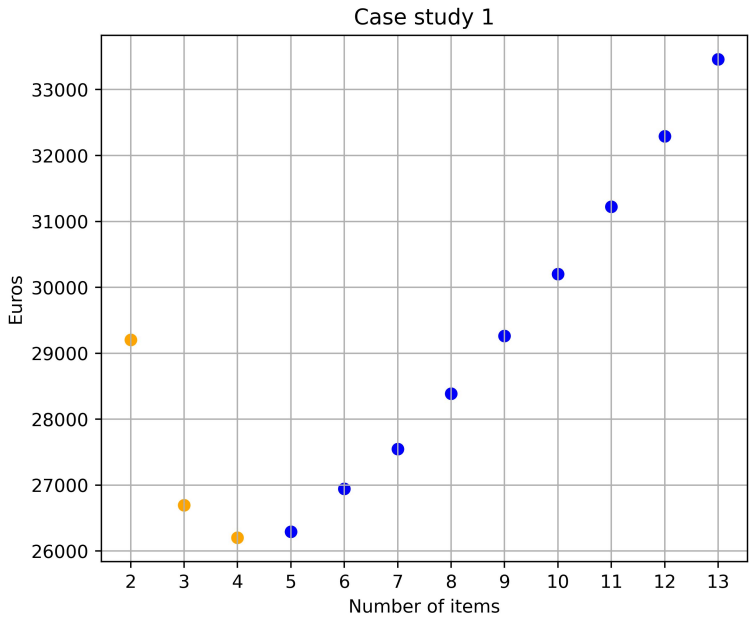

Figure 4.4: The image space for case study 1. The non-dominated set is depicted by the points highlighted in orange.

4.3.1 Case study 1: secondary cardboards for drug

The first sector for which we present a case study pertains to the production of secondary packaging for the pharmaceutical industry, i.e. packages that are not in physical contact with the product. Such packaging is not produced from materials suitable for direct contact with drugs, such as plastic, glass, and aluminum, but from easily identifiable materials such as paper, cardboard, or plastic. In this case, the packaging is made of cardboard. The purchase cost is quite low and proportional to the quantity of cardboard ordered. The supplier offers quantity discounts. Storage costs are low as the cartons do not require special environmental conditions or storage arrangements. Supply is quick, with a lead time of only a few days. The most significant cost is related to the set-up of the machines, which must guarantee diverse production in shape and size, high productivity, and flexibility. The pharmaceutical sector demands precise and reliable packaging, necessitating control systems that prevent the waste of cartons from the system. Fast format changeovers are necessary to accommodate small production batches.

In Figure 4.4, we report the non-dominated set obtained for case study 1: the non-dominated points detected by the ε-constraint algorithm are 3, highlighted in orange. To get further insights, we also report, in blue, points belonging to the image space computed as follows. Provided that the number of items is fixed to a certain value that does not lead to a non-dominated point, the total cost is minimized accordingly. For this case study, we could compute 9 additional points in such a way.

Figure 4.5 shows the cost of each set in Figure 4.4, where the objective functions are the number of items and the total cost. These trends are also shown in Table 4.2.

Purchase and set-up costs have the greatest impact on total costs, up to 60.99% for two items and up to 46.62% for thirteen items. Stock and safety stock costs have the most residual impact. For more than four items, it becomes uneconomical to consider further items: the solutions

Table 4.2: Cost components as the number of items changes for scenario 1.

Set of active items (Nr.)	Purchase Cost		Scrap Cost		Stock Cost		Safety Stock Cost		Set-up Cost		Total Cost (TC)	
	Amount (€)	% of TC	Amount (€)	% of TC	Amount (€)	% of TC	Amount (€)	% of TC	Amount (€)	% of TC	Amount (€)	% of TC
{12,13} (2)	17.466,29 €	59,80%	8.716,63 €	29,84%	596,33 €	2,04%	27,39 €	0,09%	2.400,00 €	8,22%	29.206,64 €	100,00%
{7,12,13} (3)	16.279,64 €	60,99%	6.230,45 €	23,34%	556,37 €	2,08%	28,00 €	0,10%	3.600,00 €	13,49%	26.694,46 €	100,00%
{5,6,12,13} (4)	15.783,19 €	60,23%	5.046,73 €	19,26%	535,05 €	2,04%	39,50 €	0,15%	4.800,00 €	18,32%	26.204,47 €	100,00%
{2,6,7,12,13} (5)	15.387,32 €	58,52%	4.342,28 €	16,51%	523,41 €	1,99%	41,51 €	0,16%	6.000,00 €	22,82%	26.294,52 €	100,00%
{2,5,6,11,12,13} (6)	15.214,97 €	56,47%	3.963,72 €	14,71%	515,98 €	1,91%	49,62 €	0,18%	7.200,00 €	26,72%	26.944,29 €	100,00%
{1,3,5,6,11,12,13} (7)	15.012,40 €	54,50%	3.570,88 €	12,96%	510,73 €	1,85%	53,51 €	0,19%	8.400,00 €	30,49%	27.547,52 €	100,00%
{1,3,5,6,7,11,12,13} (8)	14.894,28 €	52,47%	3.331,78 €	11,74%	507,78 €	1,79%	53,83 €	0,19%	9.600,00 €	33,82%	28.387,67 €	100,00%
{1,3,5,6,7,8,11,12,13} (9)	14.789,42 €	50,53%	3.115,86 €	10,65%	503,84 €	1,72%	56,83 €	0,19%	10.800,00 €	36,90%	29.265,95 €	100,00%
{1,2,3,5,6,7,8,11,12,13} (10)	14.704,93 €	48,69%	2.934,28 €	9,72%	500,50 €	1,66%	60,35 €	0,20%	12.000,00 €	39,74%	30.200,06 €	100,00%
{1,2,3,4,5,6,7,8,11,12,13} (11)	14.644,99 €	46,90%	2.820,45 €	9,03%	498,67 €	1,60%	61,80 €	0,20%	13.200,00 €	42,27%	31.225,91 €	100,00%
{1,2,3,4,5,6,7,8,9,11,12,13} (12)	14.606,57 €	45,23%	2.724,99 €	8,44%	497,01 €	1,54%	64,61 €	0,20%	14.400,00 €	44,59%	32.293,18 €	100,00%
{1,2,3,4,5,6,7,8,9,10,11,12,13} (13)	14.594,10 €	43,62%	2.703,6 €	8,08%	496,7 €	1,48%	66,69 €	0,20%	15.600,00 €	46,62%	33.461,05 €	100,00%

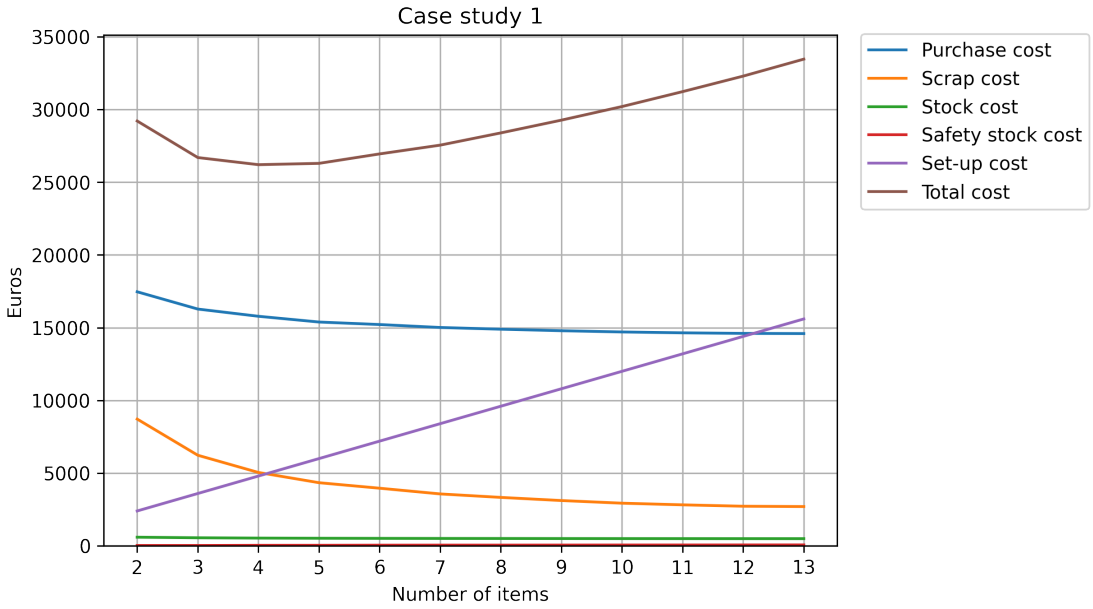

Figure 4.5: Cost trends as the number of items changes for scenario 1.

are all dominated. This is because, even if the increase in the number of items leads to lower purchase and scrap costs, the setup costs exceed the scrap costs when the number of items is equal to five, and it gets progressively closer to the purchase costs until it exceeds it at 12 items. Considering these results and the biobjective minimization (i.e., number of items and costs), the following sets of items can be considered: {12,13}; {7,12,13}; {5,6,12,13}. From a management perspective, to reduce overall costs, an intervention in the set-up process would be necessary to make it less onerous. The results indicate that it is not cost-effective for the production process under consideration to process a wide variety of raw materials. That is, the scrap costs associated with a limited number of items do not compromise production efficiency, and on the contrary, a greater number of items entails significant set-up costs. Furthermore, suppliers offer quantity-based discounts, which is consistent with the decision to reduce the number of different items purchased in favor of greater quantities of individual items. This approach also helps to reduce purchasing lead times and increase control over them, enabling the company to be more punctual and responsive to the market. When dealing with a commodity with regular demand and high price sensitivity, the delivery time and the price of the product are crucial aspects. Considering these factors, the optimal approach should be the progressive cost-reduction, i.e. gradually reducing the manufacturing costs to keep the price of the product constant and to increase the company's profitability. This means focusing on internal processes, in this case in the set-up design. Internal inefficiencies resulting in high production costs may force the company to increase the price of the product to guarantee a profit, damaging the satisfaction of the end customer and, consequently, the effectiveness of the process.

4.3.2 Case study 2: leather for haute couture

The second sector pertains to the processing of leather for haute couture, a niche industry known for its high-quality raw materials and manufacturing processes. The purchase cost is high and

not proportional to the quantity of material, as large-sized leather is rare and difficult to procure. Suppliers do not provide quantity discounts. Storage costs are relatively low compared to the value of the product. Although security measures must be taken to store valuable goods, there are no special environmental or atmospheric conditions for storage. Set-up costs are slightly lower than in the carton sector due to the lower speed and flexibility required in production. Procurement time for raw materials can take several months as they are often sourced directly from the countries of origin to ensure product traceability and avoid intermediaries. The direct contact with suppliers, even if may take longer, guarantees the quality of the final product.

In Figure 4.7, we report the non-dominated set obtained for case study 2. As before, the non-dominated points detected by the ε-constraint algorithm are highlighted in orange. Considering a number of items up to 5 leads to non-dominated points. On the other hand, choosing 6 items and minimizing the related total cost, leads to the dominated point highlighted in blue.

Figure 4.8 shows the cost of each set of items from Figure 4.7, where the objective functions minimize the number of items and the total cost respectively. These trends are also shown in Table 4.3.

The scrap cost has the highest impact on the total cost, with a maximum percentage of 56.69% for one item. Purchase and stock costs also have a significant impact on the total cost, with maximum percentages of 36.42% and 25.6% respectively for six items. Set-up and safety stock costs have a residual impact. Clearly, it is not cost-effective to increase the number of items to more than 5. This is due to the slight increase in the cost of safety stock, stock, and set-up. Considering these results and the biobjective minimization (i.e., number of items and costs), the following sets of items can be considered: {2}; {2, 4}; {2, 4, 5}; {1, 4, 5, 6}; {1, 3, 4, 5, 6}. From a management perspective, to reduce overall costs, interventions should be targeted at storage and/or purchasing and/or scrap costs. In the case study of leather for haute couture, it is clear that the potential to reduce purchasing costs is limited. The luxury product is expensive and suppliers are not willing to offer quantity discounts. Therefore, it is advisable to diversify the number of purchased items. Purchasing raw materials with dimensions as close as possible to the dimensions of the products to be manufactured can significantly reduce scrap costs, which are the most influential in the process. Increasing the number of items and types of raw materials purchased from suppliers could lead to longer and less controllable lead times, resulting in reduced timeliness and punctuality. However, as the product is considered a luxury item, customers may not be as sensitive to such aspects. It is important to maintain a perception of exclusivity, which may lead customers to be willing to wait longer than expected to obtain the product. Additionally, implementing recycling, reuse, and reconditioning of discarded materials can improve process management and production efficiency.

4.3.3 Case study 3: active ingredient for pharma

The third analyzed sector pertains to the pharmaceutical industry, specifically the production of active pharmaceutical ingredients. These are then combined with excipients, substances used to prepare the final pharmaceutical products. This sector is crucial for public health, as it serves as the foundation for the manufacture of drugs. The active ingredients must meet the highest standards of quality, traceability, and reliability in the chemical industry. Suppliers may offer quantity discounts, and purchasing a larger quantity can result in a lower unit cost. However, it is important to note that the product must be kept refrigerated and is subject to degradation, so the storage costs are high compared to the value of the product. The set-up cost is lower than in previous sectors due to the simpler machinery setup process, which involves operations exclusively linked to the weight of the raw materials. The supply time is typically just over a month.

Table 4.3: Cost components as the number of items changes for scenario 2.

Set of active items (Nr.)	Purchase Cost		Scrap Cost		Stock Cost		Safety Stock Cost		Set-up Cost		Total Cost (TC)	
	Amount (€)	% of TC	Amount (€)	% of TC	Amount (€)	% of TC	Amount (€)	% of TC	Amount (€)	% of TC	Amount (€)	% of TC
{2} (1)	19.303.200,00 €	27,24%	40.167.603,60 €	56,69%	11.211.298,56 €	15,82%	177.767,60 €	0,25%	1.080,00 €	0,00%	70.860.949,76 €	100%
{2, 4} (2)	15.192.600,00 €	30,25%	24.752.853,60 €	49,28%	10.080.883,56 €	20,07%	202.248,48 €	0,40%	2.160,00 €	0,00%	50.230.745,64 €	100%
{2, 4, 5} (3)	13.873.200,00 €	30,97%	20.900.205,60 €	46,65%	9.798.356,04 €	21,87%	223.219,99 €	0,50%	3.240,00 €	0,01%	44.798.221,63 €	100%
{1, 4, 5, 6} (4)	13.331.350,00 €	34,54%	15.632.529,60 €	40,50%	9.412.059,80 €	24,38%	218.995,41 €	0,57%	4.320,00 €	0,01%	38.599.254,81 €	100%
{1, 3, 4, 5, 6} (5)	13.160.900,00 €	36,45%	13.443.951,60 €	37,23%	9.251.564,08 €	25,62%	245.203,62 €	0,68%	5.400,00 €	0,01%	36.107.019,30 €	100%
{1, 2, 3, 4, 5, 6} (6)	13.161.000,00 €	36,42%	13.444.611,60 €	37,20%	9.251.612,48 €	25,60%	274.831,56 €	0,76%	6.480,00 €	0,02%	36.138.535,64 €	100%

In Figure 4.10, we report the non-dominated set obtained for case study 3. In this case, every possible choice concerning the number of items leads to a non-dominated point, found by the ε-constraint algorithm. The non-dominated set in this case is made of 6 points.

Figure 4.11 shows the cost of each set in Figure 4.10, where the objective function used are the number of items and the total cost. These trends are also shown in Table 4.4.

The scrap cost has the highest impact on the total cost, with a maximum percentage of 94% for a single item. The other cost items have a residual influence, except for the stock cost, which reaches an influence on the total cost of 18.81% for five items. The results indicate that increasing the number of items always leads to lower total costs due to the significant reduction of the scrap cost. The cost of scrap decreases its influence by 14.2%, from 94% to 79.8%, when choosing to buy only one item or all items. This saving outweighs the increase in the cost of stock, which rises from 5.78% to 18.81%. Considering these results and the biobjective minimization (i.e., number of items and costs), the following sets of items can be considered: {6}; {3, 6}; {2, 3, 6}; {2, 3, 5, 6}; {1, 2, 3, 5, 6}; {1, 2, 3, 4, 5, 6}. From a management perspective, to reduce overall costs, interventions on scrap costs are needed primarily. However, when considering the case study, i.e., the active ingredient for pharmaceuticals, it is clear that there is little potential to reduce these costs. The raw material cannot be recycled, as it deteriorates after a short time within the process and has to be discarded. Also, in terms of storage costs, there is little opportunity to intervene. The product is extremely sensitive to environmental conditions, and it is crucial to ensure the best storage conditions at all times. Even slight alterations to the storage parameters can irreversibly damage the active ingredients, resulting in the obligation to discard the entire material. Considering production efficiency, it is always beneficial to diversify the type of input material and purchase batches as close as possible to the required quantity for production. This significantly reduces waste levels, which otherwise would be difficult to recycle.

4.4 Conclusions and future work

This chapter deals with the industrial problem of "raw material sizes commonality", which refers to using the same raw material for multiple products, the impossibility of reusing the input material remaining from the difference of raw material size and batch request. We introduced and explained all the characteristics of this problem introducing it as a novel optimization challenge, referring to it as the Best Allocation of Resource Sizes (BARS) problem. The objective is to minimize the number of purchased items as input of the process, exploring the cost implications of reducing the raw materials sizes. The total costs include purchase, scrap, storage, safety stock, and set-up costs, which are differently affected by the number and sizes of raw materials. The problem is modeled as a biobjective nonlinear integer program, considering both the minimization of the number of items and the different cost components considered. It is solved using an adapted version of the ε-constraint method. It is noteworthy that in the biobjective model, which aims to minimize the cost and number of raw materials sizes, the costs of storage and safety stock are consistent with the minimization of the number of items and therefore do not contribute to the trade-offs in the Pareto frontier. The proposed model has been tested and validated in three real case studies. The case studies pertain to three distinct industrial sectors, each utilizing specific raw materials: secondary cardboard for pharmaceuticals, leather for haute couture, and active ingredients for pharmaceutical manufacturing. Sectors differ in the availability of quantity discounts from suppliers and in the impact of individual cost components on total costs. The model provides a decision support tool to allow management to understand the consequences associated with the number of sizes of raw materials. For example, depending on the strategic objectives, a limit can be imposed on the costs to be incurred, either in total or in part, i.e.

Table 4.4: Cost components as the number of items changes for scenario 3.

Set of active items (Nr.)	Purchase Cost		Scrap Cost		Stock Cost		Safety Stock Cost		Set-up Cost		Total Cost (TC)	
	Amount (€)	% of TC	Amount (€)	% of TC	Amount (€)	% of TC	Amount (€)	% of TC	Amount (€)	% of TC	Amount (€)	% of TC
{6} (1)	117.435,43 €	0,06%	189.378.000,00 €	94,00%	11.637.000,00 €	5,78%	323.689,19 €	0,16%	360,00 €	0,00%	201.456.484,63 €	100%
{3, 6} (2)	56.814,38 €	0,13%	37.778.000,00 €	88,15%	4.815.000,00 €	11,23%	206.996,97 €	0,48%	720,00 €	0,00%	42.857.531,35 €	100%
{2, 3, 6} (3)	50.664,86 €	0,19%	22.878.000,00 €	83,85%	4.144.500,00 €	15,19%	210.546,91 €	0,77%	1.080,00 €	0,00%	27.284.791,76 €	100%
{2, 3, 5, 6} (4)	49.215,50 €	0,21%	19.278.000,00 €	81,94%	3.982.500,00 €	16,93%	215.154,36 €	0,91%	1.440,00 €	0,01%	23.526.309,86 €	100%
{1, 2, 3, 5, 6} (5)	48.561,19 €	0,22%	17.428.000,00 €	80,69%	3.899.250,00 €	18,05%	222.290,75 €	1,03%	1.800,00 €	0,01%	21.599.901,94 €	100%
{1, 2, 3, 4, 5, 6} (6)	48.059,70 €	0,23%	16.328.000,00 €	79,80%	3.849.750,00 €	18,81%	233.502,39 €	1,14%	2.160,00 €	0,01%	20.461.472,09 €	100%

concerning specific cost items, and the feasible solutions can be analyzed in terms of the sets of items that can be purchased. It is still possible to understand how certain cost items can be reduced and how the process can be made more efficient, given the available resources and the current processes. The results indicate that selecting a wide variety of raw materials sizes for the secondary cardboard for drugs is not cost-effective. This is because the waste costs associated with a limited number of articles do not compromise production efficiency. On the contrary, a larger number of items leads to considerable set-up costs. In the second scenario, which involves leather for haute couture, it is recommended to purchase a variety of items to reduce waste costs, which have the highest impact on total cost. Recycling discarded materials can also be beneficial. While diversifying the number of items may result in longer lead times and reduced punctuality, it is important to note that the product is considered a luxury item and customers may not be as concerned with these factors. In the third scenario, which concerns active pharmaceutical ingredients, the results indicate that it is always beneficial to diversify the type of input material and to purchase batches as close as possible to the quantity required for production. This significantly reduces waste levels, which cannot be recycled as they deteriorate after only a short time entering the production process. The limitations of the research lie in the specific cost components included in the analysis. Furthermore, assumptions have been applied to the model, such as the impossibility of recycling discarded material, or other assumptions defined in the problem. However, the input data used to solve the problem come from real world and the results are consistent with the industrial context considered. Future developments aim to extend the model to include additional cost components, as the biobjective model is flexible in this respect. Further sensitivity analyses can assess the impact of different sets of items when the weights of individual cost items vary. Finally, the model could be implemented directly by the companies concerned, as it is an accessible and straightforward tool that can be readily integrated into existing processes. It offers a simple and effective solution for analyzing the cost impact of purchasing choices of different raw materials and, therefore, on production methods.

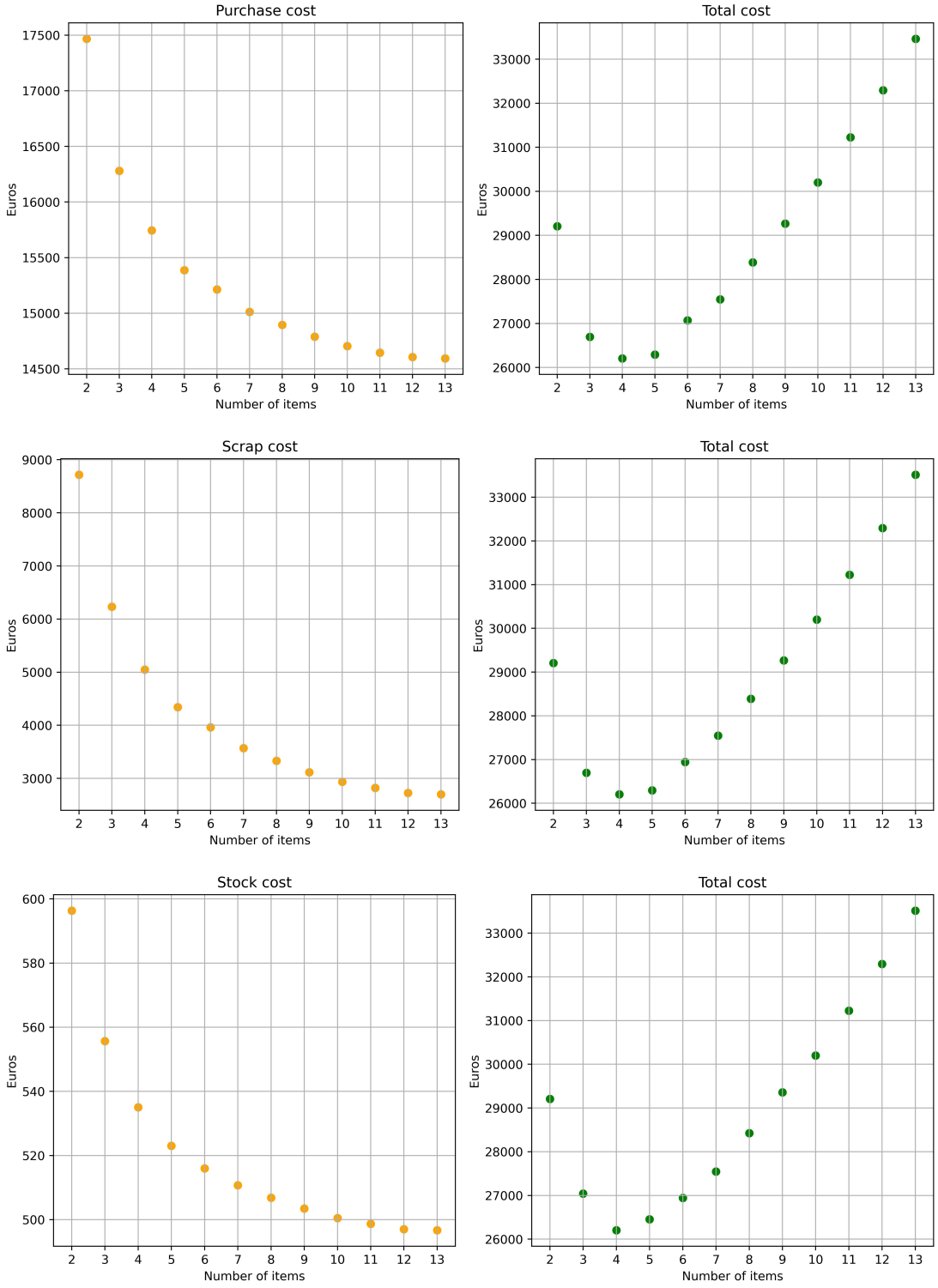

Figure 4.6: Case study 1. On the left, the image space for the models was obtained considering the purchase cost, the scrap cost, and the stock cost, respectively. On the right, is the corresponding total cost.

