TECHNIQUES FOR MULTIOBJECTIVE OPTIMIZATION WITH DISCRETE VARIABLES: BOXED LINE METHOD AND TCHEBYCHEV WEIGHT SET DECOMPOSITION

A Dissertation
Presented to
The Academic Faculty

By

Tyler A. Perini

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Georgia Institute of Technology

Georgia Institute of Technology

August 2021

TECHNIQUES FOR MULTIOBJECTIVE OPTIMIZATION WITH DISCRETE VARIABLES: BOXED LINE METHOD AND TCHEBYCHEV WEIGHT SET DECOMPOSITION

Approved by:

Dr. Natashia Boland, Advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Martin Savelsbergh
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Santanu Dey
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Pascal Van Hentenryck
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Amy Langville
Department of Mathematics
The College of Charleston

Date Approved: June 9, 2021

Graphical excellence is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space.

Edward R. Tufte

This work is dedicated to a best friend that I am too lucky to have and who is appropriately named Star. No matter how far I go, you will always be my center point. One might say that you belong to the kernel set of my life.

ACKNOWLEDGEMENTS

I would like to first give gratitude to the all-star list of women in operations research who educated, motivated, and inspired me to become the person I am today. Special thanks to Dr. Amy Langville, Dr. Natashia Boland, Dr. Pinar Keskinocak, and Dr. Julie Swann for the opportunities, lessons, and experiences that will last a lifetime.

Second, I must share my appreciation to my brilliant and dedicated research partner, Stephan Helfrich, with whom a thrilling collaboration led to exploring insights, elevating ideas, developing timely breakthroughs, and (literally) climbing to new heights.

This work was supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1650044.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Many real-world applications involve multiple competing objectives, but due to conflict between the objectives, it is generally impossible to find a feasible solution that optimizes all, simultaneously. In contrast to single objective optimization, the goal in multiobjective optimization is to generate a *set* of solutions that induces the *nondominated (ND) frontier*. This thesis presents two techniques for multiobjective optimization problems with discrete decision variables. First, the Boxed Line Method is an exact, criterion space search algorithm for biobjective mixed integer programs (Chapter 2). A basic version of the algorithm is presented with a recursive variant and other enhancements. The basic and recursive variants permit complexity analysis, which yields the first complexity results for this class of algorithms. Additionally, a new instance generation method is presented, and a rigorous computational study is conducted. Second, a novel weight space decomposition method for integer programs with three (or more) objectives is presented with unique geometric properties (Chapter 3). The weighted Tchebychev scalarization used for this weight space decomposition provides the benefit of including unsupported ND images but at the cost of convexity of weight set components. This work proves convexity-related properties of the weight space components, including star-shapedness. Further, a polytopal decomposition is used to properly define dimension for these nonconvex components. The weighted Tchebychev weight set decomposition is then applied as a "dual" perspective on the class of multiobjective "primal" algorithms (Chapter 4). It is shown that existing algorithms do not yield enough information for a complete decomposition, and the necessary modifications required to yield the missing information is proven. Modifications for primal algorithms to compute inner and outer approximations of the weight space components are presented. Lastly, a primal algorithm is restricted to solving for a subset of the ND frontier, where this subset represents the compromise between multiple decision makers' weight vectors.