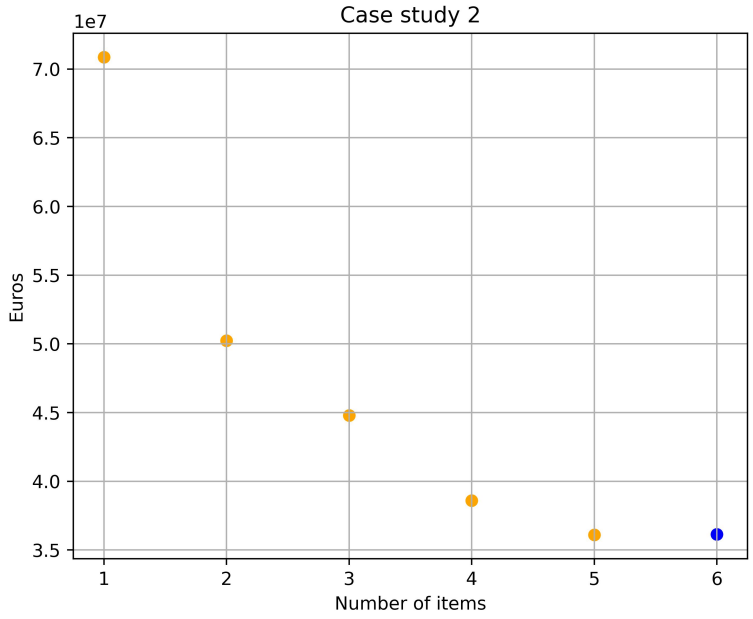

Figure 4.7: The image space for case study 2. The non-dominated set is depicted by the points highlighted in orange.

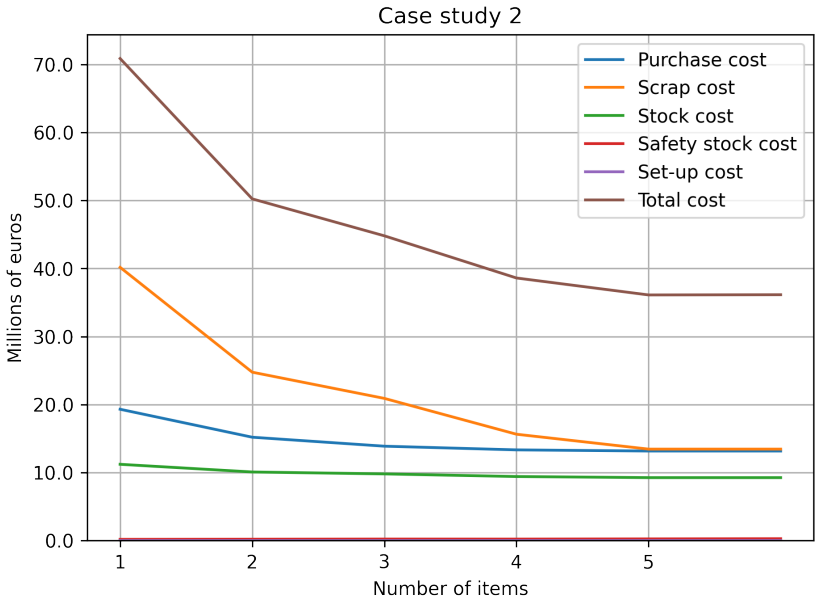

Figure 4.8: Cost trends as the number of items changes for scenario 2.

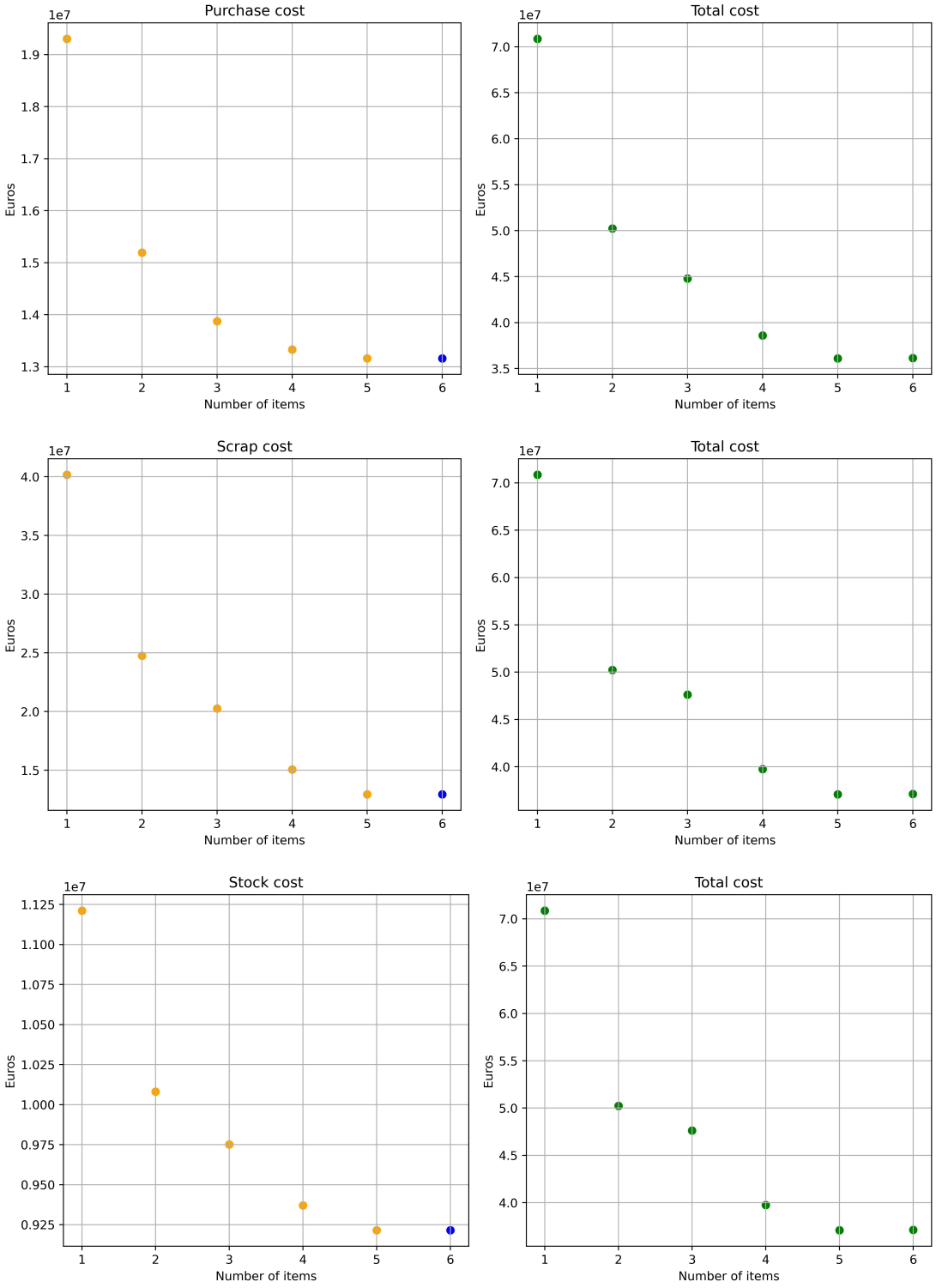

Figure 4.9: Case study 2. On the left, the image space for the models was obtained considering the purchase cost, the scrap cost, and the stock cost, respectively. On the right, is the corresponding total cost.

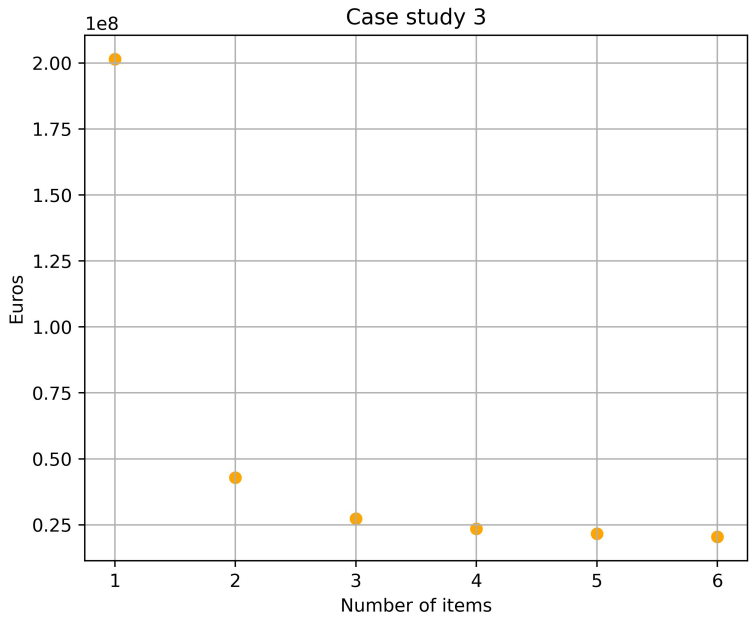

Figure 4.10: The image space for case study 3. The non-dominated set is depicted by the points highlighted in orange.

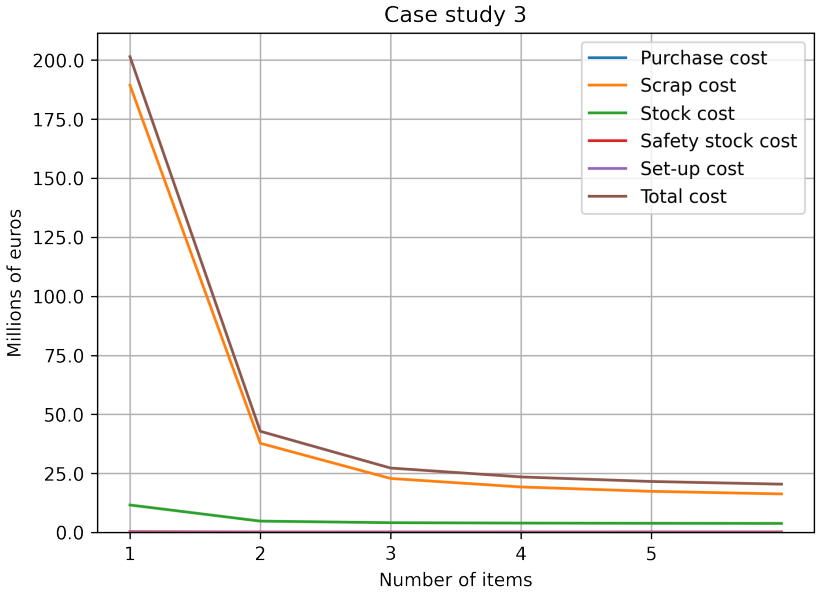

Figure 4.11: Cost trends as the number of items changes for scenario 3.

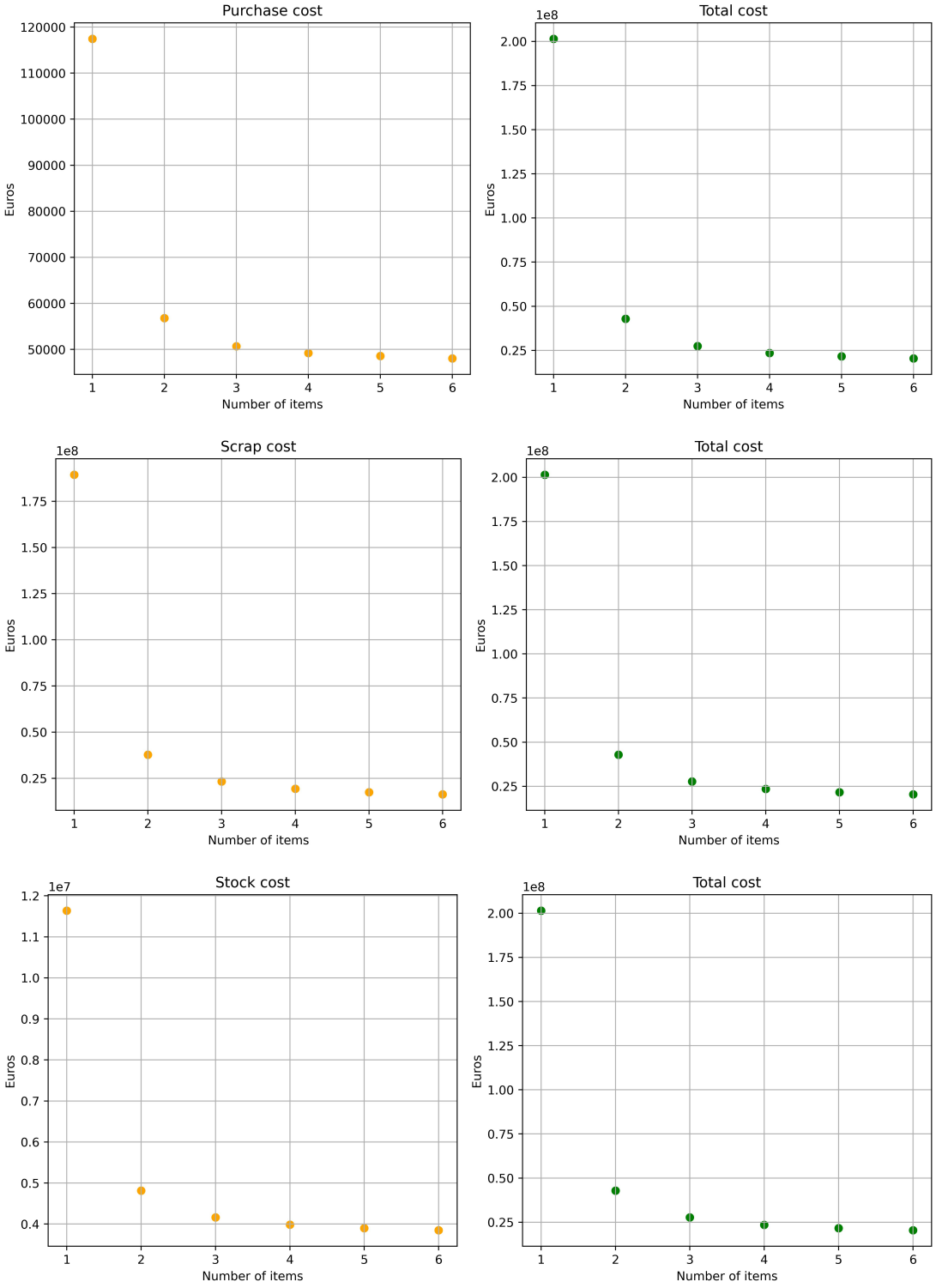

Figure 4.12: Case study 3. On the left, the image space for the models was obtained considering the purchase cost, the scrap cost, and the stock cost, respectively. On the right, is the corresponding total cost.

Chapter 5

Designing sustainable diet plans by solving triobjective integer programs*

It is the purpose of this chapter to define and solve triobjective integer models to design nutritionally adequate and sustainable diet plans. In particular, we look for diet plans for school cafeterias that simultaneously minimize their carbon, water and nitrogen footprints. The triobjective problems we want to solve have the following form:

$$\begin{aligned} \min \quad & (f_1(x), f_2(x), f_3(x))^T \\ \text{s.t.} \quad & x \in \mathcal{X} \cap \mathbb{Z}^n, \end{aligned} \tag{TOIP}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ and $f_1, f_2, f_3 : \mathbb{R}^n \to \mathbb{R}$ are continuous functions. Nonlinear functions can be handled by our approach, as long as they satisfy the so called positive γ property introduced in [32] which has already been recalled in Chapter 3. In Section 5.1, we present our method for triobjective integer nonlinear problems. We analyze its correctness and present a strategy to avoid the detection of weakly nondominated points. A comparison on linear and quadratic instances with other solvers is reported in Section 5.1.2. In Section 5.2, we present our application and two triobjective integer programs modeling the design of sustainable diet plans. We finally discuss the results obtained by solving the models using our algorithm. In Section 5.4, we draw some conclusions.

5.1 Algorithm `TrIntOpt`

The algorithm we propose extends the ideas used in [29], also described in Chapter 3, in order to define an exact criterion space method for triobjective nonlinear integer programs. Our algorithm, named `TrIntOpt`, is based on the ε-constraint method. The idea is to iteratively solve biobjective subproblems, defined by adding further constraints to the original feasible set. More precisely, given (TOIP), at every iteration k our method determines the nondominated set

*The content of this chapter is part of the paper [6].

of biobjective problems of the following form:

$$\begin{aligned} \min \quad & (f_1(x), f_2(x))^\top \\ \text{s.t.} \quad & f_3(x) \leq \varepsilon^k \\ & x \in \mathcal{X} \cap \mathbb{Z}^n, \end{aligned} \tag{BOIP^k}$$

where the parameter ε^k varies between $\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} f_3(x)$ and the value $f_3(\hat{x}^0) - \delta$, being δ a positive step size and $\hat{x}^0$ the point defined as follows. Among the efficient points of the biobjective problem $\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} (f_1(x), f_2(x))^\top$, $\hat{x}^0$ is one that achieves the maximum with respect to the objective function f_3. As it will be clarified later on, the step size δ controls the exacteness of TrIntOpt. The role of f_1, f_2 and f_3 in the definition of Problem (BOIP^k) can be interchanged.

For the definition of TrIntOpt we need to have an oracle able to detect the nondominated set of the biobjective nonlinear integer problem (BOIP^k):

Assumption 5.1.1. *There exists an oracle able to detect the complete nondominated set of Problem (BOIP^k) after having addressed a finite number B^k of single-objective integer programs. For each nondominated point $y \in \mathbb{R}^2$ detected, the oracle is able to compute one of its preimage, namely one efficient point $x \in \mathcal{X} \cap \mathbb{Z}^n$ such that $(f_1(x), f_2(x)) = y$.*

In the following, we denote by $\mathcal{E}^k$ the set of efficient points detected by the oracle in Assumption 5.1.1. We cite [29, 32] as works where algorithms satisfying Assumption 5.1.1 are defined. Furthermore, we need to assume the existence of the ideal point, in order to be guaranteed that the nondominated set is a finite set:

Assumption 5.1.2. *We assume that the ideal objective values $f_i^{\text{id}} := \min_{\mathcal{X} \cap \mathbb{Z}^n} f_i(x)$, $i = 1, 2, 3$, and thus the ideal point $f^{\text{id}} := (f_1^{\text{id}}, f_2^{\text{id}}, f_3^{\text{id}}) \in \mathbb{R}^3$, exists.*

We report in Algorithm 2 the scheme of our method TrIntOpt. TrIntOpt starts by computing $x^* \in \mathcal{X} \cap \mathbb{Z}^n$ as the minimum with respect to f_3 and $\mathcal{E}^0$ as the set of efficient points detected when addressing $\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} (f_1(x), f_2(x))^\top$. The images of points in $\mathcal{E}^0$ are nondominated points of Problem (TOIP) and define the set $\mathcal{M}^0$. The output of TrIntOpt, $\mathcal{M} \subseteq \mathbb{R}^3$, is initially set equal to $\mathcal{M}^0$. The starting ε^1 is set equal to $f_3(\hat{x}^0) - \delta$ and we enter in a loop. At every iteration k, the biobjective problem (BOIP^k) is handled, the set $\mathcal{E}^k$ and its image $\mathcal{M}^k$ are computed and $\mathcal{M}$ is enriched by the points in $\mathcal{M}^k$. As it is shown in Proposition 5.1.4, the points in $\mathcal{E}^k$ are at least weakly efficient. Then, the new value ε^{k+1} is set equal to $f_3(\hat{x}^k) - \delta$, being $\hat{x}^k$ a maximum with respect to f_3 over the set $\mathcal{E}^k$ and we go on until ε^k is less than $f_3(x^*)$, meaning that the whole criterion space has been visited.

In order to prove that TrIntOpt detects the complete nondominated set of Problem (TOIP), we need to assume that the objective functions are positive γ-functions, a concept that we already introduced in Section 3.1.

Assumption 5.1.3. *The functions $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, 2, 3$ in Problem (TOIP) are positive γ-functions as in Definition 3.1.3 for some $\gamma > 0$.*

As already mentioned in Chapter 3, linear or quadratic functions defined over $\mathbb{Q}^n$ are examples of functions satisfying Assumption 5.1.3. Let Assumption 5.1.3 hold with $\gamma > 0$. Let $\delta > 0$ be the input parameter for TrIntOpt. In case $\delta > \gamma$, TrIntOpt could miss some nondominated points of Problem (TOIP), since the step size δ may be wider than the distance between two nondominated points. On the other hand, if δ is chosen to be less than or equal to γ, we have that TrIntOpt is able to detect the complete nondominated set of (TOIP), as shown in the following.

Algorithm 2 Scheme of `TrIntOpt`

Input: (TOIP), $\delta > 0$, k = 1;
Output: Set $\mathcal{Y}_N \subseteq \mathcal{M} \subseteq \mathcal{Y}_{wN}$ of nondominated points of $(TOIP)$;

Compute $x^* \in \text{argmin}_{x \in \mathcal{X} \cap \mathbb{Z}^n} f_3(x)$
Compute $\mathcal{E}^0$ by addressing $\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} (f_1(x), f_2(x))^\top$
Compute $\hat{x}^0 \in \text{argmax}_{x \in \mathcal{E}^0} f_3(x)$
Set $\mathcal{M} = \mathcal{M}^0 = \{f(x) \mid x \in \mathcal{E}^0\}$
Set $\varepsilon^1 = f_3(\hat{x}^0) - \delta$
while $\varepsilon^k \geq f_3(x^*)$ **do**
    Compute $\mathcal{E}^k$ by addressing $\min_{x \in \mathcal{X}^k \cap \mathbb{Z}^n} (f_1(x), f_2(x))^\top$,
     with $\mathcal{X}^k = \mathcal{X} \cap \{x \in \mathbb{R}^n : f_3(x) \leq \varepsilon^k\}$
    Compute $\hat{x}^k \in \text{argmax}_{x \in \mathcal{E}^k} f_3(x)$
    Set $\mathcal{M}^k = \{f(x) \mid x \in \mathcal{E}^k\}$
    Set $\mathcal{M} = \mathcal{M} \cup \mathcal{M}^k$
    Set $\varepsilon^{k+1} = f_3(\hat{x}^k) - \delta$
    Set $k = k + 1$
end while
Return $\mathcal{M}$

In Proposition 5.1.4, we first prove that any point detected by `TrIntOpt` is at least a weakly nondominated point. Then, Proposition 5.1.5 shows that every nondominated point is detected at some iteration of `TrIntOpt`, so that no nondominated point is left undetected, meaning that the set $\mathcal{M}$, output of `TrIntOpt`, is a superset of the nondominated set $\mathcal{Y}_N$.

Proposition 5.1.4. *Let Assumption 5.1.1 and Assumption 5.1.2 hold. Let Assumption 5.1.3 hold with $\gamma > 0$ and assume that $\delta \leq \gamma$ in Algorithm 2. Let $\tilde{x} \in \mathcal{E}^k$. Then $f(\tilde{x}) \in \mathcal{Y}_{wN}$.*

Proof. Assume by contradiction that $f(\tilde{x}) \notin \mathcal{Y}_{wN}$, namely $x \in \mathcal{X} \cap \mathbb{Z}^n$ exists such that

$$f_i(x) < f_i(\tilde{x}) \quad i = 1, 2, 3. \tag{5.1}$$

Since $\tilde{x} \in \mathcal{X}^k \cap \mathbb{Z}^n$ we have that $f_3(\tilde{x}) \leq \varepsilon^k$. Therefore, $x \in \mathcal{X}^k \cap \mathbb{Z}^n$ and $f_3(x) \leq \varepsilon^k$, otherwise $f_3(\tilde{x}) < f_3(x)$, getting a contradiction to (5.1). Since $\tilde{x}$ is an efficient point for Problem (BOIP^k), we have that

$$\nexists\, \hat{x} \in \mathcal{X}^k \cap \mathbb{Z}^n \text{ such that } f_1(\hat{x}) \leq f_1(\tilde{x}), f_2(\hat{x}) \leq f_2(\tilde{x}),$$

with $f_i(x) \neq f_i(\tilde{x}) \quad i = 1, 2$; so that (5.1) cannot hold. □

Proposition 5.1.5. *Let Assumption 5.1.1 and Assumption 5.1.2 hold. Let Assumption 5.1.3 hold with $\gamma > 0$ and assume that $\delta \leq \gamma$ in Algorithm 2. Let $y \in \mathcal{Y}_N$. Then $k \in \mathbb{N}$ and $\tilde{x} \in \mathcal{E}^k$ exist such that $f(\tilde{x}) = y$.*

Proof. Let $k \in \mathbb{N}$ be the iteration of Algorithm 2 where it holds

$$\varepsilon^{k+1} < y_3 \leq \varepsilon^k. \tag{5.2}$$

Note that such value $k \in \mathbb{N}$ exists from the definition of ε^k within Algorithm 2 and since $f_3 : \mathbb{R}^n \to \mathbb{R}$ satisfies Assumption 5.1.3. Note also that since $y_3 \leq \varepsilon^k$, we have that $x \in \mathcal{X}^k \cap \mathbb{Z}^n$ exists such that $y = f(x)$. Assume by contradiction that $x \notin \mathcal{E}^k$. Two possibilities need to be considered:

i) if we assume that $x \notin \mathcal{E}^k$ as it is not an efficient point for (BOIP^k), we have that $(y_1, y_2) = (f_1(x), f_2(x))$ does not belong to the nondominated set of Problem (BOIP^k). Then, $\hat{x} \in \mathcal{X}^k \cap \mathbb{Z}^n$ exists such that

$$f_1(\hat{x}) \leq y_1,\ f_2(\hat{x}) \leq y_2,$$

with $f_i(\hat{x}) \neq y_i,\ i = 1, 2$. Since $y \in \mathcal{Y}_N$ it must hold

$$y_3 < f_3(\hat{x}) \leq \varepsilon^k.$$

Since Assumption 5.1.3 holds and $\delta \leq \gamma$, necessarily $y_3 \leq \varepsilon^{k+1}$, that is a contradiction to (5.2),

ii) if we assume that $x \notin \mathcal{E}^k$ as it has not been detected by the Oracle satisfying Assumption 5.1.1 used within Algorithm 2, we would have that $(f_1(x), f_2(x))$ belongs to the nondominated set of Problem (BOIP^k) but $\bar{x} \in \mathcal{E}^k$, $\bar{x} \neq x$ exists such that $(f_1(\bar{x}), f_2(\bar{x})) = (f_1(x), f_2(x)) = (y_1, y_2)$. From Proposition 5.1.4, it holds that $f(\bar{x}) \in \mathcal{Y}_{wN}$. Then, since $y \in \mathcal{Y}_N$, we have that $y_3 = f_3(x) \neq f_3(\bar{x})$ only if $y_3 = f_3(x) < f_3(\bar{x})$. As before, necessarily $y_3 \leq \varepsilon^{k+1}$ and we get a contradiction to (5.2).

□

Based on the previous lemmata we are able to prove the following.

Theorem 5.1.6. *Let Assumption 5.1.1 and Assumption 5.1.2 hold. Let Assumption 5.1.3 hold with $\gamma > 0$. Let $\delta \leq \gamma$. Algorithm 2 finds the complete nondominated set $\mathcal{Y}_N$ of (TOIP) after having addressed a finite number of single-objective integer programs.*

Proof. By Proposition 5.1.4 and Proposition 5.1.5 we have $\mathcal{Y}_N \subseteq \mathcal{M}$. Thanks to Assumption 5.1.3, choosing $\delta \in (0, \gamma]$ allows the `while` loop to take at most $m = \left\lfloor \left(f_3(\hat{x}^0) - f_3(x^*)\right)/\gamma \right\rfloor$ iterations. Furthermore, taking into account Assumption 5.1.1, we have that $\mathcal{M}^k$ can be detected after having solved B^k single-objective integer programs.

Then, considering the single-objective integer programs tackled at the beginning of Algorithm 2 for the computation of x^* and $\mathcal{M}^0$, that is $B^0 + 1$, the total number of single objective integer programs addressed by Algorithm 2 is $B^0 + \sum_{k=1}^{m} B^k + 1$. □

5.1.1 Avoiding the detection of weakly nondominated points

Starting from the ideas presented in [63], we propose to modify Problem (BOIP^k) addressed at Step 7 of Algorithm 2. An additional nonnegative continuous variable $s \in \mathbb{R}$ is introduced to avoid the detection of weakly nondominated solutions along the iterations of `TrIntOpt`. The biobjective nonlinear mixed-integer problem we consider is as follows:

$$\begin{aligned} \min \quad & (f_1(x) - \rho s, f_2(x) - \rho s)^\top \\ \text{s.t.} \quad & f_3(x) + s = \varepsilon^k \\ & x \in \mathcal{X} \cap \mathbb{Z}^n \\ & s \geq 0, \end{aligned} \tag{BOMIP^k}$$

where $\rho > 0$ is an adequately small number (usually between 10^{-3} and 10^{-6}) (see [63]). Despite Problem (BOMIP^k) is a mixed-integer problem, its nondominated set is finite and can be detected by the same oracle used for addressing Problem (BOIP^k). We denote by $\tilde{\mathcal{E}}^k$ the set of efficient points detected by the oracle in Assumption 5.1.1, when solving Problem (BOMIP^k).

Proposition 5.1.7. *Let Assumption 5.1.2 hold. Let Assumption 5.1.3 hold with $\gamma > 0$. Then, Problem (BOMIP^k) has a finite nondominated set. Furthermore, if $x \in \tilde{\mathcal{E}}^k$ then $f(x) \in \mathcal{Y}_N$.*

Proof. Recall that by $\mathcal{X}^k$ we denote the set $\mathcal{X} \cap \{x \in \mathbb{R}^n : f_3(x) \leq \varepsilon^k\}$. From Assumptions 5.1.2 and 5.1.3, we have that the set $\{f_3(x) \mid x \in \mathcal{X}^k \cap \mathbb{Z}^n\} \subset \mathbb{R}$ is a finite set. Furthermore, given $\bar{x} \in \mathcal{X}^k \cap \mathbb{Z}^n$, a unique $\bar{s}$ exists such that

$$\begin{aligned} \bar{s} = \quad & \text{argmax} \quad s \\ & \text{s.t.} \quad f_3(\bar{x}) + s = \varepsilon^k \\ & \qquad\quad s \geq 0. \end{aligned}$$

Therefore, the nondominated set of Problem (BOMIP^k) is finite. Let $x' \in \mathcal{E}^k$ be an efficient solution of (BOIP^k). From Proposition 5.1.4 we have that $f(x') \in \mathcal{Y}_{wN}$. Let $(\hat{x}, \hat{s}) \in \tilde{\mathcal{E}}^k$ be an efficient solution of (BOMIP^k) and assume by contradiction that x' dominates $\hat{x}$, with

$$f_1(x') = f_1(\hat{x}) \quad \text{and} \quad f_2(x') = f_2(\hat{x}).$$

Then, $f_3(x') < f_3(\hat{x}) = \varepsilon^k - \hat{s}$, with $\hat{s} \geq 0$. This implies that $s' \geq 0$ exists such that $f_3(x') = \varepsilon^k - s'$ and $s' > \hat{s}$. However, this contradicts the efficiency of $(\hat{x}, \hat{s})$ for Problem (BOMIP^k), as (x', s') is feasible for (BOMIP^k), with

$$f_1(x') - \rho s' < f_1(\hat{x}) - \rho \hat{s} \quad \text{and} \quad f_2(x') - \rho s' < f_2(\hat{x}) - \rho \hat{s}.$$

□

5.1.2 Numerical comparisons

The performance of `TrIntOpt` strongly depends on the performance of the oracle satisfying Assumption 5.1.1 adopted. In our Python implementation of `TrIntOpt` we solve the biobjective subproblems (BOMIP^k) by the Frontier Partitioner Algorithm in its enhanced version (`FPA`*) presented in [29]. Assumption 5.1.1 is satisfied by `FPA`* and the number B^k of single-objective integer programs addressed is equal to $|\mathcal{Y}_N^k| + 2$, being $|\mathcal{Y}_N^k|$ the cardinality of the nondominated set of the subproblem (BOMIP^k). For how `TrIntOpt` works, it can happen that single-objective integer problems are addressed to detect nondominated points that have already been found. In order to avoid useless computations and save CPU time, we keep a list of the nondominated points detected along the iterations of `TrIntOpt` and use the corresponding efficient points to warmstart the solver of the single-objective integer problems. Within our Python implementation of `TrIntOpt` we use the MIP solver of GUROBI [50]. All experiments have been executed on an Intel(R) Xeon(R) Gold 6252N CPU running at 2.30GHz.
In the following, we compare our Python implementation of `TrIntOpt` over both linear and quadratic instances. `LSM` [9] and `QSM` [10] are chosen as state of the art algorithms for the comparison over linear instances, while `AdEnA` is chosen as an algorithm to compare with over quadratic instances.

Linear instances

We mention that both the algorithms described below are implemented in `C++` and rely on the IBM CPLEX Optimizer as MIP solver.

The *L-shape method* (`LSM`) [9] is an image space decomposition method. The algorithm holds a priority queue of rectangles to be explored, in the image space of two objective functions.

The method looks into the rectangles with the aim of finding all those nondominated points for (TOIP) having their projection in the rectangle itself. In case an already-found nondominated point outside the rectangle is detected the rectangle is shrunk. The search can either determine that the rectangle contains no as yet unknown nondominated point so that the rectangle is discarded, or find a new nondominated point having its projection in the rectangle. In this case, the nondominated point found induces an "L-shape" which is explored in the same way, namely it can be shrunk, discarded or split into more rectangles that are added to the priority queue.

The *Quadrant Shrinking Method* (QSM) [10] partitions the image space by splitting it into "quadrants" which are defined starting from an upper bound point in the image space of two out of three objective functions. A search procedure looks for nondominated points over the current quadrant. Then, if no nondominated point is found the quadrant is shrunk. On the other hand, when a nondominated point is detected, a new search region is defined and the search procedure looks for an as yet unknown nondominated point. Strategies to avoid the solution of redundant single-objective integer problems are implemented.

In Table 5.1, we report the comparison among $\texttt{TrIntOpt}_{ILP}$, i.e. TrIntOpt without the strategy proposed in Section 5.1.1, TrIntOpt, LSM and QSM on instances of the triobjective assignment problem (AP) available at
https://usf.app.box.com/s/ds0g3ktcjg7vgfsbegzu53ptjgsfq1jg.
Such instances are 0-1 linear and the functions involved have all integer entries, so that Assumption 5.1.3 for TrIntOpt is satisfied with $\gamma = 1$.

For each instance and each method, we report the time needed in seconds and the cardinality of the output set, namely the cardinality of $\mathcal{M}$ for both $\texttt{TrIntOpt}_{ILP}$ and TrIntOpt and the cardinality of $\mathcal{Y}_N$ for LSM and QSM. In fact, despite the adoption of the strategy described in Section 5.1.1, we noticed that TrIntOpt may also detect weakly nondominated points. From our numerical experience, this depends on the choice of the parameter ρ within (BOMIP^k). The results shown are obtained with $\rho = 10^{-6}$, that, in our experiments, leads to the lowest number of additional weakly nondominated points. We can also notice that solving (BOMIP^k), instead of (BOIP^k), comes at a not negligible computational time, as TrIntOpt is always slower with respect to $\texttt{TrIntOpt}_{ILP}$. This could depend on both the value of ρ and on the introduction of the additional variable s in (BOMIP^k). QSM is the fastest method on every instance but AP5 and AP8, where $\texttt{TrIntOpt}_{ILP}$ is the fastest. Furthermore, despite $\texttt{TrIntOpt}_{ILP}$ detects a higher number of solutions, it is faster than LSM on seven instances, suggesting that the warmstarting strategy may pay off. The discussed results suggest that TrIntOpt is a valid method for solving TOIP linear problems.

Table 5.1: Results on triobjective assignment problem instances

Instance	TrIntOpt_{ILP}		TrIntOpt		LSM		QSM	
	time	$\lvert\mathcal{M}\rvert$	time	$\lvert\mathcal{M}\rvert$	time	$\lvert\mathcal{Y}_N\rvert$	time	$\lvert\mathcal{Y}_N\rvert$
AP2	17.18	223	19.20	221	14.46	221	10.75	221
AP3	45.69	500	53.43	488	41.54	483	31.92	483
AP4	260.30	2030	340.38	1962	264.10	1942	226.66	1942
AP5	484.99	3872	617.47	3765	617.29	3750	530.74	3750
AP6	936.99	5380	1097.05	5231	1059.81	5195	874.00	5195
AP7	2169.03	10996	2913.69	10546	2545.78	10498	2231.19	10498
AP8	3289.97	15373	4693.06	14809	4068.96	14733	3552.93	14733
AP9	6971.05	25684	10142.39	24021	7315.51	23941	6626.87	23941
AP10	9437.76	30802	15100.32	29259	10313.50	29192	9337.88	29193

Quadratic instances

In the following, we report a comparison of `TrIntOpt` with `AdEnA`, the hybrid decision-criterion space method proposed in [44, Algorithm 3] on tri-objective integer nonlinear instances, with convex quadratic objective functions and linear constraints. We used the code of `AdEnA` provided on GitHub [43]. `AdEnA` is an exact method for Multi-Objective Mixed-Integer Nonlinear Problems (MOMINLPs), meaning that it is able to detect an approximation of the nondominated set of a MOMINLP according to a prescribed precision. In particular, `AdEnA` is able to compute an *enclosure* of the nondominated set, i.e. a well-structured set in the image space, as for example a union of boxes, which contains the nondominated set as a subset. The precision of an enclosure as an approximation of the nondominated set is given by its *width* (we refer to [40] for further details on the concept of enclosure). The precision required to `AdEnA` cannot be set to zero, so that the approximation of the nondominated set obtained by `AdEnA` for purely integer instances cannot be compared - in terms of quality - with the exact nondominated set detected by `TrIntOpt`, that is a finite set. We chose to set ϵ, the parameter controlling the width of the enclosure released by `AdEnA`, equal to 0.01. The instances were randomly generated with a number of variables $n = 15$, a number of constraints $m = 10$. The matrices defining the quadratic terms in the objective functions were built using the MATLAB function `sprandsym` considering three different density levels $\rho \in \{0.25, 0.50, 0.75\}$. Namely, we generated matrices with approximately $\rho \cdot n^2$ nonzeros entries. For what concerns the linear inequality constraints $Ax \leq b$, we randomly generated matrix $A \in \mathbb{R}^{m \times n}$ and vectors $b \in \mathbb{R}^m$ with $m = 10$ using the MATLAB functions `sprand` and `rand` respectively. For each ρ we produced 5 different instances and we report the CPU time (in seconds) needed by the two algorithms in Table 5.2. On the instances considered, `TrIntOpt` is not only able to reproduce the finite nondominated set, but it is also almost three times faster than `AdEnA`. This suggests that `TrIntOpt` is a suitable algorithm for solving triobjective integer non linear problems with convex quadratic objective functions.

Table 5.2: Results on randomly generated triobjective quadratic instances

instance	ρ	`TrIntOpt` time	`AdEnA` time
1	0.25	25.69	113.26
2	0.25	39.01	102.50
3	0.25	55.67	206.12
4	0.25	64.98	460.29
5	0.25	61.48	439.78
6	0.50	180.97	693.57
7	0.50	14.22	86.52
8	0.50	47.80	195.50
9	0.50	11.48	85.94
10	0.50	229.15	364.82
11	0.75	36.11	164.87
12	0.75	41.23	137.59
13	0.75	106.97	354.59
14	0.75	169.75	404.57
15	0.75	268.47	807.92

5.2 Designing sustainable diet plans through triobjective 0-1 models

Sustainable diets are defined by the Food and Agriculture Organization of the United Nations [2] as "those diets with low environmental impact that contribute to food and nutrition security and to a healthy life for present and future generations". A sustainable diet must therefore be *healthy*, have low *environmental impact* and respect *cultural habits* in order to be acceptable to the population. These issues are often incompatible, for example low-cost diets correspond to high energy density, whereas diets of higher nutrient density and nutritional quality have higher costs. The *healthiness* of food is guaranteed by following the advices of nutritionists and various medical and governmental institutions, which mainly consist of dietary guidelines [68] defining nutrient requirements, recommended nutrient intakes as well as recommended consumption levels of some foods [83].

The *environmental impact* of food production refers to the level of greenhouse gas emissions, the use of land and water resources, pollution, phosphorus depletion and the impact of chemical products such as herbicides and pesticides. *Cultural habits*, i.e. the composition of meals, food preferences and preparation techniques, are strongly influenced by the traditions, beliefs and values shared by a community. They therefore define the structure of each meal and the set of foods and dishes that are considered edible and acceptable [48]. In addition, the attractiveness and variability of meals must also be considered when designing a diet. A meal plan, or menu, consists of the sequence and composition of daily meals over a given period of time. This can be done by selecting dishes from a given set of recipes with a portion size that generally depends on age, weight, gender, and level of physical activity. The design of a menu can therefore be modeled as the assignment of dishes (resources) to given places in a schedule (slots).

The case study we consider is the design of school lunch menus for primary schools in Italy. The set of dishes available to be served was determined by collecting a sample of several Italian primary school menus (children aged 6–11 years) and results in a list of 178 dishes grouped as 66 first-course dishes (in general including pasta or other carbohydrate sources), 75 second-course dishes (in general a source of protein), 35 side dishes (vegetables, potatoes, or salad), fresh fruit, and bread. Tap water is the only beverage allowed for lunch.

Complying with the Italian recommended dietary allowances (RDAs) [76], a fixed portion size for each dish was considered. Recommended ranges for energy and nutrient intakes (fats, proteins, carbohydrates, sugars, fiber, sodium, calcium, iron, and vitamin B12) for each lunch are also obtained from RDAs. To satisfy these requirements, the energy quantity q_i^{en} and nutrient quantities q_i^p of each dish i and nutrient p were calculated from its ingredients using the Italian Food composition and Nutrition database [1]. Other recommendations include limiting or avoiding consumption of some food groups and increasing consumption of others. For example, health authorities recommend eating more plant-based foods, limiting animal products, especially red meat, and avoiding processed meats. To reflect such recommendations, different food groups are defined (white meat, red meat, eggs, fish, dairy, and vegetables) and dishes are assigned to the appropriate food group g. In addition, dishes containing processed meat are not included in the list of available dishes.

The impact of food production on the environment was characterized by some standard consumption-based indicators, such as the carbon, water, and nitrogen footprints. The first is expressed as carbon dioxide equivalent and takes into account all the primary greenhouse gases, i.e., carbon dioxide CO_2, methane CH_4, and nitrous oxide N_2O, emitted during food production. The second takes into account for the freshwater withdrawals required to produce food and the last the pollution of water bodies and ecosystems due to the excess of nitrogen in agricultural production systems. The carbon and water footprints associated to each dish i, denoted by q_i^{cf}

and q_i^{wf} respectively, were computed from its ingredients using the database developed in the framework of the EU SU-EATABLE LIFE project [69]. The nitrogen footprint q_i^{nf} associated to each dish i was estimated from its ingredient using the model presented in [60].

We report in Figure 5.1 the carbon, water and nitrogen footprints of second-course dishes and side dishes containing vegetables. As expected, the second-course dishes containing red meat are the less sustainable ones.

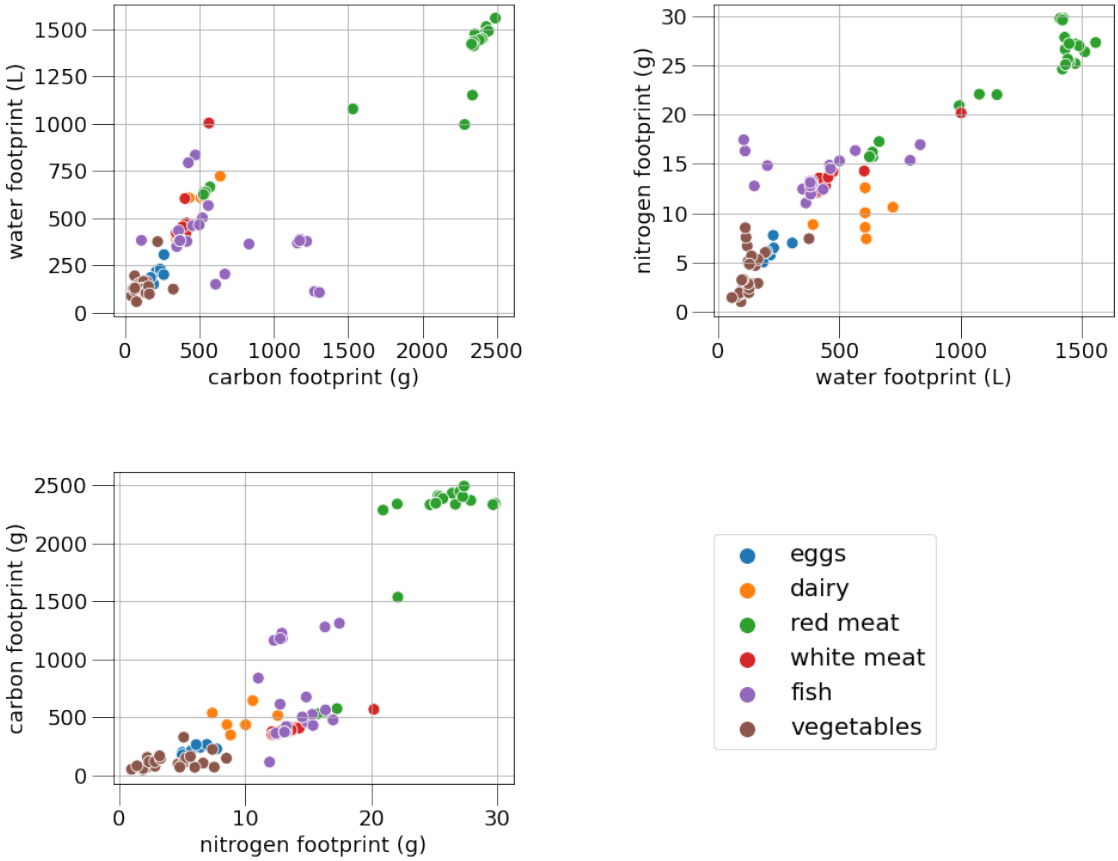

Figure 5.1: Carbon, water and nitrogen footprints of second-course dishes and side dishes containing vegetables (one fixed-size portion).

Cultural habits are naturally complied with the choice of the set of recipes from which to select the dishes of the menu, and with the structure of the lunch which must consists of one first-course dish, one second-course dish, one side-dish, fresh fruit and bread. On the other hand, attractiveness and variability of the menu is pursued by fixing the minimum and maximum number of times that each dish, and dishes of the same food group g, can be served in the menu.

The design of a lunch menu over D days can be modelled as the assignment of $D \times 178$ binary variables x_d^i, each one associated to each available dish i and day d, assuming value 1 if the dish i is served in the lunch of the day d, and 0 otherwise.

Nutritional recommendations consists of recommended nutrient intakes and can then be modeled as lower and/or upper bounds on energy and nutrients contents of each lunch and of the overall menu as follows:

$$L_p^{day} \leq \sum_{i=1}^{178} x_d^i \cdot q_i^p \leq U_p^{day}, \quad \forall d, p$$

$$L_p^{week} \leq \sum_{d=1}^{5} \sum_{i=1}^{178} x_d^i \cdot q_i^p \leq U_p^{week}, \quad \forall p \tag{5.3}$$

where L_p^{day}, U_p^{day}, L_p^{week}, and U_p^{week} are the lower and upper bounds on the recommended daily and weekly intake of nutrient p. The recommendations limiting or increasing the consumption of some food groups, as well as those related to the attractiveness and variability of the menu, can be modeled as lower and/or upper bounds on the number of dishes of the same food group g served during the week as follows:

$$m_g \leq \sum_{d=1}^{5} \sum_{i \in g} x_d^i \leq M_g, \quad \forall g \tag{5.4}$$

where m_g, and M_g are the lower and upper bound associated to group g. Moreover, variability is increased by imposing that each dish cannot be served more than once in the menu, that is

$$\sum_{d=1}^{D} x_d^i \leq 1, \quad \forall i \text{ (apart from fruit and bread)}$$

The values of the above defined lower and upper bounds are given in Tables 5.3 and 5.4.

Table 5.3: Lunch energy and nutrient constraints for children aged 6–11 years.

p	L_p^{day}	U_p^{day}	L_p^{week}	U_p^{week}
energy ($kcal$)	500	900	3000	4000
fats (g)	10	40	100	150
proteins (g)	15	40	100	175
carbohydrates (g)	70	130	450	550
sugars (g)	-	40	50	150
fiber (g)	-	20	25	75
sodium (mg)	100	700	1500	2500
calcium (mg)	-	-	1000	-
iron (mg)	-	-	25	-
vitamin B12 (μg)	0.35	-	-	-

The composition of the lunch of each day d is guaranteed by the following constraints:

$$\sum_{i \in first} x_d^i = 1, \quad \sum_{i \in second} x_d^i = 1, \quad \sum_{i \in side} x_d^i = 1, \quad \forall d$$

and $x_i^d = 1$, $\forall d$ and when i corresponds to fruit and bread.

Two different models are considered, with different objective functions and different number of days D. The first model refers to a weekly menu, i.e. $D = 5$, and the objective functions to be minimized are the carbon footprint (expressed as grams of carbon dioxide equivalent emitted),

Table 5.4: Food groups repetition constraints for health, acceptability and variability requirements.

g	m_g	M_g
white meat	1	2
red meat	-	1
eggs	1	2
fish	1	2
dairy	1	2
vegetables	4	-

the water footprint (expressed as liters of freshwater withdrawals), and the nitrogen footprint (expressed as grams of nitrogen released) of the menu, that is

$$f_1(x) = \sum_{d=1}^{D}\sum_{i=1}^{178} x_d^i \cdot q_i^{cf}, \quad f_2(x) = \sum_{d=1}^{D}\sum_{i=1}^{178} x_d^i \cdot q_i^{wf}, \quad f_3(x) = \sum_{d=1}^{D}\sum_{i=1}^{178} x_d^i \cdot q_i^{nf}. \tag{5.5}$$

This model results in a triobjective problem of 890 binary variables and 292 constraints 25 of which are equality constraints. As reported above, the objective functions and the constraints are linear and Assumption 5.1.3 is satisfied with $\gamma = 0.01$.

The second model refers to a menu for just two days ($D = 2$). In this case the weekly constraints (5.3) and (5.4) do not apply and the objective functions to be minimized are the carbon and water footprints of the menu, that is $f_1(x)$ and $f_2(x)$ in (5.5), and the mean square deviation of the daily energy intake with respect to the RDAs reference value of 700 $kcal/day$, that is

$$f_3(x) = \frac{1}{D}\sum_{d=1}^{2}\left[\left(\sum_{i=1}^{178} x_d^i \cdot q_i^{en}\right) - 700\right]^2.$$

The model is then a triobjective binary quadratic one, requiring a much heavier computational burden for our algorithm. This is the reason why the menu runs over just two days, so that only 356 binary variables are needed. Solving this triobjective model ended in identifying 5 nondominated solutions and the related 5 menus, that are reported in the table below. Note that every menu is also including bread and fruit for every day, which are not reported in Table 5.5.

5.3 Numerical results and discussion

For what concerns the first model described in Section 5.2, 635 weakly nondominated points were determined. Each point represents a set of menus with the same carbon, water, and nitrogen footprint values, and therefore equivalent with respect to the environmental impact. For example, the simple permutation of lunches within the days of the week, provides equivalent menus. Moreover, there are dishes sharing the same energy, nutrients, and environmental impact values that can be interchangeably used, and this may further increase the number of possible equivalent menus. This happens when different recipes have the same ingredients with the same quantity, i.e., they have only a different food preparation such as, for example, Backed potatoes, Boiled potatoes with olive oil, Crispy backed potatoes, and Sauteed potatoes. All the menus associated with the 635 points are nutritionally adequate, healthy and attractive, since they satisfy the constraints.

Table 5.5: Menus obtained solving the second model.

		First course	Second course	Side dish
menu 1	Day 1	Soup with rice and cabbage	Frittata	Backed potatoes
	Day 2	Creamy bean pasta	Baked anchovies with breadcrumbs	Green salad
menu 2	Day 1	Creamy potato barley soup	Baked anchovies with breadcrumbs	Cauliflower with anchovy sauce
	Day 2	Creamy bean pasta	Frittata	Stewed early potatoes
menu 3	Day 1	Creamy lentil pasta	Frittata	Stewed early potatoes
	Day 2	Creamy bean pasta	Baked anchovies with breadcrumbs	Spinach with olive oil
menu 4	Day 1	Creamy bean pasta	Frittata	Stewed early potatoes
	Day 2	Pasta with peas	Baked anchovies with breadcrumbs	Cabbage with tomato sauce
menu 5	Day 1	Creamy bean pasta	Baked anchovies with breadcrumbs	Spinach with olive oil
	Day 2	Creamy chickpea pasta	Frittata	Stewed early potatoes

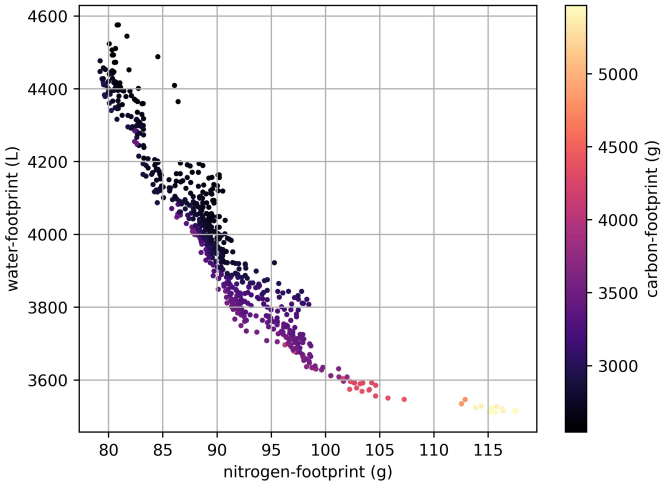

Figure 5.2: Projection of the weakly nondominated points on the plane nitrogen-water footprints.

Projections of weakly nondominated points on the three coordinate planes are shown in Figures 5.2, 5.3, and 5.4.

Figures 5.2, 5.3 show that menus with low/high nitrogen or carbon footprint are generally those with higher/lower water footprint. On the contrary, menus with high/low carbon footprint exhibit also high/low nitrogen footprint, see Figure 5.4. As a consequence, menus with high water footprint have low values for both carbon and nitrogen footprints, and menus with low water footprint have high values for both carbon and nitrogen footprint. This is clearly shown in Figure 5.4 checking the color mark for the water footprint values. An approximate quadratic relation between the footprint values associated to the (weakly) nondominated points found is obtained by least square fitting with $R^2 = 0.98$, and the corresponding surface is shown in Figure 5.5. Hence water footprint increases on average quadratically when carbon and nitrogen footprints decrease. This allows, for instance, to estimate the minimum water footprint of a menu fixing some values for the associated carbon and nitrogen footprint.

5.4 Conclusions

An exact method for detecting the nondominated set of triobjective nonlinear integer programs has been devised. The method `TrIntOpt` uses the concept of γ-positive function in order to properly combine the ε-constraint method with solvers for biobjective integer programs. A strategy to avoid the detection of weakly nondominated points, seen as a disadvantage of the ε-constraint method, is presented. By applying `TrIntOpt` to two triobjective 0-1 problems modeling the design of nutritionally adequate and healthy diet plans, we were able to collect menus, minimizing standard consumption-based indicators measuring the impact of food production on the environment. Thanks to the theoretical results proven we are guaranteed that each menu detected is a nondominated point of the problem considered and that the whole nondominated set has been recovered.

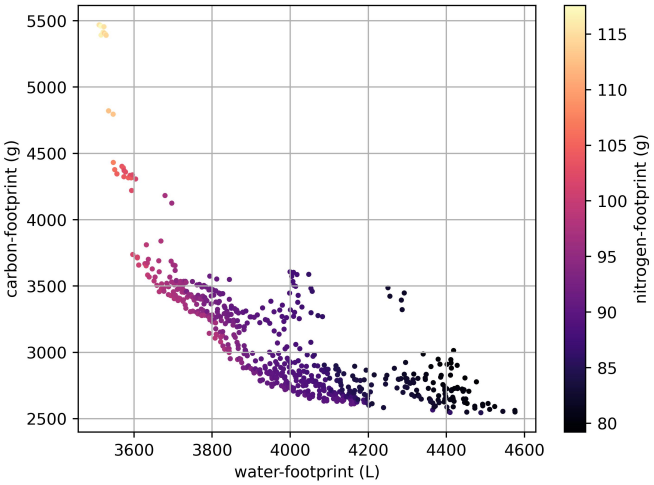

Figure 5.3: Projection of the weakly nondominated points on the plane water-carbon footprints.

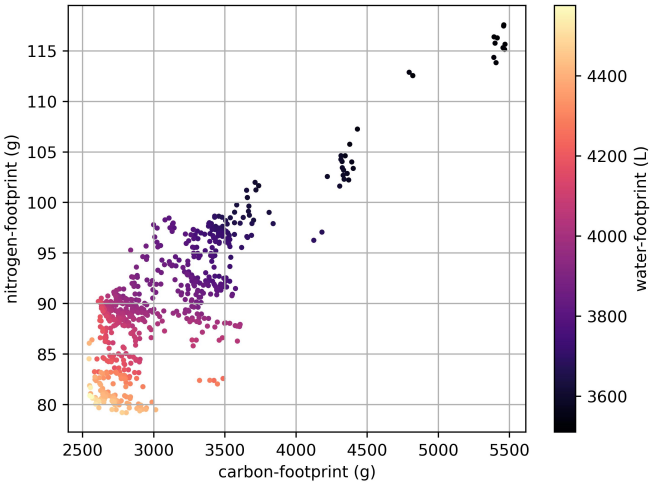

Figure 5.4: Projection of the weakly nondominated points on the plane carbon-nitrogen footprints.

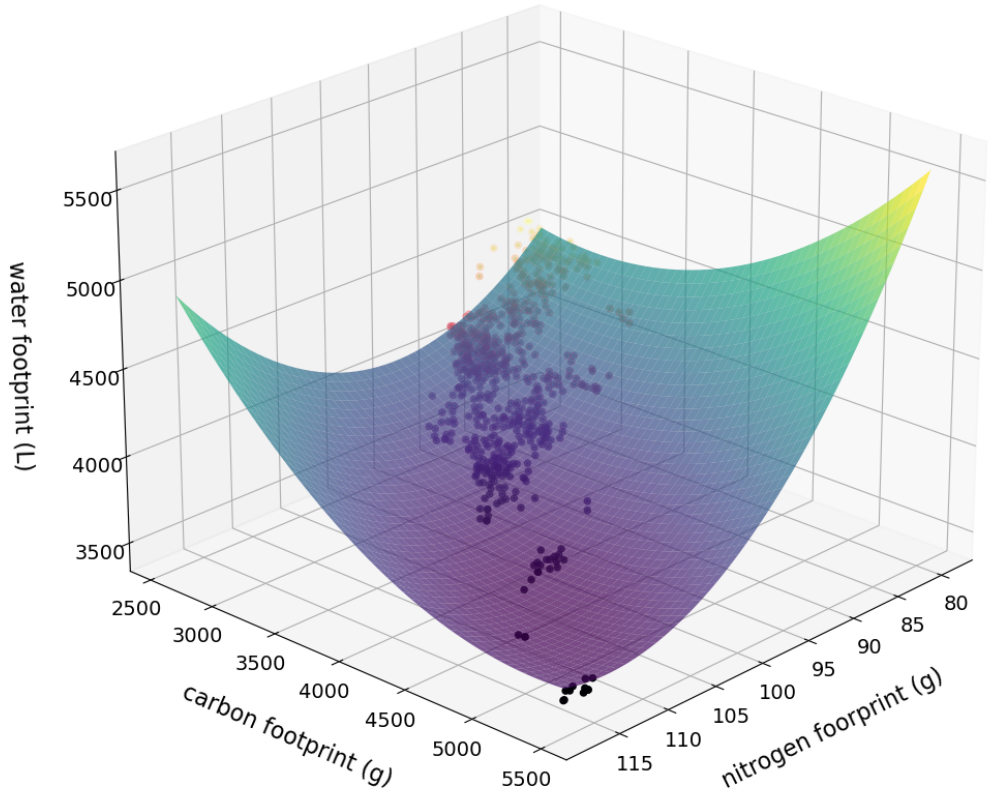

Figure 5.5: Quadratic approximate relation between weakly nondominated points ($R^2 = 0.98$).

Chapter 6

Using Dual Relaxations in Multiobjective Mixed-Integer Convex Quadratic Programming *

In this chapter, we develop a branch-and-bound method that, using dual relaxations as a key ingredient, is guaranteed to compute an enclosure for the nondominated set of a multiobjective mixed-integer convex quadratic programming problem in a finite number of iterations. Differently from approaches that work exclusively in the image space, our algorithm is also able to deliver a superset of the set of efficient integer assignments, that in turn is needed as input for some existing approaches (see, e.g., [19]). Here, an efficient integer assignment is a fixing of the integer variables in such a way that there exists an efficient point of the problem with exactly this fixing. Note that for multiobjective mixed-integer problems, it can happen that there is a large number of efficient integer assignments. In fact, there exist instances of such optimization problems that cannot be solved without a full enumeration of all integer assignments. The reason for that is that it can happen that all integer assignments contribute to different parts of the nondominated set. This clearly represents a big difference with respect to the single-objective case and at the same time a big challenge regarding the development of solution algorithms for MOMIPs. In the definition of branch-and-bound methods, this emphasizes the need for strategies for a fast enumeration of the nodes.

Starting from some of the ideas to deal with purely integer unconstrained problems presented in [27], we address the difficulties of handling the presence of both continuous and integer variables and linear inequality constraints. In contrast to the purely integer case, the nondominated set of the corresponding multiobjective mixed-integer nonlinear optimization problems is an, in general, infinite set and cannot be computed exactly. Consequently, solving such optimization problems usually refers to computing an approximation of the nondominated set, the efficient set, or both of them. Thus, different solution techniques are needed. The method that we present computes both a coverage of the nondominated set and a superset of the set of efficient integer assignments.

More formally, we focus on optimization problems where the goal is to minimize $m \geq 2$ quadratic objective functions given by $f_j \colon \mathbb{R}^n \to \mathbb{R}$,

$$f_j(x) = x^\top Q_j x + (c^j)^\top x + a_j,$$

for all $j \in [m] := \{1, \ldots, m\}$ with symmetric positive definite matrices $Q_1, \ldots, Q_m \in \mathcal{S}^n$, vectors

*The content of this chapter is part of the paper [31].

$c^1, \dots, c^m \in \mathbb{R}^n$, and scalars $a_1, \dots, a_m \in \mathbb{R}$. We recall that under the assumption that all the matrices Q_j are positive definite, we have that all the functions f_j are strongly convex. This means that there exist some $\nu > 0$ such that for all $x, y \in \mathbb{R}^n$ and $\lambda \in [0,1]$ it holds

$$\nu\lambda(1-\lambda)\|x-y\|_2^2 + f_j(\lambda\, x + (1-\lambda)\, y) \leq \lambda\, f_j(x) + (1-\lambda)\, f_j(y)$$

for $j \in [m]$, where with $\|\cdot\|_2$ we denote the Euclidean norm. This also implies the strict convexity of the functions, i.e., for any $x, y \in \mathbb{R}^n$ with $x \neq y$ and any $\lambda \in (0,1)$ we have $f_j(\lambda\, x + (1-\lambda)\, y) < \lambda\, f_j(x) + (1-\lambda)\, f_j(y)$ for $j \in [m]$. These convexity assumptions will be essential to derive the finiteness result for our new algorithm.

Assuming that the objective functions are quadratic allows us to perform many of the expensive calculations in a preprocessing phase. What is more important, we can make use of simple dual formulations within our procedure which are a main aspect to make our algorithm fast and efficient. Thus, in summary, compared to the algorithm presented in [27], the theory and the algorithm are extended to mixed-integer problems (compared to the pure integer case in [27]), to linearly constrained problems (compared to unconstrained problems in [27]), and we additionally speed up the algorithm by making use of dual formulations.

The multiobjective mixed-integer quadratic programming problem which we study in this chapter is given as

$$\begin{aligned} \min_{x} \quad & (f_1(x), \dots, f_m(x))^\top \\ \text{s.t.} \quad & Ax \leq b \\ & x_i \in \mathbb{Z} \text{ for all } i \in [k] \\ & x \in \mathbb{R}^n \end{aligned} \tag{MOMIQP}$$

with a matrix $A \in \mathbb{R}^{p \times n}$, a vector $b \in \mathbb{R}^p$, and with $1 \leq k \leq n$, i.e., we assume that at least one variable can attain integer values only. We do not need any assumption on the boundedness of the feasible set. In particular, we are not assuming any lower or upper bounds on the variables, i.e., no box constraints, as it is often required, for instance in [54] or for branch-and-bound based methods with a partitioning of the starting box in the pre-image space as in [28, 41]. Indeed, it will be later shown that both the efficient and the nondomianted sets of problem (MOMIQP) are bounded. In the following, the feasible set of (MOMIQP) is denoted by S, i.e.,

$$S := \{x \in \mathbb{Z}^k \times \mathbb{R}^{n-k} \mid Ax \leq b\}.$$

Note that for $k = n$ we have the special case of a multiobjective integer quadratic programming problem, for which our method will be exact. This means that it will be able to detect the whole nondominated set, which is a finite set.

This chapter is organized as follows. In Section 6.1, we give some standard definitions for multiobjective optimization and we formally recall what an enclosure of the nondominated set is. In Section 6.2 we define the subproblems that we address at the nodes of our branch-and-bound algorithm according to our branching strategy that works by fixing integer variables. In Section 6.3 we present the theoretical results that allow to avoid an infinite enumeration of nodes even in case problem (MOMIQP) has an unbounded feasible set. In Section 6.4 a scheme of our branch-and-bound algorithm is presented and in Section 6.5, we see how the dual relaxations may come into play to save computational effort. Eventually, in Section 6.6, numerical results are reported and in Section 6.7 we draw some conclusions.

6.1 Basic Notions and Definitions

For the following notions as well as an introduction to multiobjective optimization we refer, for instance, to [36]. We use the standard optimality notion based on the componentwise partial

ordering in the image space. A point $\bar{x} \in S$ is called an efficient point for (MOMIQP) if there is no feasible point $x \in S$ with $f(x) \neq f(\bar{x})$ and with $f(x) \leq f(\bar{x})$. Here and in the following, $\leq$ and $<$ are understood componentwise. The image $f(\bar{x})$ of an efficient point for (MOMIQP) is called nondominated, and the image set of all efficient points is denoted as the nondominated set $\mathcal{Y}_N$ (also known, specifically for $m = 2$, as Pareto front).

It is shown in the following that multiobjective optimization problems having strongly convex objective functions have a bounded efficient and nondominated set. Consider a multiobjective programming problem of the form

$$\begin{array}{ll} \min & (f_1(x), \ldots, f_m(x))^\top \\ \text{s.t.} & x \in \Omega \subseteq \mathbb{R}^n \end{array} \tag{6.1}$$

where $f_j : \mathbb{R}^n \to \mathbb{R}$ denotes a strongly convex function with parameter $\gamma_j > 0$ for all $j \in [m]$ and $\Omega \subseteq \mathbb{R}^n$ is a nonempty feasible set. The following results shows that the efficient set and the nondominated set of (6.1) are bounded sets.

Proposition 6.1.1. *Let $\mathcal{E}$ be the efficient set of (6.1). Then there exist $\bar{x}, \underline{x} \in \mathbb{R}^n$ such that $\mathcal{E} \subseteq \operatorname{int}(B_{\mathcal{E}}) =: (\bar{x}, \underline{x})$.*

Proof. Assume by contradiction that $(x^k)_{k \in \mathcal{Y}_N} \subseteq \mathcal{E}$ exists such that $\lim_{k\to\infty} \left\|x^k\right\|_2^2 = +\infty$. Let $\tilde{x} \in \Omega$ and let $\rho := f(\tilde{x})$. In particular, we have that

$$\lim_{k\to\infty} \left\|x^k - \tilde{x}\right\|_2^2 = +\infty. \tag{6.2}$$

Further, let $j \in [m]$. Since f_j is strongly convex, we have that

$$0.25\,\gamma_j \left\|x - \tilde{x}\right\|_2^2 + f_j(0.5(x + \tilde{x})) \leq 0.5 f_j(x) + 0.5 f_j(\tilde{x})$$

holds for all $x \in \mathbb{R}^n$ or, equivalently,

$$0.5\,\gamma_j \left\|x - \tilde{x}\right\|_2^2 + 2 f_j(0.5(x + \tilde{x})) - f_j(\tilde{x}) \leq f_j(x). \tag{6.3}$$

Furthermore, let x_j^* denote the unique minimizer of f_j over $\mathbb{R}^n$. From (6.3) and since $f_j(x) \geq f_j(x_j^*)$ for all $x \in \mathbb{R}^n$, we have that for all $k \in \mathcal{Y}_N$

$$f_j(x^k) \geq 0.5\,\gamma_j \left\|x^k - \tilde{x}\right\|_2^2 + 2 f_j(x_j^*) - f_j(\tilde{x}).$$

Therefore, from (6.2), it necessarily holds $\lim_{k\to\infty} f_j(x^k) = +\infty$ for all $j \in [m]$. In particular, for sufficiently large $k \in \mathcal{Y}_N$ we have that $f(x^k) > \rho = f(\tilde{x})$, contradicting that $(x^k)_{k \in \mathcal{Y}_N} \subseteq \mathcal{E}$. □

As a direct consequence from Proposition 6.1.1 we obtain the following:

Corollary 6.1.2. *Let $\mathcal{Y}_N$ be the nondominated set of (6.1). Then there exist $z, Z \in \mathbb{R}^m$ such that $\mathcal{Y}_N \subseteq \operatorname{int}(B_{\mathcal{Y}_N}) =: (z, Z)$.*

Thanks to Corollary 6.1.2, we have that the nondominated set of a MOMIP problem with strongly convex objective functions is a bounded set. In particular, due to our assumptions, this holds for our problem (MOMIQP). Hence, it is guaranteed that a closed box

$$B := [z, Z] := \{x \in \mathbb{R}^m \mid z \leq x \leq Z\} \tag{6.4}$$

with $\mathcal{Y}_N \subseteq \operatorname{int}(B) = (z, Z) := \{x \in \mathbb{R}^m \mid z < x < Z\}$, where $z, Z \in \mathbb{R}^m$, always exists. As already mentioned, we aim at approximating the nondominated set $\mathcal{Y}_N$ by an *enclosure* that can be defined as follows (see [44]).

Definition 6.1.3. *Let $\mathcal{L},\mathcal{U} \subseteq \mathbb{R}^m$ be two finite sets with $\mathcal{Y}_N \subseteq \mathcal{L}+\mathbb{R}^m_+$ and $\mathcal{Y}_N \subseteq \mathcal{U}-\mathbb{R}^m_+$. Then $\mathcal{L}$ is called a lower bound set, $\mathcal{U}$ is called an upper bound set, and the set $\mathcal{A}$ which is given as*

$$\mathcal{A} = \mathcal{A}(\mathcal{L},\mathcal{U}) := (\mathcal{L}+\mathbb{R}^m_+) \cap (\mathcal{U}-\mathbb{R}^m_+) = \bigcup_{l\in\mathcal{L}} \bigcup_{\substack{u\in\mathcal{U},\\ l\leq u}} [l,u]$$

is called an enclosure (or a box approximation) of the nondominated set $\mathcal{Y}_N$ of (MOMIQP) given $\mathcal{L}$ and $\mathcal{U}$.

Note that for the elements of the set $\mathcal{U}$ one cannot take just objective function values $f(x)$ of feasible points $x \in S$, as one might be used to from single-objective global optimization. Instead we need another concept, for instance the one of so-called local upper bounds, which we will introduce below. A lower bound set $\mathcal{L}$ can be computed as a union of ideal points of certain subproblems of (MOMIQP) which we discuss in the forthcoming Section 6.2.

The quality of an enclosure $\mathcal{A}$ is given by its width $w(\mathcal{A})$. It is defined in [40] as the optimal value of

$$\max_{l,u} \; s(l,u) \quad \text{s.t.} \quad l \in \mathcal{L},\; u \in \mathcal{U},\; l \leq u$$

where $s(l,u) := \min\{u_i - l_i \mid i \in [m]\}$ denotes the shortest edge length of a box $[l,u]$. The surprising fact that this quality measure is based on the shortest edge length of the boxes, and not on the largest as typically expected in global optimization, is due to a desired relation to ε-optimality. For $\varepsilon > 0$ a point $\bar{x} \in S$ is called ε-efficient for (MOMIQP) if there exists no $x \in S$ with $f(x) \neq f(\bar{x}) - \varepsilon e$ and $f(x) \leq f(\bar{x}) - \varepsilon e$, where e represents the all-ones vector. We denote the image set of all ε-efficient points by $\mathcal{Y}_{N\varepsilon}$. According to Lemma 3.1 in [40], if $\mathcal{A}$ is an enclosure of $\mathcal{Y}_N$ with $w(\mathcal{A}) < \varepsilon$ then any $x \in S$ with $f(x) \in \mathcal{A}$ is at least ε-efficient for (MOMIQP). In other words, $\mathcal{A} \cap f(S) \subseteq \mathcal{Y}_{N\varepsilon}$ holds. This is the natural extension of ε-optimality as used in single-objective global optimization. A more detailed discussion and extensive motivation for this quality measure is provided in [40, 42].

As already mentioned, and widely discussed in the literature, a proper concept to obtain an upper bound set $\mathcal{U}$ are the so-called local upper bounds which have been presented in [58]. In the following definition, we use the generalized definition of local upper bounds from [44, Def. 4.1] with the set B from (6.4) as the so-called initial area of interest. Within the definition we further make use of stable sets. These are sets $N \subseteq \mathbb{R}^m$ where for any two points $y^1, y^2 \in N$ with $y^1 \neq y^2$ it holds that $y^1 \not\leq y^2$, i.e., all elements of N are pairwise non-comparable.

Definition 6.1.4. *Let $N \subseteq \mathbb{R}^m$ be a finite and stable set. Then the lower search region for N is $s(N) := \operatorname{int}(B) \setminus (N + \mathbb{R}^m_+)$ and the lower search zone for some $u \in \mathbb{R}^m$ is $c(u) := \{y \in \operatorname{int}(B) \mid y < u\}$. A set $\mathcal{U} = U(N) \subseteq \mathbb{R}^m$ is called local upper bound set given N if $s(N) = \bigcup_{u\in U(N)} c(u)$ and if $\{u^1\} - \operatorname{int}(\mathbb{R}^m_+) \not\subseteq \{u^2\} - \operatorname{int}(\mathbb{R}^m_+)$ for all $u^1, u^2 \in U(N)$, $u^1 \neq u^2$. Each point $u \in U(N)$ is called a local upper bound (LUB).*

We remark that the set B within Definition 6.1.4 does not necessarily have to be a box as in (6.4), but can be chosen as an arbitrary subset of $\mathbb{R}^m$ with $\mathcal{Y}_N \subseteq \operatorname{int}(B)$, see [44, Assumption 4.3].

6.2 Building Subproblems by Fixing Variables

The algorithm we propose is a branch-and-bound method that works by fixing the values of certain integer variables. Our enumeration strategy is depth-first and we always branch by fixing the value of one of the k integer variables. A crucial property of our algorithm is that the set

of fixed variables only depends on the depth of the node in the branch-and-bound tree. We thus lose the flexibility of choosing the best branching variable, but this strategy allows us to process a single node in the tree much faster. As it will be clarified later, this branching strategy, already successfully used in single-objective mixed-integer optimization [14, 15, 12, 13], allows to perform a preprocessing phase for faster computations at the nodes. More precisely, at a generic level $d \in [k] \cup \{0\}$ of the branch-and-bound tree, the variables $x_1, \dots, x_d$ are fixed to certain integer values, say, $r_1, \dots, r_d \in \mathbb{Z}$. In particular, the order in which integer variables are fixed is predetermined. This means that we start by fixing the value of x_1 at level $d = 1$, continue by fixing the value of x_2 at level $d = 2$, and so on, until we fix the last integer variable x_k at level $d = k$. Hence, at every node of the branch-and-bound tree at the same level $d \in [k] \cup \{0\}$ the same set of integer variables is fixed to certain (different) values. We remark that at level $d = 0$ (root node) no variable is fixed and we have the original problem. The full algorithmic scheme of our method is reported in Section 6.4.

Let $r = (r_1, \dots, r_d)^\top \in \mathbb{Z}^d$ be a vector of integer fixings. This vector defines a specific node at level d of the branch-and-bound tree. For every $j \in [m]$, we define, as in [27, Lemma 3.1], the function $f_j^r : \mathbb{R}^{n-d} \to \mathbb{R}$ by $f_j^r(x) := f_j(r_1, \dots, r_d, x_1, \dots, x_{n-d})$. This function can be expressed explicitly as

$$f_j^r(x) = x^\top Q_j^d x + (c^{j,r})^\top x + a_{j,r},$$

where the positive definite symmetric matrix Q_j^d is obtained by deleting the corresponding d rows and columns of Q_j and $c^{j,r}$ and $a_{j,r}$ are set to

$$c_{i-d}^{j,r} := c_i^j + 2\sum_{l=1}^{d} q_{li} r_l, \quad \text{for} \quad i = d+1, \dots, n$$

and

$$a_{j,r} := a_j + \sum_{l=1}^{d} c_l r_l + \sum_{l=1}^{d}\sum_{i=1}^{d} q_{li} r_l r_i.$$

Similarly, we define the matrix $A^d \in \mathbb{R}^{p \times (n-d)}$ and the vector $b^r \in \mathbb{R}^p$ by taking into account the fixings, i.e., A^d denotes the matrix which is obtained from A by deleting the first d columns and $b^r := b - A(r_1, \dots, r_d, 0, \dots, 0)^\top$.

We consider the following continuous relaxation of (MOMIQP) induced by that fixing $r \in \mathbb{Z}^d$ of the first d integer variables:

$$\begin{array}{ll} \min\limits_{x} & (f_1^r(x), \dots, f_m^r(x))^\top \\ \text{s.t.} & A^d x \leq b^r \\ & x \in \mathbb{R}^{n-d}. \end{array} \tag{MOP^r}$$

In our method, we mainly use these continuous subproblems to compute a lower bound set $\mathcal{L}$ for an enclosure of the nondominated set of (MOMIQP) and to check whether the node corresponding to the fixing $r \in \mathbb{Z}^d$ of integer variables can be pruned. In fact, we do not consider (MOP^r) directly, but an outer approximation of the corresponding upper image set

$$\mathcal{P}^r := \{f^r(x) \in \mathbb{R}^m \mid A^d x \leq b^r,\ x \in \mathbb{R}^{n-d}\} + \mathbb{R}^m_+.$$

The simplest outer approximation is determined by the ideal point of this set, which is componentwise calculated as $\min\{y_j \in \mathbb{R} \mid y \in \mathcal{P}^r\}$ for $j \in [m]$. By our assumptions, these minima exist in case the feasible set $S^r := \{x \in \mathbb{R}^{n-d} \mid A^d x \leq b^r\}$ of (MOP^r) is nonempty. This approximation using the ideal point corresponds to an outer approximation derived by m

supporting hyperplanes to the set $\mathcal{P}^r$ with normal vectors equal to the m unit vectors. As this outer approximation is very rough, we allow improved outer approximations. In this respect, let $L \subseteq \{y \in \mathbb{R}^m_+ \mid \|y\|_1 = 1\}$ be a finite set of nonnegative vectors which includes all m unit vectors. We decided here for the 1-norm but any other norm can also be used for normalizing the vectors. This set defines the hyperplanes which are used for the outer approximation of $\mathcal{P}^r$. The derived approximation of the upper image set $\mathcal{P}^r$ will be involved in our pruning condition which we define later.

In order to compute the outer approximation, at a node, we solve the $|L|$ continuous single-objective subproblems

$$\begin{array}{ll} \min\limits_{x} & \ell^\top f^r(x) \\ \text{s.t.} & A^d x \leq b^r \\ & x \in \mathbb{R}^{n-d}. \end{array} \tag{$P^r(\ell)$}$$

In fact, we will examine the dual problems of these subproblems, see Section 6.5. In case problem ($P^r(\ell)$) is infeasible, i.e., in case we have

$$S^r = \{x \in \mathbb{R}^{n-d} \mid A^d x \leq b^r\} = \emptyset \ , \tag{Inf}$$

the node can be pruned, see the results in Section 6.3.1.

Otherwise, in case problem ($P^r(\ell)$) is feasible, we define $\varphi^r(\ell)$ to be its optimal value for $\ell \in L$. Moreover, we denote by $x^{*\ell,r} \in \mathbb{R}^{n-d}$ its unique minimizer, which exists due to the strong convexity of the objective function. Note that for $\ell = e^j$, the j-th unit vector, we minimize $\ell^\top f^r(\cdot) = f^r_j(\cdot)$. Hence, in that case $x^{*\ell,r}$ denotes the unique minimizer of f^r_j with respect to S^r. In particular, $\varphi^r(e^j) = f^r_j(x^{*e^j,r})$ gives the j-th component of the ideal point of the set $\mathcal{P}^r$.

Furthermore, in case of feasibility, i.e., in case $S^r \neq \emptyset$, we define

$$\alpha(d+1) := \min_{\ell \in L} x_1^{*\ell,r}, \qquad \beta(d+1) := \max_{\ell \in L} x_1^{*\ell,r} \ , \tag{6.5}$$

and the interval

$$[\,\lfloor \alpha(d+1) \rfloor\,,\, \lceil \beta(d+1) \rceil\,] \ . \tag{I}$$

Recall that the vector $r \in \mathbb{Z}^d$ of integer fixings corresponds to a node at level d of the branch-and-bound tree. In case $d < k$, the interval (I) basically defines the range of values $r'_{d+1} \in \mathbb{Z}$ for which, within our algorithm, children nodes with x_{d+1} fixed to $r'_{d+1} \in \mathbb{Z}$ need to be considered. More importantly, we will show in Lemma 6.3.10 that all children nodes corresponding to $r' := (r_1, \ldots, r_d, r'_{d+1}) \in \mathbb{Z}^{d+1}$ with r'_{d+1} outside that interval and far enough can safely be pruned. This is one of the key results that ensures finiteness of the overall algorithm.

In the following, we briefly explain how exactly the children nodes at level $d+1 \leq k$ corresponding to such vectors $r' \in \mathbb{Z}^{d+1}$ of integer fixings are explored within our algorithm. At the first child node, x_{d+1} is fixed to $r'_{d+1} = \lfloor \alpha(d+1) \rfloor$. Then its sibling nodes are computed by consecutively fixing x_{d+1} to increasing integer values $r'_{d+1} \in \{\lfloor \alpha(d+1) \rfloor + 1, \lfloor \alpha(d+1) \rfloor + 2, \ldots, \lceil \beta(d+1) \rceil\}$. The method continues with fixing x_{d+1} to increasing integer values $r'_{d+1} > \lceil \beta(d+1) \rceil$ until it reaches the first assignment of r'_{d+1} for which the node corresponding to $r' \in \mathbb{Z}^{d+1}$ can be pruned by one of the conditions we present in the forthcoming Section 6.3. Since that implies that also all children nodes with even larger values of the integer variable x_{d+1} can be pruned, the algorithm continues by exploring those nodes corresponding to fixings of $r'_{d+1} < \lfloor \alpha(d+1) \rfloor$. Again, starting from $\lfloor \alpha(d+1) \rfloor - 1$, the value of r'_{d+1} is decreased until the first child node which can be pruned based on the results from Section 6.3 is found. The rules adopted to fix the variables are outsourced in Algorithm 3. Again, the full branch-and-bound algorithm is presented in Section 6.4.

Algorithm 3 Update r_d

INPUT: $r_d, \alpha(d)$
OUTPUT: r_d

1: **if** $r_d \geq \lfloor \alpha(d) \rfloor$ **then**
2: Set $r_d = r_d + 1$
3: **else**
4: Set $r_d = r_d - 1$
5: **end if**

To conclude this section, we consider the special case $d = k$. This means that at a corresponding node all the integer variables are fixed to certain values given by $r \in \mathbb{Z}^k$. In other words, a leaf node of the branch-and-bound tree is reached. At this point, the sets $\mathcal{L}$ of lower bounds and $\mathcal{U}$ of upper bounds for the enclosure of the nondominated set $\mathcal{Y}_N$ of (MOMIQP) are built up as detailed in the following.

We initialize $\mathcal{L} = \emptyset, \mathcal{U} = U(\emptyset) = \{Z\}$, and $N = \emptyset$. At the leaf node corresponding to $r \in \mathbb{Z}^k$, we solve the problems $(P^r(\ell))$ for all $\ell \in L$. The optimal solutions $x^{*\ell,r} \in \mathbb{R}^{n-k}$ of (MOP^r) lead to feasible points $(r, x^{*\ell,r}) \in \mathbb{Z}^k \times \mathbb{R}^{n-k}$ of (MOMIQP). The upper bound set $\mathcal{U}$ is then updated as

$$\mathcal{U} = U(N \cup \{f^r(x^{*\ell,r}) \mid \ell \in L\}). \tag{6.6}$$

More precisely, we use [58, Algorithm 3] (which is the same as [44, Algorithm 1]) to do so. By [44, Lemma 4.7], for the resulting local upper bound set it holds that

$$\mathcal{N} \subseteq \mathcal{U} - \mathbb{R}^m_+. \tag{6.7}$$

On the other hand, the lower bound set $\mathcal{L}$ is updated by

$$\mathcal{L} = \mathcal{L} \cup \{(\varphi^r(e^1), \ldots, \varphi^r(e^m))\}, \tag{6.8}$$

i.e., by the ideal point of the upper image set $\mathcal{P}^r$. We will show in Lemma 6.4.2 that the resulting set $\mathcal{L}$ computed by our algorithm is indeed a lower bound set in the sense of Definition 6.1.3. We remark that, while $\mathcal{U}$ is always an upper bound set for an enclosure in that sense, for $\mathcal{L}$ this only holds at the end of our algorithm.

6.3 Pruning of Nodes

As already mentioned, given a certain node at level d of the branch-and-bound tree, the interval (I) defines the smallest range of integer values for which corresponding children nodes at level $d + 1$ need to be computed. In this section, we analyze how to stop the computation of new children nodes when fixing variable x_{d+1} to integer values outside this interval. In particular, we provide sufficient conditions that allow to consider a finite number of integer assignments. This implies that our method needs to explore only a finite number of nodes even in the case that the original problem has an unbounded feasible set.

6.3.1 Pruning by Infeasibility

Whenever for some $d \in [k]$ and a vector $r = (r_1, \ldots, r_d) \in \mathbb{Z}^d$ the problem $(P^r(\ell))$ is infeasible, i.e., condition (Inf) holds, the corresponding node and all its children can of course be pruned:

Lemma 6.3.1. *Let the condition (Inf) hold for some $d \in [k]$ and some vector $r \in \mathbb{Z}^d$ of integer fixings. Then there is no feasible point $\bar{x} \in \mathbb{Z}^k \times \mathbb{R}^{n-k}$ of (MOMIQP) such that $(\bar{x}_1, \dots, \bar{x}_d) = (r_1, \dots, r_d)$.*

In Lemma 6.3.2 we prove that, in case (Inf) holds, for integer fixings $r \in \mathbb{Z}^d$ with $r_d > \lceil \beta(d) \rceil$ or $r_d < \lfloor \alpha(d) \rfloor$, thanks to linearity of the constraints, we can also prune the outer siblings of that node. Note that the condition (Inf) cannot occur for integer fixings $r \in \mathbb{Z}^d$ with $r_d \in [\, \alpha(d)\,,\, \beta(d)\,]$.

Lemma 6.3.2. *Let the condition (Inf) hold for some $d \in [k]$ and some vector $r \in \mathbb{Z}^d$ of integer fixings.*

(a) If $r_d = \delta \geq \lceil \beta(d) \rceil$, then there is no feasible point $\bar{x} \in \mathbb{Z}^k \times \mathbb{R}^{n-k}$ of (MOMIQP) such that $(\bar{x}_1, \dots, \bar{x}_d) = (r_1, \dots, r_{d-1}, \bar{r}_d) =: \bar{r} \in \mathbb{Z}^d$ with $\bar{r}_d \geq \delta$.

(b) If $r_d = \delta \leq \lfloor \alpha(d) \rfloor$, then there is no feasible point $\bar{x} \in \mathbb{Z}^k \times \mathbb{R}^{n-k}$ of (MOMIQP) such that $(\bar{x}_1, \dots, \bar{x}_d) = (r_1, \dots, r_{d-1}, \bar{r}_d) =: \bar{r} \in \mathbb{Z}^d$ with $\bar{x}_d \leq \delta$.

Proof. We only prove (a). The proof of (b) is analogous.

Assume by contradiction that there exists a feasible point $\bar{x} \in \mathbb{Z}^k \times \mathbb{R}^{n-k}$ of (MOMIQP) with $(\bar{x}_1, \dots, \bar{x}_d) = \bar{r}$ and $\bar{r}_d \geq \delta$. By the definition of $\beta(d)$ there exists a feasible point of $(P^{r'}(\ell))$ for $r' := (r_1, \dots, r_{d-1}) \in \mathbb{Z}^{d-1}$ and some $\ell \in L$. In particular, there exists some $x' \in \mathbb{R}^n$ with $Ax' \leq b$, $(x'_1, \dots, x'_{d-1}) = (r_1, \dots, r_{d-1})$, and $x'_d \leq \beta(d)$. For any $\lambda \in [0,1]$ let $q(\lambda)$ be the point defined as

$$q(\lambda) := \lambda \bar{x} + (1 - \lambda) x'.$$

By linearity, it holds $Aq(\lambda) \leq b$ for all $\lambda \in [0,1]$. Moreover, $q_i(\lambda) = r_i$ for all $i \in [d-1]$ and for all $\lambda \in [0,1]$. For the d-th component of $q(\lambda)$ we have that $q_d(0) = x'_d \leq \beta(d) \leq \delta \leq \bar{x}_d = q_d(1)$. As a result, there exists $\bar{\lambda} \in [0,1]$ such that $q_d(\bar{\lambda}) = \delta = r_d$. This contradicts (Inf). □

Of course, problem (MOMIQP) can still have an unbounded feasible set and the situation that (Inf) is satisfied might never occur. However, we will show in the forthcoming Section 6.3.2 that even in case (Inf) is never satisfied, we can still prune nodes and their siblings under certain conditions.

6.3.2 Pruning by Lower and Upper Bounds

In this section, we analyze what happens when infeasibility does not occur. In particular, we make the following assumption.

Assumption 6.3.3. *Let $d \in [k]$ and $r = (r_1, \dots, r_d) \in \mathbb{Z}^d$ be a vector of integer fixings. Assume that (Inf) does not hold, i.e. $S^r = \{x \in \mathbb{R}^{n-d} \mid A^d x \leq b^r\} \neq \emptyset$.*

In order to be able to prune certain nodes and their siblings as in Section 6.3.1, we define a pruning condition based on lower and upper bound sets. We say that $LB^r \subseteq \mathbb{R}^m$ is a lower bound set for the node corresponding to the vector $r \in \mathbb{Z}^d$ of integer fixings if

$$\{f(x) \in \mathbb{R}^m \mid x \in \mathbb{Z}^k \times \mathbb{R}^{n-k},\, (x_1, \dots, x_d) = (r_1, \dots, r_d),\, Ax \leq b\} \subseteq LB^r + \mathbb{R}^m_+.$$

A sufficient condition for this to hold is that $\mathcal{P}^r \subseteq LB^r + \mathbb{R}^m_+$. Due to the definition of $\varphi^r(\ell)$ and since $L \subseteq \mathbb{R}^m_+$, we have for any $\ell \in L$ that $\mathcal{P}^r \subseteq \{y \in \mathbb{R}^m \mid \ell^\top y \geq \varphi^r(\ell)\}$. Thus a valid lower bound set for the node is given by

$$LB^r := \left\{ y \in \mathbb{R}^m \;\middle|\; \ell^\top y \geq \varphi^r(\ell)\ \forall \ell \in L \right\}. \tag{6.9}$$

Further, due to $L \subseteq \mathbb{R}^m_+$, we have that $LB^r = LB^r + \mathbb{R}^m_+$. Since we assume that the set L contains the m unit vectors, we obtain for the ideal point $\mathrm{id}^r := (\varphi^r(e^1), \dots, \varphi^r(e^m))$ of the set $\mathcal{P}^r$ that $LB^r \subseteq \{\mathrm{id}^r\} + \mathbb{R}^m_+$.

Intersecting the set of local upper bounds $\mathcal{U}$ with the lower bound set LB^r gives a pruning condition:

$$\forall\, u \in \mathcal{U} :\ u \notin LB^r. \tag{Cond}$$

We need the following lemma for proving that this is indeed a pruning condition.

Lemma 6.3.4. *Let Assumption 6.3.3 hold. If (Cond) holds then for the nondominated set $\mathcal{N}$ of (MOMIQP) we have $\mathcal{N} \cap LB^r = \emptyset$.*

Proof. Since $L \subseteq \mathbb{R}^m_+$, for any $h \in -\mathbb{R}^m_+$ it holds that $\ell^\top h \leq 0$ for all $\ell \in L$. As a result, we have that $u \notin LB^r$ if and only if $(\{u\} - \mathbb{R}^m_+) \cap LB^r = \emptyset$. Hence, (Cond) holds if and only if $(\mathcal{U} - \mathbb{R}^m_+) \cap LB^r = \emptyset$. Together with (6.7) we obtain that if (Cond) holds then also $\mathcal{N} \cap LB^r = \emptyset$. □

Since for any feasible point $\bar{x} \in \mathbb{Z}^d \times \mathbb{R}^{n-d}$ of (MOMIQP) with $(\bar{x}_1, \dots, \bar{x}_d) = (r_1, \dots, r_d)$ it holds that $f(\bar{x}) \in LB^r$, we immediately conclude from Lemma 6.3.4 the following result for pruning:

Lemma 6.3.5. *Let Assumption 6.3.3 hold. Further, let $LB^r \in \mathbb{R}^m$ be a lower bound set as in (6.9) and let (Cond) hold. Then there is no efficient point $\bar{x} \in \mathbb{Z}^k \times \mathbb{R}^{n-k}$ for (MOMIQP) with $(\bar{x}_1, \dots, \bar{x}_d) = (r_1, \dots, r_d)$.*

In the forthcoming Lemma 6.3.6, we prove that as soon as (Cond) holds for a node $r \in \mathbb{Z}^d$ with $r_d \notin [\lfloor \alpha(d) \rfloor, \lceil \beta(d) \rceil]$, we can prune its outer siblings $\bar{r} \in \mathbb{Z}^d$ with $(\bar{r}_1, \dots, \bar{r}_{d-1}) = (r_1, \dots, r_{d-1})$ and $\bar{r}_d > r_d$ or $\bar{r}_d < r_d$.

Lemma 6.3.6. *Let Assumption 6.3.3 hold for some $d \in [k]$ and some vector $r \in \mathbb{Z}^d$ of integer fixings with $r_d \notin [\lfloor \alpha(d) \rfloor, \lceil \beta(d) \rceil]$. Further, let $LB^r \in \mathbb{R}^m$ be the lower bound set computed as in (6.9) and let (Cond) hold.*

(a) If $r_d = \delta > \lceil \beta(d) \rceil$, then there is no efficient point $\bar{x} \in \mathbb{Z}^k \times \mathbb{R}^{n-k}$ of (MOMIQP) such that $(\bar{x}_1, \dots, \bar{x}_d) = (r_1, \dots, r_{d-1}, \bar{r}_d) =: \bar{r} \in \mathbb{Z}^d$ with $\bar{r}_d \geq \delta$.

(b) If $r_d = \delta < \lfloor \alpha(d) \rfloor$, then there is no efficient point $\bar{x} \in \mathbb{Z}^k \times \mathbb{R}^{n-k}$ of (MOMIQP) such that $(\bar{x}_1, \dots, \bar{x}_d) = (r_1, \dots, r_{d-1}, \bar{r}_d) =: \bar{r} \in \mathbb{Z}^d$ with $\bar{r}_d \leq \delta$.

Proof. We only prove (a). The proof of (b) is analogous.

If (Inf) holds for $\bar{r}$ then there cannot be an efficient point $\bar{x}$ of (MOMIQP) with $(\bar{x}_1, \dots, \bar{x}_d) = \bar{r}$, see Lemma 6.3.1. Thus, in the following we consider the case where (Inf) does not hold, i.e., Assumption 6.3.3 holds for $\bar{r} \in \mathbb{Z}^d$. Then we can determine the set $LB^{\bar{r}}$ as in (6.9) based on the values $\varphi^{\bar{r}}(\ell)$ for $\ell \in L$. We will show that it holds

$$\varphi^r(\ell) \leq \varphi^{\bar{r}}(\ell) \quad \text{for all} \quad \ell \in L \tag{6.10}$$

as then we have $LB^{\bar{r}} \subseteq LB^r$ and (Cond) also holds for $LB^{\bar{r}}$. Lemma 6.3.5 then concludes the proof.

To show (6.10), let $\ell \in L$ and denote by $\bar{f}^\ell : \mathbb{R}^{n-d+1} \to \mathbb{R}$ the function

$$\bar{f}^\ell(z) := \ell^\top f(r_1, \dots, r_{d-1}, z_1, \dots, z_{n-d+1}).$$

Within this proof, we use the notation $\bar{b} := b - A(r_1, \dots, r_{d-1}, 0, \dots, 0)$ and denote as usual by A^{d-1} the matrix which is obtained from A by deleting the first $d-1$ columns. Then $A(r_1, \dots, r_{d-1}, z)^\top \leq b$ reduces to $A^{d-1}z \leq \bar{b}$. Using this notation, we have

$$\varphi^r(\ell) = \min_z\{\bar{f}^\ell(z) \mid z_1 = r_d,\ A^{d-1}z \leq \bar{b},\ z \in \mathbb{R}^{n-d+1}\}$$

and

$$\varphi^{\bar{r}}(\ell) = \min_z\{\bar{f}^\ell(z) \mid z_1 = \bar{r}_d,\ A^{d-1}z \leq \bar{b},\ z \in \mathbb{R}^{n-d+1}\}.$$

The first components of the unique minimal solutions $u^{*\ell} \in \operatorname{argmin}_z\{\bar{f}^\ell(z) \mid A^{d-1}z \leq \bar{b},\ z \in \mathbb{R}^{n-d+1}\}$ determine the interval $[\lfloor\alpha(d)\rfloor, \lceil\beta(d)\rceil]$, cf. (6.5), and we have that

$$u_1^{*\ell} \leq \beta(d) \leq \lceil\beta(d)\rceil < \delta = r_d \leq \bar{r}_d.$$

For $\gamma \in \mathbb{R}$ with $u_1^{*\ell} < \gamma \leq \bar{r}_d$ we define the parametric optimization problem $P(\gamma)$ by

$$\begin{array}{ll} \min\limits_z & \bar{f}^\ell(z) \\ \text{s.t.} & z_1 \geq \gamma, \\ & A^{d-1}z \leq \bar{b}, \\ & z \in \mathbb{R}^{n-d+1}. \end{array} \tag{$P(\gamma)$}$$

Since Assumption 6.3.3 holds for $\bar{r} \in \mathbb{Z}^d$, all of these optimization problems are solvable. We denote their optimal value by $v(\gamma)$ for $\gamma \in (u_1^{*\ell}, \bar{r}_d]$. Due to $\bar{r}_d \geq r_d$ we have that $v(r_d) \leq v(\bar{r}_d)$. Next, we prove by contradiction that for all $\gamma \in (u_1^{*\ell}, \bar{r}_d]$ it holds that

$$\begin{aligned} &\min_z\{\bar{f}^\ell(z) \mid z_1 \geq \gamma,\ A^{d-1}z \leq \bar{b},\ z \in \mathbb{R}^{n-d+1}\} \\ =&\min_z\{\bar{f}^\ell(z) \mid z_1 = \gamma,\ A^{d-1}z \leq \bar{b},\ z \in \mathbb{R}^{n-d+1}\}. \end{aligned} \tag{6.11}$$

Let $\gamma \in (u_1^{*\ell}, \bar{r}_d]$ and $\bar{z} \in \mathbb{R}^{n-d+1}$ be the optimal solution of ($P(\gamma)$) with $\bar{z}_1 > \gamma$. We set $q(\lambda) := (1-\lambda)\bar{z} + \lambda u^{*\ell}$, i.e., $q_1(0) = \bar{z}_1 > \gamma$ and $q_1(1) = u_1^{*\ell} < \gamma$. Note that $A^{d-1}q(\lambda) \leq \bar{b}$ holds for all $\lambda \in [0,1]$. Let $0 < \bar{\lambda} < 1$ be such that $q_1(\bar{\lambda}) = \gamma$. Moreover, by the definition of $u^{*\ell}$ we have $\bar{f}^\ell(u^{*\ell}) \leq \bar{f}^\ell(\bar{z})$. Then, from the strict convexity of $\bar{f}^\ell$, we derive

$$\bar{f}^\ell(q(\bar{\lambda})) = \bar{f}^\ell((1-\bar{\lambda})\bar{z} + \bar{\lambda}u^{*\ell}) < (1-\bar{\lambda})\bar{f}^\ell(\bar{z}) + \bar{\lambda}\bar{f}^\ell(u^{*\ell}) \leq \bar{f}^\ell(\bar{z}),$$

which contradicts the minimality of $\bar{z}$ for ($P(\gamma)$). Consequently, (6.11) holds, implies that $\varphi^r(\ell) = v(r_d) \leq v(\bar{r}_d) = \varphi^{\bar{r}}(\ell)$, and we are done with showing (6.10). □

All pruning results within this subsection are based on the condition (Cond). The next result simplifies the evaluation of (Cond). It exploits the fact that for any $u \in \mathcal{U}$ it holds $u \notin LB^r$ if and only if there exists $\ell \in L$ with $\ell^\top u < \varphi^r(\ell)$.

Lemma 6.3.7. *Let Assumption 6.3.3 hold and define for $u \in \mathcal{U}$ the value $\sigma(u)$ by*

$$\sigma(u) := \min\{\ell^\top u - \varphi^r(\ell) \mid \ell \in L\}. \tag{6.12}$$

Then (Cond) holds if and only if $\sigma(u) < 0$ for all $u \in \mathcal{U}$.

The costs for evaluating (Cond) can be further reduced:

Remark 6.3.8. *In order to verify whether* (Cond) *is satisfied it is sufficient to check whether* $\sigma(u) < 0$ *only for those* $u \in \mathcal{U}$ *with* $u \geq \mathrm{id}^r$*, where* $\mathrm{id}^r \in \mathbb{R}^m$ *is the ideal point of* $\mathcal{P}^r$ *or some underestimator of it. This holds because of* $LB^r \subseteq \{\mathrm{id}^r\} + \mathbb{R}^m_+$*. The ideal point is obtained as a byproduct when calculating* LB^r*.*

As we are making use of dual formulations of the problems ($P^r(\ell)$) (see Section 6.5) and as we will try to avoid to solve them exactly, we sometimes calculate just lower bounds for $\varphi^r(\ell)$ in (6.9) and thus derive only sets LB' which are supersets of LB^r. Still those sets can be used to formulate a sufficient condition for (Cond):

Remark 6.3.9. *Let* $LB^r \in \mathbb{R}^m$ *be the lower bound set computed as in* (6.9) *and let* $LB' \supseteq LB^r$ *be an arbitrary superset of it. Then, if* (Cond) *holds for* LB'*, i.e. if for all* $u \in \mathcal{U}$ *we have* $u \notin LB'$*, then* (Cond) *holds also for* LB^r*.*

In Section 6.5, we explain how to make use of Lemma 6.3.7 in combination with Remark 6.3.9 to speed up the pruning strategy in our branch-and-bound algorithm.

6.3.3 Occurring of Pruning Conditions

In the last two subsections, we formulated conditions for pruning a node in case (Inf) or (Cond) hold. Furthermore, we have given conditions in order to prune all the outer siblings of a node in case (Inf) or (Cond) are satisfied at that node. However, since we are not assuming boundedness of the feasible region of (MOMIQP), it may happen that an infinite number of nodes is visited, as neither (Inf) nor (Cond) are satisfied at any node. This would imply that our algorithm never stops. The following lemma shows that this cannot happen and that for all $d \in [k]$ there exist only finitely many integer fixings $r \in \mathbb{Z}^d$ such that neither (Inf) nor (Cond) are satisfied. The strong convexity of the objective functions is key to the proof of the result.

Lemma 6.3.10. *Let* $d \in [k-1] \cup \{0\}$ *and let Assumption 6.3.3 hold at level* d *for* $r \in \mathbb{Z}^d$*, i.e.,* (Inf) *does not hold. Then there exists* $\gamma \in \mathbb{Z}$ *such that for all* $\bar{r} \in \mathbb{Z}^{d+1}$ *with* $(\bar{r}_1, \dots, \bar{r}_d) = r$ *and*

$$\bar{r}_{d+1} \notin [\lfloor \alpha(d+1) \rfloor - \gamma, \lceil \beta(d+1) \rceil + \gamma]$$

either (Cond) *or* (Inf) *is satisfied.*

Proof. First, select an arbitrary element $\ell \in L$. For the current finite and nonempty set of local upper bounds $\mathcal{U}$ define $\xi := \max\{\ell^\top u \mid u \in \mathcal{U}\}$. Since the objective functions $f_j, j \in [m]$, are strongly convex, $\ell^\top f \colon \mathbb{R}^n \to \mathbb{R}$ is a strongly convex quadratic function, too. Thus, there exists some $\nu > 0$ such that

$$\nu\lambda(1-\lambda)\|x - x'\|_2^2 + \ell^\top f(\lambda x + (1-\lambda)x') \leq \lambda \ell^\top f(x) + (1-\lambda)\ell^\top f(x') \tag{6.13}$$

for all $x, x' \in \mathbb{R}^n$ and all $\lambda \in [0,1]$. Let $x^{*\ell,r}$ denote the unique minimizer of the problem ($P^r(\ell)$). By definition it holds $\alpha(d+1) \leq x_1^{*\ell,r}$ and $\beta(d+1) \geq x_1^{*\ell,r}$.

Now, let $\delta \geq 0$ and consider the optimization problem

$$\begin{aligned} \min_x \quad & \ell^\top f^r(x) \\ \text{s.t.} \quad & A^d x \leq b^r \\ & x_1 = \lceil \beta(d+1) \rceil + \delta \\ & x \in \mathbb{R}^{n-d}. \end{aligned} \tag{6.14}$$

First, assume that there exists some $\bar{\delta} \geq 0$ such that (6.14) is infeasible. Then we define $\bar{\gamma} := \lceil \bar{\delta} \rceil \in \mathcal{Y}_N$ and it holds that (6.14) remains infeasible for all $\delta \geq \bar{\gamma} \geq \bar{\delta}$. To see this, assume that there exists some $\delta' > \bar{\delta}$ such that (6.14) is feasible. The corresponding minimizer $x' \in \mathbb{R}^{n-d}$ is not only feasible for (6.14), but also for ($P^r(\ell)$). Further, it holds that

$$x_1^{*\ell,r} \leq \lceil \beta(d+1) \rceil + \bar{\delta} < \lceil \beta(d+1) \rceil + \delta' = x'_1.$$

However, since both $x^{*\ell,r}$ and x' are feasible for ($P^r(\ell)$) and the constraints are all linear, there exists some feasible point $\hat{x} \in \mathbb{R}^{n-d}$ for ($P^r(\ell)$) with $\hat{x}_1 = \lceil \beta(d+1) \rceil + \bar{\delta}$ which contradicts the assumption that (6.14) is infeasible for $\bar{\delta}$.

Next, assume that (6.14) is feasible for all $\delta \geq 0$ and denote for each $\delta \geq 0$ by $\bar{x}(\delta) \in \mathbb{R}^{n-d}$ its unique minimizer. Note that for $\bar{r} \in \mathbb{Z}^{d+1}$ with $(\bar{r}_1, \dots, \bar{r}_d) = (r_1, \dots, r_d)$ and $\bar{r}_{d+1} = \lceil \beta(d+1) \rceil + \delta$ it holds $\varphi^{\bar{r}}(\ell) = \ell^\top f^r(\bar{x}(\delta))$.

We obtain from (6.13) with $\lambda = 1/2$ and for any $\delta \geq 0$ that

$$0.25\,\nu \left\| x^{*\ell,r} - \bar{x}(\delta) \right\|_2^2 + \ell^\top f^r(0.5\,x^{*\ell,r} + 0.5\,\bar{x}(\delta)) \leq 0.5\,\ell^\top f^r(x^{*\ell,r}) + 0.5\,\ell^\top f^r(\bar{x}(\delta)).$$

Since $\bar{x}(\delta)$ and $x^{*\ell,r}$ are feasible for ($P^r(\ell)$), so are all convex combinations of them and in particular the point $0.5\,x^{*\ell,r} + 0.5\,\bar{x}(\delta)$. As $x^{*\ell,r}$ is the unique minimizer of ($P^r(\ell)$) we have $\ell^\top f^r(x^{*\ell,r}) \leq \ell^\top f^r(0.5\,x^{*\ell,r} + 0.5\,\bar{x}(\delta))$. Further, we have $\ell^\top f^r(x^{*\ell,r}) \leq \ell^\top f^r(\bar{x}(\delta))$ and hence

$$0.25\,\nu \left\| x^{*\ell,r} - \bar{x}(\delta) \right\|_2^2 + \ell^\top f^r(x^{*\ell,r}) \leq \ell^\top f^r(\bar{x}(\delta)).$$

Finally, making use of $\bar{x}(\delta)_1 = \lceil \beta(d+1) \rceil + \delta \geq x_1^{*\ell,r} + \delta$, we obtain that

$$0.25\nu\delta^2 + \ell^\top f^r(x^{*\ell,r}) \leq \ell^\top f^r(\bar{x}(\delta)).$$

For $\delta \geq 0$ larger than some $\bar{\gamma} \in \mathcal{Y}_N$, we have that the left hand side of this inequality exceeds ξ such that $\xi < \ell^\top f^r(\bar{x}(\delta))$ for all $\delta \geq \bar{\gamma}$.

Analogously, replacing the constraint $x_1 = \lceil \beta(d+1) \rceil + \delta$ in (6.14) by $x_1 = \lfloor \alpha(d+1) \rfloor - \delta$ and using that $\alpha(d+1) \leq x_1^{*\ell,r}$, one obtains that there exists some $\underline{\gamma} \in \mathcal{Y}_N$ such that either (6.14) becomes infeasible or $\xi < \ell^\top f^r(\bar{x}(\delta))$ for all $\delta \geq \underline{\gamma}$.

Thus, for $\gamma := \max\{\bar{\gamma}, \underline{\gamma}\}$ and an arbitrary vector $\bar{r} \in \mathbb{Z}^{d+1}$ of integer fixings with $(\bar{r}_1, \dots, \bar{r}_d) = (r_1, \dots, r_d)$ and $\bar{r}_{d+1} \notin [\lfloor \alpha(d+1) \rfloor - \gamma, \lceil \beta(d+1) \rceil + \gamma]$ we obtain that either (Cond) holds since $\varphi^{\bar{r}}(\ell) > \xi \geq \ell^\top u$ for all $u \in \mathcal{U}$ or that (Inf) holds. □

6.4 DEIA-BB: algorithmic scheme and finiteness

In order to summarize what we have presented so far, we report in Algorithm 4 the scheme of our branch-and-bound method, called DEIA-BB (for *Detector of Efficient Integer Assignments-Branch-and-Bound*). As already mentioned, DEIA-BB computes two things. Primarily, it computes a lower bound set $\mathcal{L}$ and an upper bound set $\mathcal{U}$ for an enclosure of the nondominated set of (MOMIQP). Thereby, it also computes a superset $\mathcal{S}$ of the set of efficient integer assignments. In the following, we will briefly describe step-by-step how the algorithm works and how the steps are related to the theoretical results presented in the previous sections.

The algorithm first checks in line 3 whether the continuous relaxation of (MOMIQP) is feasible, i.e., whether the corresponding feasible set $\{x \in \mathbb{R}^n \mid Ax \leq b\}$ is nonempty. If it is nonempty, then $\alpha(1)$ and $\beta(1)$ are computed and the algorithm continues with the main loop in line 9. Otherwise, the algorithm detects the infeasibility of (MOMIQP) and stops.

Algorithm 4 DEIA-BB: Detector of Efficient Integer Assignments

INPUT: m strongly convex quadratic functions $f_j : \mathbb{R}^n \to \mathbb{R}$, $j = 1, \dots, m$, linear constraints $Ax \leq b$, finite set $L \subseteq \mathbb{R}^m_+$ with $e^j \in L$ for all $j \in [m]$

OUTPUT: $\mathcal{L}$, $\mathcal{U}$, $\mathcal{S}$

Perform a preprocessing phase to speed up computations (see Algorithm 5)
Set $U = U(\emptyset) = \{Z\}$, $\mathcal{S} = \emptyset$ and $d = 0$
if $\{x \in \mathbb{R}^n \mid Ax \leq b\} \neq \emptyset$ **then**
    Compute $\alpha(1)$ and $\beta(1)$ according to (6.5)
    Set $d = 1$, $r_1 = \lfloor \alpha(1) \rfloor$
else
    Stop the algorithm and state infeasibility of (MOMIQP)
end if
while $d > 0$ **do**
    Evaluate whether (Inf) or (Cond) holds for d and $r = (r_1, \dots, r_d)$
    if not ((Cond) or (Inf)) **then**
        if $d = k$ **then**
            Update $\mathcal{U}$ and $\mathcal{L}$ according to (6.6), (6.8)
            Update $\mathcal{S} = \mathcal{S} \cup \{r\}$
        else
            Compute $\alpha(d+1)$ and $\beta(d+1)$ according to (6.5)
        end if
    end if
    if ((Cond) or (Inf)) and $r_d < \lfloor \alpha(d) \rfloor$ **then**
        Set $d = d - 1$;
        if $(d > 0)$ Update r_d with Algorithm 3 **end if**
    else if ((Cond) or (Inf)) and $r_d \geq \lfloor \alpha(d) \rfloor$ **then**
        if $r_d \geq \lceil \beta(d) \rceil$ **then**
            Set $r_d = \lfloor \alpha(d) \rfloor - 1$
        else
            Set $r_d = r_d + 1$
        end if
    else if $d \leq k - 1$ **then**
        $d = d + 1$;
        Set $r_d = \lfloor \alpha(d) \rfloor$
    else
        Update r_d with Algorithm 3
    end if
end while

The while loop basically computes feasible leaf nodes, i.e., integer fixings $\bar{r} \in \mathbb{Z}^k$ such that $\{x \in \mathbb{R}^{n-k} \mid A^k x \leq b^{\bar{r}}\} \neq \emptyset$, with a depth-first approach. It starts at depth $d = 1$ with the integer fixing $r = r_1 = \lfloor \alpha(1) \rfloor \in \mathbb{Z}^d$. We remark that the depth $d \in \mathcal{Y}_N$ never exceeds k since it is only increased in line 29 of the algorithm and this line is only called if $d \leq k-1$. We also remark that the first node that is considered at level $d+1$ is always the one with the vector $r = (r_1, \dots, r_d, \lfloor \alpha(d+1) \rfloor)$ of integer fixings.

For an arbitrary node, i.e., an arbitrary vector $r \in \mathbb{Z}^d$ of integer fixings, at depth $d \in [k]$ the algorithm checks (line 11) whether this node needs to be explored. Namely, it checks whether (MOMIQP) with the first d variables fixed to $r \in \mathbb{Z}^d$ is feasible and (Cond) does not hold. If this is the case and $d = k$ the algorithm has reached a leaf node and thus a feasible integer fixing for (MOMIQP). This allows us to update the lower and upper bound sets $\mathcal{L}$ and $\mathcal{U}$ for the initial enclosure, see line 13. Otherwise, we have not reached a leaf node and compute the bounds $\alpha(d+1)$ and $\beta(d+1)$ for the integer fixings at level $d+1$.

Next, the algorithm checks whether the children or siblings of the current node can be pruned based on the results of Sections 6.3.1 and 6.3.2. Note that for the former we need (Inf) and for the latter we need (Cond) to hold in order to prune. If neither (Inf) nor (Cond) hold, then the conditions in lines 19 and 22 of Algorithm 4 are not satisfied, nothing is pruned, and the algorithm moves on (with its depth-first approach) to level $d+1$, see line 29. In case $d = k$, i.e., at a leaf node, the algorithm stays at level $d = k$ and explores the siblings of the current node, see line 32. We remark that by Lemma 6.3.10 at each level $d \in [k]$ there only exist finitely many nodes where neither (Inf) nor (Cond) are satisfied.

Thus, for Algorithm 4, it remains the setting that (Inf) or (Cond) are satisfied, i.e., that either the clause in line 19 or in line 22 is true. If (Inf) holds then we can prune all the children nodes of the current node corresponding to $r \in \mathbb{Z}^d$ and in particular the node itself. Also if (Cond) holds we can prune all the children nodes by Lemma 6.3.5. Thus the algorithm only needs to decide whether siblings of the current node need to be explored or can be pruned. Since at each level d we always start with $r_d = \lfloor \alpha(d) \rfloor$ and the value of r_d is only changed if one of the clauses in line 19 or 22 is true, we first consider the case in line 22. If $\lfloor \alpha(d) \rfloor \leq r_d \leq \lceil \beta(d) \rceil$ we cannot prune any siblings of the current node based on the results from the previous sections. Thus, we need to explore them. This is done by setting $r_d = r_d + 1$ in line 26 of Algorithm 4. If $r_d > \lceil \beta(d) \rceil$ it is known from Lemmas 6.3.2 and 6.3.6 that all siblings with $\tilde{r}_d > r_d$ can be pruned. Thus, the algorithm will not continue to explore such nodes and goes on exploring nodes to the left of $\lfloor \alpha(d) \rfloor$ by setting $r_d = \lfloor \alpha(d) \rfloor - 1$, see line 24. For any siblings of the current node this means that the condition in line 22 will never be satisfied again. Instead, the condition in line 19 will be satisfied for any future sibling where (Inf) or (Cond) holds. If this is the case, then also all siblings with $\tilde{r}_d < r_d$ can be pruned by Lemma 6.3.2 or 6.3.6. As a result, since the node corresponding to the current integer assignment $r \in \mathbb{Z}^d$ itself and all of its sibling nodes can be pruned, also its parent node at level $d-1$ can be pruned. Thus, Algorithm 4 moves back to level $d-1$, see line 20, and continues by exploring a sibling of that parent node.

This whole procedure is repeated until we reach line 20 with $d = d-1 = 0$, return to the root node, and terminate the algorithm since the condition $d > 0$ of the while loop is no longer satisfied. Together with Lemma 6.3.10 and the fact that each of the intervals $[\lfloor \alpha(d+1) \rfloor, \lceil \beta(d+1) \rceil]$ is bounded, we eventually reach this line within a finite number of iterations and obtain the following finiteness result for Algorithm 4.

Theorem 6.4.1. *Algorithm 4 stops after a finite number of iterations returning the sets $\mathcal{L}$, $\mathcal{U}$ and $\mathcal{S}$.*

From the sets $\mathcal{L}$ and $\mathcal{U}$ we can build an enclosure of the nondominated set $\mathcal{Y}_N$ of (MOMIQP). Indeed, the following lemma shows that the output set $\mathcal{L} \subseteq \mathbb{R}^m$ of `DEIA-BB` is a lower bound

set of (MOMIQP). This in turn implies (see Corollary 6.4.3) that `DEIA-BB` is able to release an enclosure (or box approximation) in a finite number of iterations.

Lemma 6.4.2. *The set $\mathcal{L} \subseteq \mathbb{R}^m$ computed by Algorithm 4 is a lower bound set in the sense of Definition 6.1.3.*

Proof. By Theorem 6.4.1 we know that $\mathcal{L}$ is finite. It only remains to prove that $\mathcal{Y}_N \subseteq \mathcal{L} + \mathbb{R}^m_+$. So let $z \in \mathcal{Y}_N$ and let $\bar{x} \in S$, $r \in \mathbb{Z}^k$, with $(\bar{x}_1, \ldots, \bar{x}_k) = (r_1, \ldots, r_k)$, such that $f(\bar{x}) = z$. We prove that `DEIA-BB` has explored the leaf node $r \in \mathbb{Z}^k$. This would imply by (6.8) that for the ideal point $\mathrm{id}^r = (\varphi^r(e^1), \ldots, \varphi^r(e^m)) \in \mathcal{L}$ computed at this leaf node it holds that $\mathrm{id}^r \leq z$. By contradiction, assume that the leaf node $r \in \mathbb{Z}^k$ has not been explored. Then the parent node $r' \in \mathbb{Z}^d$ of this leaf node at a certain level $d \in [k]$ with $(r'_1, \ldots, r'_d) = (r_1, \ldots, r_d)$ was pruned. Since $\bar{x} \in \mathbb{R}^n$ is feasible for (MOMIQP) this parent node was not pruned by (Inf). Hence, it was pruned by (Cond). This means that for all $u \in \mathcal{U}$ we have that $u \notin LB^{r'}$. By (6.7) we also have that $z = f(\bar{x}) \in \mathcal{Y}_N \subseteq \mathcal{U} - \mathbb{R}^m_+$. But then there exists $u \in \mathcal{U}$ such that $u \geq z \in LB^r + \mathbb{R}^m_+ = LB^r \subseteq LB^{r'}$ which is a contradiction. □

Corollary 6.4.3. *Let $\mathcal{L}, \mathcal{U} \subseteq \mathbb{R}^m$ be the finite sets obtained by Algorithm 4, $\sigma > 0$ a small offset, and denote by $e \in \mathbb{R}^m$ the all-ones vector. Then $B := \mathcal{A}(L', U') = (L' + \mathbb{R}^m_+) \cap (U' - \mathbb{R}^m_+)$ with $L' := \mathcal{L} - \{\sigma e\}$ and $U' := \mathcal{U} + \{\sigma e\}$ is an enclosure of the nondominated set $\mathcal{Y}_N$ of (MOMIQP) such that $\mathcal{N} \subseteq \mathrm{int}(B)$.*

6.5 Using Dual Relaxations

Let $r = (r_1, \ldots, r_d)^\top \in \mathbb{Z}^d$ be the vector of integer fixings defining a node at level d in our branch-and-bound algorithm. As already stated, the multiobjective continuous relaxation of problem (MOMIQP), where the integer variables are fixed to $r \in \mathbb{R}^d$, is (MOP^r). Instead of addressing the problem (MOP^r), our aim is to compute LB^r by addressing the dual of the $|L|$ single-objective problems ($P^r(\ell)$), where the objective function is defined by

$$\begin{aligned} \ell^\top f^r(x) &= \sum_{j=1}^m \ell_j (x^\top Q_j^d x + (c^{j,r})^\top x + a_{j,r}) \\ &= x^\top \left(\sum_{j=1}^m \ell_j Q_j^d \right) x + \left(\sum_{j=1}^m \ell_j c^{j,r} \right)^\top x + \sum_{j=1}^m \ell_j a_{j,r}. \end{aligned}$$

We do so in order to accelerate the pruning process for those nodes that can be pruned because of (Cond). As we will see, addressing the dual of the $|L|$ single-objective problems ($P^r(\ell)$) may allow to stop the computation of LB^r to a rough though effective-for-pruning set and this clearly helps in saving computational effort.

For ease of notation, we introduce the following:

$$\bar{Q}_\ell^d = \sum_{j=1}^m \ell_j Q_j^d, \qquad \bar{c}^{\ell,r} = \sum_{j=1}^m \ell_j c^{j,r}, \qquad \bar{a}_{\ell,r} = \sum_{j=1}^m \ell_j a_{j,r}.$$

We obtain the dual of problem ($P^r(\ell)$) by first forming the Lagrangian $\mathcal{L}_\ell^d \colon \mathbb{R}^{n-d} \times \mathbb{R}^p \to \mathbb{R}$,

$$\mathcal{L}_\ell^d(x, \lambda) = x^\top \bar{Q}_\ell^d x + (\bar{c}^{\ell,r})^\top x + \bar{a}_{\ell,r} + \lambda^\top (A^d x - b^r)$$

which depends on x and the Lagrange multipliers $\lambda \in \mathbb{R}^p$. Then, for fixed $\lambda \in \mathbb{R}^p$, one has to minimize $\mathcal{L}_\ell^d$ with respect to the primal variables x. As $\bar{Q}_\ell^d$ is under our assumptions positive

definite, $\mathcal{L}_\ell^d(\cdot,\lambda)$ is a strictly convex quadratic function and its unique minimizer can be computed in closed form as

$$x_\ell^d(\lambda) = -\frac{1}{2}(\bar{Q}_\ell^d)^{-1}(\bar{c}^{\ell,r} + A^{d\top}\lambda).$$

Then, the dual of problem ($P^r(\ell)$) is

$$\begin{aligned} \max_{\lambda} \quad & \mathcal{L}_\ell^d(\lambda) \\ & \lambda \in \mathbb{R}_+^p, \end{aligned} \tag{6.15}$$

with

$$\mathcal{L}_\ell^d(\lambda) := \mathcal{L}_\ell^d(x_\ell^d(\lambda),\lambda) = \lambda^\top \left(-\frac{1}{4}A^d(\bar{Q}_\ell^d)^{-1}A^{d\top}\right)\lambda - \left(b^{r\top} + \frac{1}{2}(\bar{c}^{\ell,r})^\top(\bar{Q}_\ell^d)^{-1}A^{d\top}\right)\lambda$$
$$- \frac{1}{4}(\bar{c}^{\ell,r})^\top(\bar{Q}_\ell^d)^{-1}(\bar{c}^{\ell,r}) + \bar{a}_{\ell,r}.$$

Note that $-\frac{1}{4}A^d(\bar{Q}_\ell^d)^{-1}A^{d\top} =: -\widetilde{Q}_\ell^d$ is a negative semidefinite matrix, so that problem (6.15) can be seen as a convex quadratic minimization problem with simple nonnegativity constraints. Also note that since all constraints of problem ($P^r(\ell)$) are affine, strong duality holds if the primal problem ($P^r(\ell)$) is feasible [67]. On the other hand, if problem ($P^r(\ell)$) is infeasible, the dual problem (6.15) is unbounded [67].

Thanks to weak duality, we also have that $\mathcal{L}_\ell^d(\lambda) \leq \varphi^r(\ell)$ for each feasible $\lambda \geq 0$. This means that Lemma 6.3.7 can be easily extended, cf. Remark 6.3.9, as follows:

Lemma 6.5.1. *Let Assumption 6.3.3 hold and define for $u \in \mathcal{U}$, $\lambda \in \mathbb{R}_+^p$ the value $\sigma_{\mathcal{L}}(u,\lambda)$ by*

$$\sigma_{\mathcal{L}}(u,\lambda) := \min\{\ell^\top u - \mathcal{L}_\ell^d(\lambda) \mid \ell \in L\}.$$

Then (Cond) holds if and only if for all $u \in \mathcal{U}$ there exists some $\lambda := \lambda(u) \in \mathbb{R}_+^p$ such that $\sigma_{\mathcal{L}}(u,\lambda) < 0$.

Proof. First we will show that if for all $u \in \mathcal{U}$ there exists $\lambda(u) \in \mathbb{R}_+^p$ such that $\sigma_{\mathcal{L}}(u,\lambda(u)) < 0$ then this implies (Cond). Assume by contradiction that (Cond) does not hold. Then there exists $\bar{u} \in \mathcal{U}$ such that $\bar{u} \in LB^r$. This implies $\ell^\top\bar{u} - \varphi^r(\ell) \geq 0$ for all $\ell \in L$. Due to weak duality it holds for all $\lambda \in \mathbb{R}_+^p$ that $\mathcal{L}_\ell^d(\lambda) \leq \varphi^r(\ell)$. Hence, we also have that $\ell^\top\bar{u} - \mathcal{L}_\ell^d(\lambda) \geq 0$ for all $\ell \in L$ and all $\lambda \in \mathbb{R}_+^p$ which contradicts $\sigma_{\mathcal{L}}(\bar{u},\lambda(\bar{u})) < 0$.

We now show that if (Cond) holds, then for all $u \in \mathcal{U}$ there exists $\lambda \in \mathbb{R}_+^p$ such that $\sigma_{\mathcal{L}}(u,\lambda) < 0$. For that fix some $u \in \mathcal{U}$. By Lemma 6.3.7, there exists some $\ell \in L$ such that $\ell^\top u < \varphi^r(\ell)$. Since strong duality holds, $\hat{\lambda}^\ell \in \mathbb{R}_+^p$ exists such that $\mathcal{L}_\ell^d(\hat{\lambda}^\ell) = \varphi^r(\ell)$. This implies that also $\sigma_{\mathcal{L}}(u,\hat{\lambda}^\ell) < 0$. □

Remark 6.5.2. *If Assumption 6.3.3 does not hold, namely ($P^r(\ell)$) is infeasible, for each $\ell \in L$ a sequence of points $\{\lambda^{\ell,k}\} \subseteq \mathbb{R}_+^p$ exists such that $\lim_{k\to\infty}\mathcal{L}_\ell^d(\lambda^{\ell,k}) = +\infty$. In particular, for each $\ell \in L$ there is some sufficiently large $\bar{k}(\ell) \in \mathcal{Y}_N$ such that*

$$\left(\max_{u\in\mathcal{U}} \ell^\top u\right) - \mathcal{L}_\ell^d(\lambda^{\ell,\bar{k}(\ell)}) < 0.$$

Thus, for all $u \in \mathcal{U}$ and all $\ell \in L$ there is some $\lambda := \lambda^{\ell,\bar{k}(\ell)}$ such that $\ell^\top u - \mathcal{L}_\ell^d(\lambda) < 0$ which implies that for each $u \in \mathcal{U}$ there is some $\lambda \in \mathbb{R}_+^p$ with $\sigma_{\mathcal{L}}(u,\lambda) < 0$. In fact, there even exists one $\lambda' \in \mathbb{R}_+^p$ for all $u \in \mathcal{U}$ such that $\sigma_{\mathcal{L}}(u,\lambda') < 0$.

We address problem (6.15) with `FAST-QPA`, an active set feasible method devised in [13] that uses conjugate gradient directions. The reduced matrices $\bar{Q}_\ell^d$, $(\bar{Q}_\ell^d)^{-1}$, and A^d only depend on the depth d, but not on specific integer fixings $r \in \mathbb{Z}^d$. Hence, the quadratic part of the reduced dual objective functions $\widetilde{Q}_\ell^d$ can be computed in the preprocessing phase, as it only depends on $(\bar{Q}_\ell^d)^{-1}$ and A^d. What is more, also the maximum eigenvalue $\lambda_{\max}(\widetilde{Q}_\ell^d)$, needed for ensuring a proper setting of the parameter for the active set estimate and the convergence of `FAST-QPA` (see [13]), can be computed in the preprocessing phase. The preprocessing phase used in our implementation is detailed in Algorithm 5.

Algorithm 5 Preprocessing

INPUT: m strongly convex quadratic functions $f_j : \mathbb{R}^n \to \mathbb{R}$, $j = 1, \dots, m$, linear constraints $Ax \leq b$, finite set of vectors L, number of integer variables k
OUTPUT: $(\bar{Q}_\ell^d)$, $(\bar{Q}_\ell^d)^{-1}$, A^d, $\widetilde{Q}_\ell^d$, $\lambda_{max}(\widetilde{Q}_\ell^d)$ for $d = 0, \dots, n-1$, for $\ell \in L$;

1: For $d = 0, \dots, n-1$ let A^d be the submatrix of A given by columns $d+1, \dots, n$;
2: For $d = 0, \dots, n-1$ and $\ell \in L$ compute the submatrix $\bar{Q}_\ell^d$;
3: For $d = 0, \dots, n-1$ and $\ell \in L$ compute $(\bar{Q}_\ell^d)^{-1}$;
4: For $d = 0, \dots, n-1$ and $\ell \in L$ compute $\widetilde{Q}_\ell^d = A^d (\bar{Q}_\ell^d)^{-1} {A^d}^\top$
5: For $d = 0, \dots, n-1$ and $\ell \in L$ compute $\lambda_{\max}(\widetilde{Q}_\ell^d)$, the maximum eigenvalue of $\widetilde{Q}_\ell^d$,

Let $\{\lambda^k\}$ be the sequence of points produced by `FAST-QPA` when dealing with problem (6.15). Given the properties of `FAST-QPA`, λ^k is feasible for all $k \in \mathcal{Y}_N$ and $\{\mathcal{L}_\ell^d(\lambda^k)\}$ is a monotonically increasing sequence. From the convergence results shown in [13, Proposition 11], in case problem (6.15) admits a maximal solution, under specific assumptions on the parameter used in the active set estimate, we have that

$$\lim_{k \to +\infty} \| \max\{0, \nabla \mathcal{L}_\ell^d(\lambda^k)\} \| = 0.$$

By [13, Theorem 13] this implies that every limit point of the sequence $\{\lambda^k\}$ produced by `FAST-QPA` satisfies the standard first-order optimality conditions for problem (6.15). Furthermore, since problem (6.15) is a convex problem (maximization of a concave function over a convex feasible set), this in turn implies that every limit point of $\{\lambda^k\}$ is an optimal point. In our implementation of `FAST-QPA`, we declare optimality when the point λ^k satisfies the condition

$$\| \max\{0, \nabla \mathcal{L}_\ell^d(\lambda^k)\} \| \leq 10^{-5}, \tag{6.16}$$

having then a guarantee that the algorithm stops after a finite number of iterations.

Handling problem (6.15) with a feasible method (i.e., an optimization method able to produce a sequence of feasible points) allows us to implement a strategy for which the node corresponding to $r \in \mathbb{Z}^d$ can be pruned before computing the lower bound set LB^r. We call this phenomenon *early pruning*. We now describe the implemented strategy and give an example of early pruning.

Given $u \in \mathcal{U}$, thanks to Lemma 6.5.1 and Remark 6.5.2, when dealing with problem (6.15) for a specific $\ell \in L$, we stop `FAST-QPA` as soon as one of the following occurs:

i) we get, at iteration $\hat{k}(\ell)$, that $\ell^\top u < \mathcal{L}_\ell^d(\lambda^{\ell,\hat{k}(\ell)})$ implying

$$\sigma_{\mathcal{L}}(u, \lambda^{\ell,\hat{k}(\ell)}) < 0, \tag{6.17}$$

ii) we have that (6.16) holds at iteration $k(\ell)$ and we set $\varphi^r(\ell) := \mathcal{L}_\ell^d(\lambda^{\ell,k(\ell)})$.

Note that, in case i), $\hat{k}(\ell) \leq k(\ell)$ and $\mathcal{L}_\ell^d(\lambda^{\ell,\hat{k}(\ell)}) \leq \mathcal{L}_\ell^d(\lambda^{\ell,k(\ell)})$. If Assumption 6.3.3 holds for the current vector $r \in \mathbb{Z}^d$ of integer fixings at level d, i.e., $S^r \neq \emptyset$, and (6.17) holds for every $u \in \mathcal{U}$, condition (Cond) holds and the node can be pruned. Moreover, in case (Inf) holds, i.e., $S^r = \emptyset$, the $|L|$ dual problems (6.15) are unbounded. Then, i) occurs and (6.17) is satisfied for every $u \in \mathcal{U}$ so that the node is pruned after a finite number of iterations of `FAST-QPA` in that case as well. Consequently, by applying `FAST-QPA` in our implementation of `DEIA-BB`, we can possibly prune the node using only $\hat{k}(\ell)$ iterations of `FAST-QPA`, see i), instead of $k(\ell)$ iterations to compute $\varphi^r(\ell)$ exactly, see ii).

We now discuss a biobjective example where *early pruning* is possible, see also Figure 6.1. Consider an integer fixing $r = (r_1, \dots, r_d) \in \mathbb{Z}^d$, a set of local upper bounds $\mathcal{U} = \{u^1, u^2, u^3\}$, and a set of vectors $L = \{(1,0),\ (0,1),\ (0.5, 0.5)\}$. In Figure 6.1a the optimal lower bound set LB^r is represented by the blue dashed lines. We will show by this example that our pruning strategy allows us to prune the node without the need of computing LB^r exactly, but just a rough approximation of it. At the beginning, `DEIA-BB` takes into account the first local upper bound $u^1 \in \mathcal{U}$ and calls `FAST-QPA` on problem (6.15) with $\ell^1 = (1,0)^\top$. In Figure 6.1b we see how `FAST-QPA` stops when i) is satisfied. In other words, for u^1, `FAST-QPA` stops at iteration $\hat{k}(\ell^1)$ and detects a $\lambda^{\ell^1,\hat{k}(\ell^1)}$ such that $\sigma_{\mathcal{L}}(u^1, \lambda^{\ell^1,\hat{k}(\ell^1)}) < 0$. Since $\sigma_{\mathcal{L}}(u^1, \lambda^{\ell^1,\hat{k}(\ell^1)}) < 0$, `DEIA-BB` can move to the next local upper bound $u^2 \in \mathcal{U}$. Since $\sigma_{\mathcal{L}}(u^2, \lambda^{\ell^1,\hat{k}(\ell^1)}) > 0$, `FAST-QPA` is resumed on problem (6.15) with $\ell^1 = (1,0)^\top$, from iteration $\hat{k}(\ell^1)$. From Figure 6.1c we see again how `FAST-QPA` stops before reaching optimality or, in other words, it stops at an iteration $\bar{k}(\ell^1) > \hat{k}(\ell^1)$ such that $\sigma_{\mathcal{L}}(u^2, \lambda^{\ell^1,\bar{k}(\ell^1)}) < 0$. As before, since $\sigma_{\mathcal{L}}(u^2, \lambda^{\ell^1,\bar{k}(\ell^1)}) < 0$, `DEIA-BB` can move to the next local upper bound $u^3 \in \mathcal{U}$.

Since $\sigma_{\mathcal{L}}(u^3, \lambda^{\ell^1,\bar{k}(\ell^1)}) > 0$, `FAST-QPA` is resumed on problem (6.15) with $\ell^1 = (1,0)^\top$, from iteration $\bar{k}(\ell^1)$. From Figure 6.1d we notice (see the blue dashed line) that in this case `FAST-QPA` stops reaching optimality, or in other words at an iteration $k(\ell^1)$ such that (6.16) holds. However, we still have that $\sigma_{\mathcal{L}}(u^3, \lambda^{\ell_1,k(\ell^1)}) > 0$. Then, the new vector $\ell^2 \in L$ is considered and `FAST-QPA` is called on problem (6.15) with $\ell^2 = (0,1)^\top$. From Figure 6.1d we see that `FAST-QPA` stops at iteration $\hat{k}(\ell^2)$ detecting $\lambda^{\ell^2,\hat{k}(\ell^2)}$ such that $\sigma_{\mathcal{L}}(u^3, \lambda^{\ell^2,\hat{k}(\ell^2)}) < 0$. Then, by Lemma 6.5.1, we have that (Cond) holds and the node can be pruned. Note that `DEIA-BB` did not have to compute LB^r. In particular, we did not need to solve the (duals of) $|L|$ single-objective problems ($P^r(\ell)$) to optimality.

6.6 Numerical results

In order to investigate the performance of `DEIA-BB`, we considered randomly generated instances of (MOMIQP) with $m \in \{2, 3, 4\}$. The instances were built with a number of variables $n \in \{5, 10, 15, 20\}$, a number of constraints $p = 15$, and a percentage of integer variables out of the total number of variables equal to $\%\text{int} \in \{25, 50, 75, 100\}$ (rounded up). Matrices $Q_j \in \mathbb{R}^{n\times n}, j \in [m]$ were built using the MATLAB function `sprandsym` and we considered three different density levels $\rho \in \{0.25, 0.50, 0.75\}$. Namely, we generated matrices with approximately $\rho \cdot n^2$ nonzeros entries. For what concerns the linear inequality constraints, we randomly generated matrix $A \in \mathbb{R}^{p\times n}$ and vectors $b \in \mathbb{R}^p$ with $m = 15$ using the MATLAB functions `sprandsym` and `rand` respectively. For each combination of $n, m, \%int, \rho$ we produced 5 different instances, having a total of 720 instances. All the algorithms considered have been implemented in MATLAB. In our implementation of `DEIA-BB` we considered $\{1, \dots, k\}$ as order for fixing. Note that different orderings, as well as alternative branching strategies, could be explored. However, in these numerical tests we focus in analyzing the performance with approximations of the upper image set $\mathcal{P}^r$, obtained from different values for $|L|$ and the related benefits in using dual subproblems. All

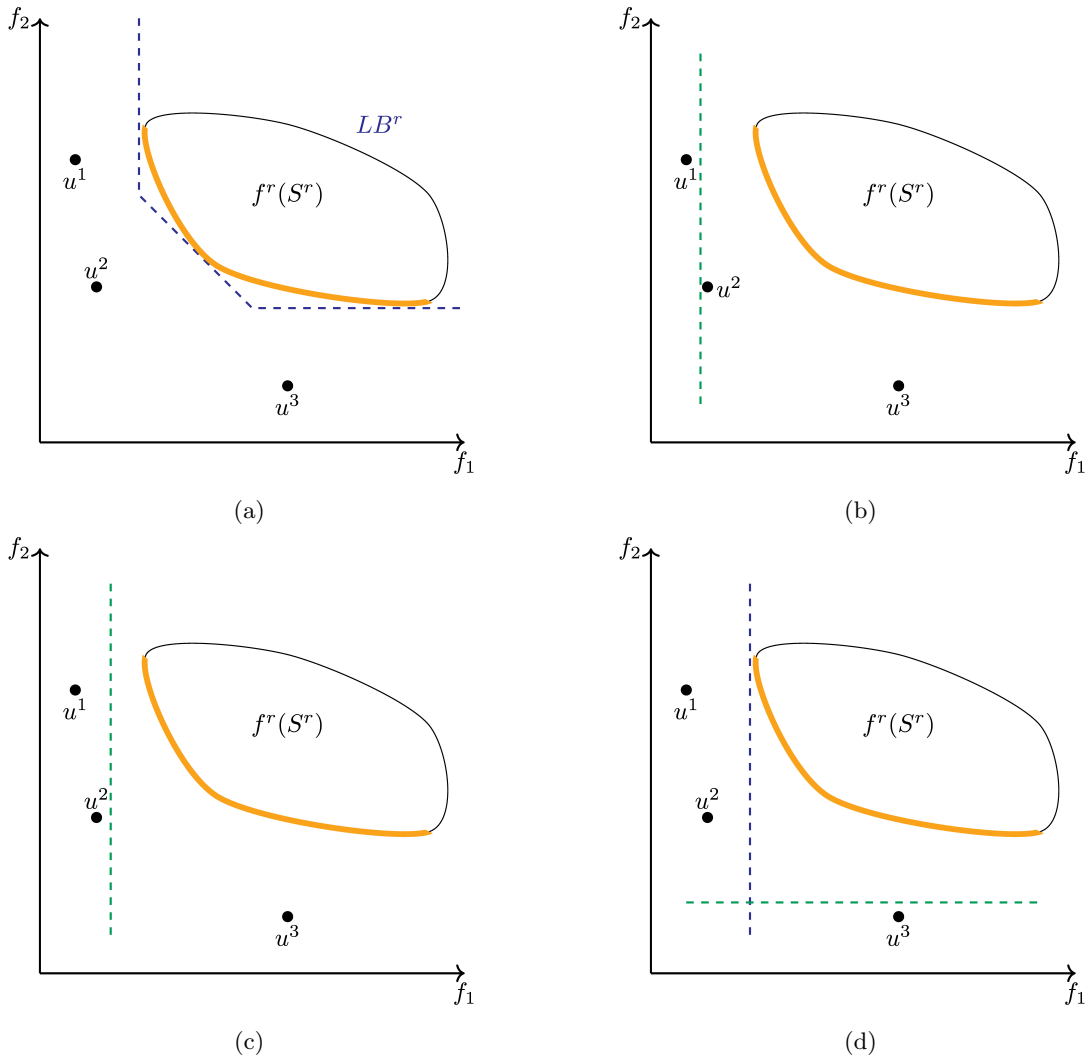

Figure 6.1: Outer approximations of $f^r(S^r)$ obtained through dual relaxations - example of *early pruning*.

experiments have been performed on an Intel® Core™ i5-14400F processor running at 2.80GHz under Linux. Each instance was addressed by `DEIA-BB` and, in case the algorithm stopped within one hour, we considered the instance solved by `DEIA-BB` and the sets $\mathcal{L}$ and $\mathcal{U}$ were considered to build an enclosure of the nondominated set.

In Tables 6.1,6.2 and 6.3 we report the results obtained running `DEIA-BB` on instances with $m = 2$, $m = 3$ and $m = 4$, respectively, checking the width of the enclosure obtained when varying the percentage of integer variables. In each table, we report a comparison among three versions of `DEIA-BB`, where we considered a different number of hyperplanes for building the linear outer approximation of $\mathcal{P}^r$. In particular, we set $|L| \in \{m, m+1, m+1+m*(m-1)\}$, having:

- $|L| = m$, if we consider the m vectors of the standard basis;
- $|L| = m+1$, if we consider the m vectors of the standard basis plus the vector $\frac{e}{|e|}$;
- $|L| = m+1+m*(m-1)$, if we consider the m vectors of the standard basis plus the vector $\frac{e}{|e|}$ and all the possible combination of vectors with two components different from zero, with one component equal to 0.25 and the other one equal to 0.75.

Therefore, for $m = 4$ we consider up to $|L| = 17$, meaning that at every node of `DEIA-BB` we need to address 17 subproblems to get the approximation of the upper image set $\mathcal{P}^r$.

For each version of `DEIA-BB`, we report the number of instances solved within the time limit (sol), the average CPU time (in seconds), the average number of nodes, the average width of the enclosure obtained and the average cardinality of the set $\mathcal{L}$. number of instances solved within the time limit among the 15 instances built for fixed n and %int. We can notice that the quality of the enclosure obtained with `DEIA-BB` improves with the percentage of integer variables considered. Note that the value reported is an average on the quality of the enclosures obtained, on instances with matrices of different density. Therefore, its interpretation is not obvious. However, as expected, the width of the computed enclosure is 0 for purely integer instances. Indeed,`DEIA-BB` is able to detect the exact nondominated set when dealing with purely integer instances. We can notice that as the cardinality of L increases, the number of nodes that `DEIA-BB` needs to explore decreases, meaning that the improved quality of the approximation of the upper image set pays off. In some cases, this has a very positive impact: note, for example, that the number of instances solved within the time limit for $n = 20$ and $m = 3$ and $m = 4$ increases with the cardinality of L. However, increasing the cardinality of L does not always correspond to a saving in terms of CPU time. This can be explained by the fact that in every node for which the pruning condition is not satisfied, a larger number of subproblems needs to be solved. The cardinality of the set L has an impact with respect to the cardinality of the lower bound set delivered by `DEIA-BB`: in general, the higher the cardinality of L the smaller is the cardinality of the lower bound set $\mathcal{L}$ (after reducing it to a stable set). We also notice that the CPU time needed by `DEIA-BB` strongly increases with the number of variables.

In Figure 6.2, we report the enclosure produced by `DEIA-BB` on an instance with $m = 2$, $n = 15$, $\%int = 75$ considering $|L| = m = 2$. The lower and upper bound sets are highlighted. The width of the obtained enclosure is 0.3859 (which is smaller than the average value obtained for all the instances having $n = 15$).

Despite `DEIA-BB` is able to compute an enclosure of the nondominated set of (MOMIQP), its quality, when dealing with mixed integer instances, cannot be controlled by any of the algorithm's parameters. This makes the comparison of `DEIA-BB` with other solvers a difficult task. We opted for showing the performance of `DEIA-BB` on purely integer instances in comparison with the performance of `AdEnA`, the hybrid decision-criterion space method proposed in [44, Algorithm

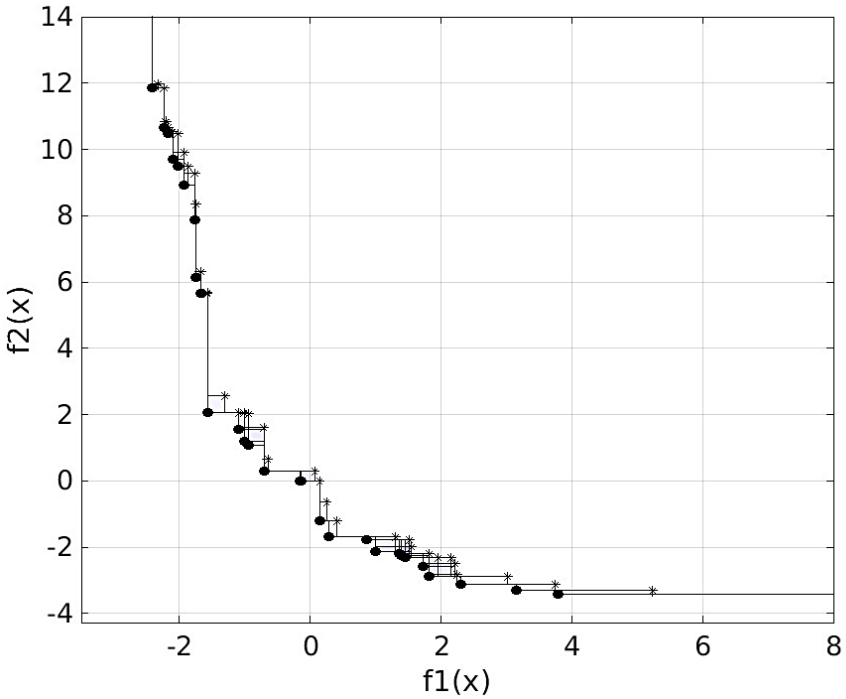

Figure 6.2: Enclosure produced by `DEIA-BB` on an instance with $m = 2$, $n = 15$, % int $=$ 75.

3]. For this, we used the code of `AdEnA` provided on GitHub [43]. By default, `AdEnA` starts with an enclosure of the nondominated set which is given as a single box and refines it until an enclosure with a prescribed width is computed. Clearly, this width cannot be set to zero, so that a fair comparison was not possible. We chose to set ϵ, the parameter controlling the quality of the enclosure released by `AdEnA`, equal to 0.1. In Table 6.4, we show results on instances with n up to 15, as we had numerical issues in calling `AdEnA` on some instances with $n = 20$ and $m = 3$ and $m = 4$. The results are averaged with respect to ρ, the parameter controlling the density of the matrices. As already mentioned `DEIA-BB` particularly suffers from an increasing number of integer variables, as the number of nodes to be visited strongly grows with respect to this number. However, the time needed by `DEIA-BB` seems to scale better with the number of objective functions than with the number of variables, so that for $m = 4$ the performance of `DEIA-BB` are comparable or even better that those of `AdEnA`.

6.7 Conclusions

We devised a branch-and-bound method able to compute a superset of the set of efficient integer assignments for multiobjective mixed-integer convex quadratic programs. The solution of dual formulations of specific subproblems combined with a corresponding preprocessing phase enables a fast enumeration of the nodes. For all the major results strong convexity of the objective functions is needed as a basic assumption. Already for linear objective functions, i.e., for convex functions which are not strongly convex, the results would fail. This can be seen by the simple example $f\colon \mathbb{R} \to \mathbb{R}^2$, with $f(x) = (x, -x)$ and feasible set $\mathbb{Z}$. All feasible points are efficient and the nondominated set is unbounded. Also for using the dual relaxations the convexity is

needed as a necessary condition for strong duality. Moreover, for explicitly stating the dual problem as done in (6.15), the analytically given unique solution $x(\lambda)$ is required which is, for nonlinear problems, in general only available for strictly convex quadratic problems. A possible way to make use of those dual bounds for more general nonlinear problems could be to first find strictly convex quadratic underestimators for the objective functions, i.e., strictly convex quadratic functions $g_j : \mathbb{R}^n \to \mathbb{R}$ with $g_j(x) \leq f_j(x)$ for all $x \in \mathbb{R}^n$ for $j = 1, \ldots, m$, and then to apply the dual lower bounding procedure on those. That would still give lower bounds for the original functions f_j.

Under the assumption of strong convexity of the objective functions, the algorithm is guaranteed to compute an enclosure of the nondominated set in a finite number of iterations. No assumption on the boundedness of the feasible set is needed. Numerical results on biobjective instances as well as on instances with three and four objectives are reported, showing the ability of the method in computing an enclosure of the nondominated set of multiobjective mixed-integer convex quadratic programs. However, `DEIA-BB` can be considered exact only in the case of purely integer instances, where the complete nondominated set is delivered. Indeed, for mixed-integer instances the quality of the enclosure obtained cannot be controlled a priori by any of `DEIA-BB` parameters and from our numerical tests we notice that such quality got worse as the number of continuous variables increases.

Table 6.1: Performance of DEIA-BB according to the cardinality of L, $m = 2$.

n	%int	$\|L\| = m$					$\|L\| = m+1$					$\|L\| = m+1+m(m-1)$				
		sol	time	nodes	width	$\text{card}_{\mathcal{L}}$	sol	time	nodes	width	$\text{card}_{\mathcal{L}}$	sol	time	nodes	width	$\text{card}_{\mathcal{L}}$
5	25	15	0.01	14.47	0.37	4.47	15	0.01	14.53	0.37	4.47	15	0.01	14.53	0.37	4.47
5	50	15	0.02	32.13	0.18	6.13	15	0.03	31.67	0.18	6.07	15	0.04	31.33	0.18	6.07
5	75	15	0.06	60.00	0.06	6.07	15	0.07	63.20	0.06	6.47	15	0.10	62.53	0.06	6.47
5	100	15	0.10	98.40	0.00	5.00	15	0.12	97.80	0.00	5.00	15	0.16	97.93	0.00	5.00
10	25	15	0.04	100.33	1.97	34.87	15	3.78	83.47	1.97	29.40	15	3.76	80.87	1.97	28.20
10	50	15	0.07	405.73	1.05	78.40	15	3.81	307.27	1.05	59.27	15	3.82	289.87	1.05	55.93
10	75	15	0.39	1486.40	0.42	71.93	15	4.15	1224.93	0.42	65.80	15	4.20	1164.07	0.42	63.33
10	100	15	1.01	2951.27	0.00	48.33	15	4.82	2672.80	0.00	48.87	15	5.05	2617.40	0.00	49.73
15	25	15	9.71	471.07	3.22	182.93	15	9.75	361.00	3.22	138.20	15	9.80	332.60	3.25	125.33
15	50	15	35.45	7245.13	1.60	1192.87	15	23.16	3666.13	1.60	553.80	15	24.96	3187.80	1.62	483.00
15	75	15	33.79	16766.53	0.49	417.27	15	26.28	10228.53	0.49	329.60	15	27.40	9139.60	0.49	301.47
15	100	15	48.82	35666.07	0.00	191.67	15	40.87	26302.53	0.00	196.40	15	40.34	24763.87	0.00	197.53
20	25	15	0.29	2338.80	4.20	870.73	15	0.25	1524.00	4.20	577.00	15	0.31	1397.60	4.20	528.20
20	50	15	593.37	111133.73	2.19	21186.07	15	432.58	32305.20	2.10	5586.13	15	480.05	27332.13	2.11	4778.80
20	75	15	900.83	553223.13	1.17	48487.60	15	684.37	134567.60	1.15	9535.33	15	810.70	112583.07	1.15	8377.93
20	100	15	970.71	579603.87	0.00	723.33	14	780.57	429258.00	0.00	698.93	14	872.44	396691.21	0.00	830.00

Table 6.2: Performance of DEIA-BB according to the cardinality of L, $m = 3$.

n	%int	$\lvert L\rvert = m$					$\lvert L\rvert = m+1$					$\lvert L\rvert = m+1+m(m-1)$				
		sol	time	nodes	width	$\text{card}_{\mathcal{L}}$	sol	time	nodes	width	$\text{card}_{\mathcal{L}}$	sol	time	nodes	width	$\text{card}_{\mathcal{L}}$
5	25	15	0.02	15.40	0.57	4.93	15	0.01	15.00	0.57	4.93	15	0.02	13.87	0.57	4.67
5	50	15	0.05	38.53	0.41	7.53	15	0.06	38.87	0.41	7.53	15	0.09	36.60	0.41	7.27
5	75	15	0.14	71.40	0.13	6.87	15	0.14	68.20	0.13	6.87	15	0.27	71.80	0.14	7.07
5	100	15	0.23	107.73	0.00	6.93	15	0.25	104.47	0.00	6.87	15	0.43	102.20	0.00	6.80
10	25	15	1.09	117.67	1.89	39.27	15	1.10	117.07	1.89	39.27	15	3.72	88.20	1.89	29.67
10	50	15	1.16	623.27	1.16	105.60	15	1.17	616.60	1.16	103.73	15	3.84	458.40	1.16	77.60
10	75	15	1.07	2378.93	0.53	112.67	15	1.18	2357.87	0.53	112.20	15	4.19	1977.87	0.53	106.73
10	100	15	4.55	5308.33	0.00	91.33	15	4.86	5297.20	0.00	90.87	15	8.73	4780.07	0.00	95.87
15	25	15	1.58	646.00	3.33	253.67	15	1.60	644.07	3.33	251.93	15	6.65	435.60	3.34	166.27
15	50	15	188.14	12864.87	1.89	1948.47	15	237.46	12486.60	1.89	1885.60	15	194.00	5645.47	1.89	866.20
15	75	15	141.90	42518.60	0.68	1334.67	15	189.53	40386.87	0.68	1275.73	15	165.58	21000.20	0.66	911.67
15	100	15	197.10	101983.20	0.00	645.53	15	249.39	98308.60	0.00	644.33	15	217.71	67315.47	0.00	659.20
20	25	15	1.30	5278.87	5.97	2313.07	15	1.40	5166.33	5.97	2257.87	15	1.71	3531.33	5.97	1507.93
20	50	13	1287.60	711606.77	3.59	179597.77	13	1429.31	649473.69	3.55	163624.62	12	1251.97	261886.42	3.40	63594.83
20	75	10	1749.49	2094593.20	1.59	227925.60	11	1868.48	2381734.91	1.69	264657.18	12	1481.86	428816.67	1.79	41700.17
20	100	10	1971.48	2390142.10	0.00	5516.90	11	2176.98	2322670.27	0.00	6353.55	11	1713.09	624516.27	0.00	6031.36

Table 6.3: Performance of DEIA-BB according to the cardinality of L, $m = 4$.

n	%int	$\|L\| = m$					$\|L\| = m+1$					$\|L\| = m+1+m(m-1)$				
		sol	time	nodes	width	$\text{card}_{\mathcal{L}}$	sol	time	nodes	width	$\text{card}_{\mathcal{L}}$	sol	time	nodes	width	$\text{card}_{\mathcal{L}}$
5	25	15	0.02	20.07	0.63	6.33	15	0.03	20.07	0.63	6.33	15	0.05	18.07	0.63	5.87
5	50	15	0.13	50.20	0.48	9.67	15	0.13	50.40	0.48	9.80	15	0.23	46.53	0.48	8.93
5	75	15	0.30	94.00	0.20	11.40	15	0.33	93.80	0.18	11.33	15	0.54	89.87	0.20	11.20
5	100	15	0.51	151.93	0.00	10.47	15	0.57	154.67	0.00	10.47	15	0.94	144.93	0.00	10.07
10	25	15	0.16	118.53	1.93	42.40	15	0.17	118.53	1.93	42.40	15	0.59	96.47	1.93	30.67
10	50	15	0.30	689.87	1.18	129.80	15	0.32	689.07	1.18	129.60	15	0.80	463.27	1.18	91.20
10	75	15	1.81	3055.80	0.54	173.27	15	1.92	3051.40	0.54	173.53	15	3.25	2470.13	0.56	159.73
10	100	15	8.59	7236.40	0.00	136.33	15	8.97	7243.67	0.00	136.47	15	12.99	6696.53	0.00	146.47
15	25	15	16.70	818.20	3.63	322.87	15	16.78	817.80	3.63	322.67	15	38.58	560.47	3.63	217.00
15	50	14	395.61	20490.79	2.08	3605.07	14	397.70	20445.07	2.08	3596.14	14	355.50	9248.79	2.08	1590.86
15	75	14	266.02	71079.36	0.80	2659.14	14	270.52	70546.14	0.80	2634.00	13	342.41	35930.00	0.80	1760.15
15	100	14	413.01	191340.07	0.00	1425.36	14	422.65	190224.00	0.00	1427.14	13	568.62	135069.23	0.00	1599.15
20	25	15	4.10	8033.00	6.24	3874.07	15	4.32	8029.47	6.24	3872.60	15	5.01	5048.27	6.24	2295.67
20	50	9	1826.44	699052.67	3.46	151797.00	9	1862.29	696639.67	3.46	151561.78	11	2163.88	545789.73	4.34	142788.55
20	75	1	2670.72	1058692.00	1.32	77642.00	1	2703.42	1039554.00	1.32	75880.00	3	2874.29	869462.33	1.97	82915.00
20	100	0	-	-	-	-	0	-	-	-	-	3	3021.39	1133625.00	0.00	17408.00

Table 6.4: Comparison between `DEIA-BB` and `AdEnA` on purely integer instances.

	n	ρ	$\lvert L\rvert = m$		$\lvert L\rvert = m+1$		$\lvert L\rvert = m+1+m(m-1)$		`AdEnA`	
			sol	time	sol	time	sol	time	sol	time
$m=2$	5	25	**5**	**0.10**	5	0.12	5	0.15	5	0.15
	5	50	**5**	**0.12**	5	0.15	5	0.19	5	0.15
	5	75	**5**	**0.09**	5	0.11	5	0.14	5	0.17
	10	25	5	0.99	5	1.02	5	1.23	**5**	**0.78**
	10	50	**5**	**0.82**	5	12.20	5	12.43	5	0.92
	10	75	5	1.23	5	1.26	5	1.47	**5**	**0.57**
	15	25	5	70.67	5	78.52	5	77.31	**5**	**2.80**
	15	50	5	43.05	5	20.52	5	22.42	**5**	**3.55**
	15	75	5	32.74	5	23.57	5	21.29	**5**	**4.85**
$m=3$	5	25	**5**	**0.25**	5	0.24	5	0.43	5	0.39
	5	50	**5**	**0.28**	5	0.32	5	0.52	5	0.73
	5	75	**5**	**0.17**	5	0.20	5	0.35	5	0.51
	10	25	**5**	**4.43**	5	4.74	5	6.22	5	6.99
	10	50	**5**	**3.15**	5	3.35	5	11.77	5	6.77
	10	75	**5**	**6.08**	5	6.48	5	8.21	5	8.63
	15	25	5	62.34	5	116.04	5	142.23	**5**	**38.68**
	15	50	5	154.92	5	246.17	5	300.08	**5**	**40.77**
	15	75	5	374.04	5	385.97	5	210.82	**5**	**77.63**
$m=4$	5	50	**5**	**0.49**	5	0.56	5	0.93	5	1.62
	5	75	**5**	**0.45**	5	0.47	5	0.85	5	1.64
	10	25	**5**	**11.55**	5	11.96	5	17.08	5	32.26
	10	50	**5**	**7.85**	5	8.17	5	11.38	5	39.58
	10	75	**5**	**6.37**	5	6.78	5	10.52	5	31.15
	15	25	4	679.33	4	684.23	4	803.42	**5**	**217.54**
	15	50	5	264.72	5	269.90	5	406.83	**5**	**252.29**
	15	75	5	348.25	5	366.13	4	536.06	**5**	**303.34**

Chapter 7

Conclusions and future work

In this thesis, we considered multiobjective optimization problems both in the purely integer and mixed-integer context. Our contribution is related to the definition of exact methods for these problems, i.e. methods able to detect the nondominated set (and/or the efficient set) of multiobjective (mixed)-integer problems (up to a certain precision in the case of mixed-integer problems).

In Chapter 3, we described assumptions under which the ε-constraint method can be proven to be exact when dealing with multiobjective integer nonlinear programming problems, showing that it needs a finite number of iterations for finding the whole Pareto frontier. In particular, it is described how the existence of a constant $\gamma \in \mathbb{R}$ bounding the distance between points in the image space of the objective function which is moved to the constraints allows us to find all the nondominated points of BOIPs. It might be interesting to use the results from [63] in order to avoid the detection of weakly nondominated points. On the other hand, such results involve adding a continuous variable and an equality constraint to the standard model which the ε-constraint method solves iteratively. This would result in a harder model to solve for the employed solver. Furthermore, when nonlinear functions are taken into account, the equality constraint added to the model would result in a nonconvex constraint, so that the application of such strategy maybe not straightforward.

In Chapter 4, we presented an application for the algorithm studied in the previous chapter. A model taking into account both the total cost and number of items of different size chosen is built in order to have a decision tool for analyzing the different solution scenarios of the raw material sizes commonality problem. A possible improvement for the model is to consider the possibility of taking advantage of the scraps. In particular, one could consider an industrial scenario for which a percentage of the scraps produced can be re-entered in the production process, thanks to a recycling industrial line. Both the activating cost and the operational cost of such line should be considered. In this scenario, a possible choice of an additional objective function could be the amount of scraps, given the choice of items, which re-enter the production cycle thanks to the recycling line, in order to evaluate whether the activating cost is paying off. This would require an algorithm able to find the set of solutions of a model having three objectives and so Algorithm 2 from Chapter 5 would be a suitable choice.

Chapter 5, devises a new exact algorithm for TOIPs based on the results from Chapter 3. The method is then effectively employed for designing sustainable diet plans. Despite some efficiency procedures, like the check for feasible points among the already-found ones described in Section

5.1.2, are already implemented, there is room for improvement for the `TrIntOpt` algorithm. In particular, it could be possible to consider one search zone at the time instead of the whole search region as the feasible set for the subproblems solved by the BOIP solver. In addition, one could think of different ways for avoiding the detection of weakly efficient points as, for example, employing a two stage method.

In Chapter 6, we devised a method for generating an enclosure of a multiobjective mixed-integer quadratic programming problem. The proposed algorithm named `DEIA-BB` is not capable of giving any guarantees of the quality of the enclosure it determines, however it allows to detect a superset of the efficient integer assignments. Hence, this weakness of `DEIA-BB` opens the possibility, as future work, of studying post-processing procedures able to refine the enclosure obtained up to a desired precision.

Bibliography

[1] Centro di ricerca alimenti e nutrizione. https://www.alimentinutrizione.it/tabelle-nutrizionali/ricerca-per-alimento. Accessed: 19/01/2023.

[2] Sustainable diets and biodiversity: directions and solutions for policy, research and action. In *International Scientific Symposium Biodiversity and Sustainable Diets United Against Hunger*. FAO, 2012.

[3] Paul Armand and Christian Malivert. Determination of the efficient set in multiobjective linear programming. *Journal of optimization theory and applications*, 70:467–489, 1991.

[4] Pietro Belotti, Christian Kirches, Sven Leyffer, Jeff Linderoth, James Luedtke, and Ashutosh Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–131, 2013.

[5] Pietro Belotti, Banu Soylu, and Margaret M Wiecek. A branch-and-bound algorithm for biobjective mixed-integer programs. *Optimization Online*, pages 1–29, 2013.

[6] Luca Benvenuti, Alberto De Santis, Marianna De Santis, and Daniele Patria. Designing sustainable diet plans by solving triobjective integer programs. *Mathematical Methods of Operations Research*, pages 1–19, 2024.

[7] Kostas Bimpikis and Mihalis G. Markakis. Inventory Pooling Under Heavy-Tailed Demand. *Management Science*, 62(6):1800–1813, June 2016. Publisher: INFORMS.

[8] Thorsten Blecker and Nizar Abdelkafi. The development of a component commonality metric for mass customization. *IEEE Transactions on Engineering Management*, 54(1):70–85, 2007.

[9] N. Boland, H. Charkhgard, and M. Savelsbergh. The L-shape search method for triobjective integer programming. *Math. Program. Comput.*, 8(2):217–251, 2016.

[10] N. Boland, H. Charkhgard, and M. Savelsbergh. The quadrant shrinking method: A simple and efficient algorithm for solving tri-objective integer programs. *Eur. J. Oper. Res.*, 260(3):873–885, 2017.

[11] Natashia Boland, Hadi Charkhgard, and Martin Savelsbergh. A new method for optimizing a linear function over the efficient set of a multiobjective integer program. *European journal of operational research*, 260(3):904–919, 2017.

[12] C. Buchheim, A. Caprara, and A. Lodi. An effective branch-and-bound algorithm for convex quadratic integer programming. *Math. progr.*, 135(1-2):369–395, 2012.

[13] C. Buchheim, M. De Santis, S. Lucidi, F. Rinaldi, and L. Trieu. A feasible active set method with reoptimization for convex quadratic mixed-integer programming. *SIAM J. Optim.*, 26(3):1695–1714, 2016.

[14] C. Buchheim, M. De Santis, and L. Palagi. A fast branch-and-bound algorithm for nonconvex quadratic integer optimization subject to linear constraints using ellipsoidal relaxations. *Operations Research Letters*, 43(4):384–388, 2015.

[15] C. Buchheim, M. De Santis, L. Palagi, and M. Piacentini. An exact algorithm for nonconvex quadratic integer minimization using ellipsoidal relaxations. *SIAM J. Optim.*, 23(3):1867–1889, 2013.

[16] Christoph Buchheim, Marianna De Santis, Francesco Rinaldi, and Long Trieu. A frank–wolfe based branch-and-bound algorithm for mean-risk optimization. *Journal of Global Optimization*, 70(3):625–644, 2018.

[17] Regina S Burachik, C Yalçın Kaya, and Mohammed Mustafa Rizvi. Algorithms for generating pareto fronts of multi-objective integer and mixed-integer programming problems. *Engineering Optimization*, 54(8):1413–1425, 2022.

[18] Regina Sandra Burachik, C Yalçin Kaya, and MM3659636 Rizvi. A new scalarization technique and new algorithms to generate pareto fronts. *SIAM Journal on Optimization*, 27(2):1010–1034, 2017.

[19] Guillermo Cabrera-Guerrero, Matthias Ehrgott, Andrew J Mason, and Andrea Raith. Bi-objective optimisation over a set of convex sub-problems. *Annals of Operations Research*, pages 1–26, 2022.

[20] Francesco Cesarone, Andrea Scozzari, and Fabio Tardella. A new method for mean-variance portfolio optimization with cardinality constraints. *Annals of Operations Research*, 205(1):213–234, 2013.

[21] Francesco Cesarone and Fabio Tardella. Equal risk bounding is better than risk parity for portfolio selection. *Journal of Global Optimization*, 68(2):439–461, 2017.

[22] Gökhan Ceyhan, Murat Köksalan, and Banu Lokman. Finding a representative nondominated set for multi-objective mixed integer programs. *European Journal of Operational Research*, 272(1):61–77, 2019.

[23] Kenneth Chircop and David Zammit-Mangion. On ε-constraint based methods for the generation of pareto frontiers. *J. Mech. Eng. Autom*, 3(5):279–289, 2013.

[24] Singa Wang Chiu, Liang Wei You, Tsu Ming Yeh, and Tiffany Chiu. The collective influence of component commonality, adjustable-rate, postponement, and rework on multi-item manufacturing decision. *Mathematics*, 8(9), 2020.

[25] Francesco Costantino, Margherita Bernabei, Silvia Colabianchi, Marianna De Santis, and Daniele Patria. Raw material size commonality decision tool: a biobjective integer programming model. submitted to Expert Systems With Applications, 2024.

[26] Kerstin Dächert, Tino Fleuren, and Kathrin Klamroth. A simple, efficient and versatile objective space algorithm for multiobjective integer programming. *Mathematical Methods of Operations Research*, pages 1–34, 2024.

[27] Marianna De Santis and Gabriele Eichfelder. A decision space algorithm for multiobjective convex quadratic integer optimization. *Computers & Operations Research*, 134:105396, 2021.

[28] Marianna De Santis, Gabriele Eichfelder, Julia Niebling, and Stefan Rocktäschel. Solving multiobjective mixed integer convex optimization problems. *SIAM Journal on Optimization*, 30(4):3122–3145, 2020.

[29] Marianna De Santis, Gabriele Eichfelder, and Daniele Patria. On the exactness of the ε-constraint method for biobjective nonlinear integer programming. *Operations Research Letters*, 50(3):356–361, 2022.

[30] Marianna De Santis, Gabriele Eichfelder, and Daniele Patria. On the exactness of the ε-constraint method for biobjective nonlinear integer programming. *Operations Research Letters*, 50(3):356–361, 2022.

[31] Marianna De Santis, Gabriele Eichfelder, Daniele Patria, and Leo Warnow. Using dual relaxations in multiobjective mixed-integer convex quadratic programming. *Journal of Global Optimization*, pages 1–28, 2024.

[32] Marianna De Santis, Giorgio Grani, and Laura Palagi. Branching with hyperplanes in the criterion space: The frontier partitioner algorithm for biobjective integer programming. *European Journal of Operational Research*, 283(1):57–69, 2020.

[33] Erik Diessel. An adaptive patch approximation algorithm for bicriteria convex mixed-integer problems. *Optimization*, 71(15):4321–4366, 2022.

[34] Ilgın Doğan, Banu Lokman, and Murat Köksalan. Representing the nondominated set in multi-objective mixed-integer programs. *European Journal of Operational Research*, 296(3):804–818, 2022.

[35] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

[36] Matthias Ehrgott. *Multicriteria Optimization*. Springer-Verlag Berlin Heidelberg, 2005.

[37] Gabriele Eichfelder. An adaptive scalarization method in multiobjective optimization. *SIAM Journal on Optimization*, 19(4):1694–1718, 2009.

[38] Gabriele Eichfelder. Scalarizations for adaptively solving multi-objective optimization problems. *Computational Optimization and Applications*, 44:249–273, 2009.

[39] Gabriele Eichfelder. Twenty years of continuous multiobjective optimization in the twenty-first century. *EURO Journal on Computational Optimization*, 9:100014, 2021.

[40] Gabriele Eichfelder, Peter Kirst, Laura Meng, and Oliver Stein. A general branch-and-bound framework for continuous global multiobjective optimization. *Journal of Global Optimization*, 80(1):195–227, 2021.

[41] Gabriele Eichfelder, Oliver Stein, and Leo Warnow. A solver for multiobjective mixed-integer convex and nonconvex optimization. *Journal of Optimization Theory and Applications*, pages 1–31, 2023.

[42] Gabriele Eichfelder and Leo Warnow. An approximation algorithm for multi-objective optimization problems using a box-coverage. *Journal of Global Optimization*, 83(2):329–357, 2022.

[43] Gabriele Eichfelder and Leo Warnow. AdEnA. https://github.com/LeoWarnow/AdEnA, 2023. Accessed 2023-09-20.

[44] Gabriele Eichfelder and Leo Warnow. Advancements in the computation of enclosures for multi-objective optimization problems. *European Journal of Operational Research*, 310(1):315–327, 2023.

[45] Gabriele Eichfelder and Leo Warnow. A hybrid patch decomposition approach to compute an enclosure for multi-objective mixed-integer convex optimization problems. *Mathematical Methods of Operations Research*, 100(1):291–320, 2024.

[46] James P Evans and Ralph E Steuer. A revised simplex method for linear multiple objective programs. *Mathematical Programming*, 5(1):54–72, 1973.

[47] Amit Eynan and Meir J. Rosenblatt. Component commonality effects on inventory costs. *IIE Transactions (Institute of Industrial Engineers)*, 28(2):93–104, 1996.

[48] C. Fjellstrom. Mealtime and meal patterns from a cultural perspective. *Scandinavian Journal of Nutrition*, pages 4161–4, 2004.

[49] Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 7.0. Technical report, Optimization Online, 2020.

[50] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020.

[51] Yacov Haimes. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE transactions on systems, man, and cybernetics*, (3):296–297, 1971.

[52] Pascal Halffmann, Luca E Schäfer, Kerstin Dächert, Kathrin Klamroth, and Stefan Ruzika. Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey. *Journal of Multi-Criteria Decision Analysis*, 2022.

[53] Kazuhiro Izui, Shinji Nishiwaki, Masataka Yoshimura, Haruki Kariya, Yoshiya Ogihara, and Shuichi Hayashi. Switchgear component commonality design based on trade-off analysis among inventory level, delivery lead-time and product performance. *International Journal of Production Research*, 48(10):2821–2840, 2010.

[54] Pubudu L.W. Jayasekara Merenchige and Margaret Wiecek. A branch and bound algorithm for biobjective mixed integer quadratic programs. *Optimization Online*, (21294), 2022.

[55] Il Yong Kim and Olivier L de Weck. Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation. *Structural and multidisciplinary optimization*, 31(2):105–116, 2006.

[56] Gokhan Kirlik and Serpil Sayın. A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research*, 232(3):479–488, 2014.

[57] N. Klamroth, R. Lacour, and D. Vanderpooten. On the representation of the search region in multi-objective optimization. *Eur. J. Oper. Res.*, 245(3):767–778, 2015.

[58] N. Klamroth, R. Lacour, and D. Vanderpooten. On the representation of the search region in multi-objective optimization. *Eur. J. Oper. Res.*, 245(3):767–778, 2015.

[59] Marco Laumanns, Lothar Thiele, and Eckart Zitzler. An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932–942, 2006.

[60] Allison M. Leach, Ariel N. Majidi, James N. Galloway, and Andrew J. Greene. Toward institutional sustainability: A nitrogen footprint model for a university. *Sustainability*, 6(4):211–219, 2013.

[61] Dinh The Luc. *Multiobjective linear programming: an introduction*. Springer International Publishing, 2016.

[62] Shihua Ma, Wei Wang, and Liming Liu. Commonality and postponement in multistage assembly systems. *European Journal of Operational Research*, 142(3):523–538, 2002.

[63] George Mavrotas. Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213(2):455–465, 2009.

[64] George Mavrotas and Danae Diakoulaki. A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research*, 107(3):530–541, 1998.

[65] George Mavrotas and Danae Diakoulaki. Multi-criteria branch and bound: A vector maximization algorithm for mixed 0-1 multiple objective linear programming. *Applied mathematics and computation*, 171(1):53–71, 2005.

[66] Christos A Nicolaou and Nathan Brown. Multi-objective optimization methods in drug design. *Drug Discovery Today: Technologies*, 10(3):e427–e435, 2013.

[67] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 1999.

[68] Institute of Medicine. *Dietary Reference Intakes for Energy, Carbohydrate, Fiber, Fat, Fatty Acids, Cholesterol, Protein, and Amino Acids*. The National Academies Press, Washington, DC, 2005.

[69] Tashina Petersson, Luca Secondi, Andrea Magnani, Marta Antonelli, Katarzyna Dembska, Riccardo Valentini, Alessandra Varotto, and Simona Castaldi. A multilevel carbon and water footprint dataset of food commodities. *Scientific Data*, 8:127, 2021.

[70] Birgit Rudloff, Firdevs Ulus, and Robert Vanderbei. A parametric simplex algorithm for linear vector optimization problems. *Mathematical Programming*, 163:213–242, 2017.

[71] Stefan Ruzika and Margaret M Wiecek. Approximation methods in multiobjective programming. *Journal of optimization theory and applications*, 126(3):473–501, 2005.

[72] Nikolaos V Sahinidis. Baron: A general purpose global optimization software package. *Journal of global optimization*, 8(2):201–205, 1996.

[73] Serpil Sayin. An algorithm based on facial decomposition for finding the efficient set in multiple objective linear programming. *Operations Research Letters*, 19(2):87–94, 1996.

[74] Elsa Silva, José Fernando Oliveira, Tiago Silveira, Leandro Mundim, and Maria Antónia Carravilla. The Floating-Cuts model: a general and flexible mixed-integer programming model for non-guillotine and guillotine rectangular cutting problems. *Omega (United Kingdom)*, 114:102738, 2023.

[75] Sakraan Sitcharangsie, T. C. Wong, and Winifred Ijomah. Bi-objective optimisation of remanufacturing decisions with component commonality under different reworking scenarios. *International Journal of Computer Integrated Manufacturing*, 34(10):1086–1107, 2021.

[76] Società Italiana di Nutrizione Umana. *Livelli di Assunzione di Riferimento di Nutrienti ed Energia per la Popolazione Italiana (LARN), 4th ed.*, 2014.

[77] Banu Soylu and Gazi Bilal Yıldız. An exact algorithm for biobjective mixed integer linear programming problems. *Computers & Operations Research*, 72:204–213, 2016.

[78] Shun Takai and Sankar Sengupta. An Approach to Evaluate the Profitability of Component Commonality. *Journal of Mechanical Design*, 139(7):1–6, 2017.

[79] Satya Tamby and Daniel Vanderpooten. Enumeration of the nondominated set of multiobjective discrete optimization problems. *INFORMS Journal on Computing*, 33(1):72–85, 2021.

[80] Aly-Joy Ulusoy, Filippo Pecci, and Ivan Stoianov. Bi-objective design-for-control of water distribution networks with global bounds. *Optimization and Engineering*, pages 1–51, 2021.

[81] Wenyuan Wang, Daniel Y. Mo, Yue Wang, and Mitchell M. Tseng. Assessing the cost structure of component reuse in a product family for remanufacturing. *Journal of Intelligent Manufacturing*, 30(2):575–587, 2019.

[82] David Williamsson and Ulf Sellgren. An Approach to Integrated Modularization. *Procedia CIRP*, 50:613–617, 2016.

[83] World Health Organization. *Healthy Diet. Fact sheet N. 394*, 2015.

[84] Panagiotis Xidonas, George Mavrotas, and John Psarras. Equity portfolio construction and selection using multiobjective mathematical programming. *Journal of Global Optimization*, 47(2):185–209, 2010.

[85] Han Zhong, Wei Guan, Wenyi Zhang, Shixiong Jiang, and Lingling Fan. A bi-objective integer programming model for partly-restricted flight departure scheduling. *Plos one*, 13(5):e0196146, 2018.

[86] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and evolutionary computation*, 1(1):32–49, 2011.