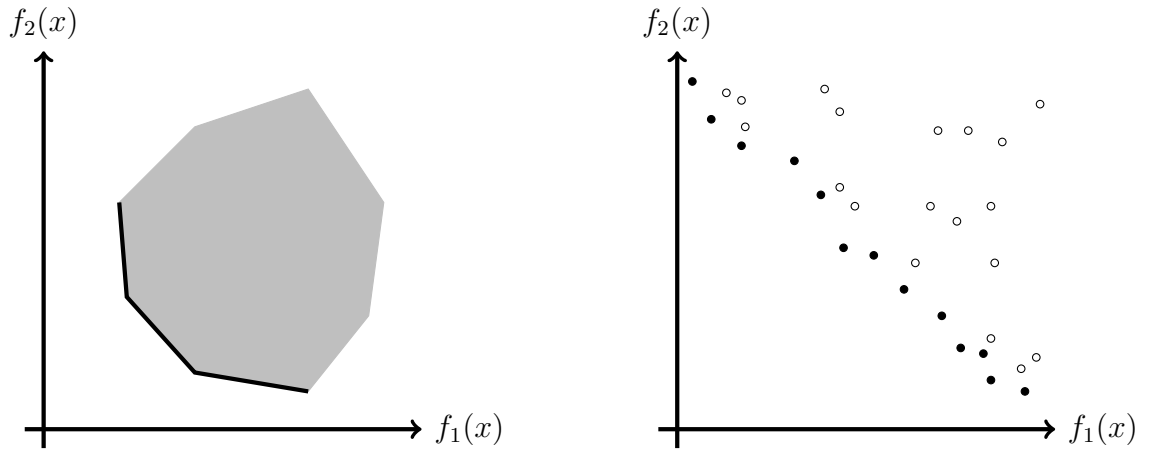
CHAPTER 1
INTRODUCTION AND BACKGROUND

Many real-world applications involve multiple competing objectives. Due to conflict between objectives, it is generally impossible to find a feasible solution that optimizes all, simultaneously. An implicit tradeoff exists between these objectives, and *multiobjective optimization* explicitly reveals this tradeoff for decision makers. Historically, multiobjective optimization is one of the earliest fields of study in operations research.

In contrast to single-objective optimization, the goal in multiobjective optimization is to generate a *set* of solutions that induces the *nondominated frontier*, also known as the *Pareto front*. An *image* refers to a vector of objective values evaluated at a feasible solution. An image is *nondominated (ND)* if there exists no other feasible solution that is at least as good in all objective values and is better in one or more of them. The ND frontier is the set of ND images.

Multiobjective optimization problems with discrete decision variables arise in many fields, including scheduling [1], geographic information systems [2], facility location [3], health care [4], and many more [5]. There has been enormous interest in these problems from the evolutionary algorithms community; see, for example, the surveys of [6, 7, 8]. This interest largely concerns problems with continuous variables; the number of evolutionary algorithms dealing with discrete variables, which will be our focus, is considerably smaller [9, 10, 11]. Algorithms based on branch-and-bound, working in the space of the decision variables, are given by [12, 13, 14].

We focus on problems with *linear* objective functions and constraints, and on *exact* algorithms, which are guaranteed, in theory, to produce the complete ND frontier. Biobjective discrete optimization problems with all continuous or all discrete variables have a simple ND frontier (Figures 1.1a and 1.1b, respectively) and have been studied for several



(a) Biobjective linear program.

(b) Biobjective integer program.

Figure 1.1: Image set $\mathcal{Y}$ for classic biobjective problems. The ND set for minimization is darkened.

decades now; see, for example, the surveys of [15, 16]. Over the last thirty years, dozens of methods for generating the ND frontier for pure integer problems have been developed. However, there are have been critical gaps in the library of multiobjective techniques for other classes of problems. We present two techniques that fill such gaps.

First, compared to pure continuous or pure integer problems, little attention has been paid to the development of computationally effective algorithms for problems that mix continuous and discrete variables. Biobjective mixed integer linear programs (BOMIPs) have only recently received vigorous interest, in part due to their additional numerical challenges. The ND frontiers of BOMIPs have a complex structure that includes elements from both the pure integer and pure discrete frontiers, as illustrated in Figure 1.2. The frontier can contain closed, open, and half-open line segments, as well as isolated points.

Second, for triobjective discrete optimization problems, much more attention has been given to criterion space search algorithms than weight space algorithms (see citations in Figure 1.4). Weight space decomposition has only been analyzed and applied for the weighted sum scalarization, which is restricted to a subset of the ND images. By excluding some proportion of ND images, the resulting weight set decomposition gives a warped

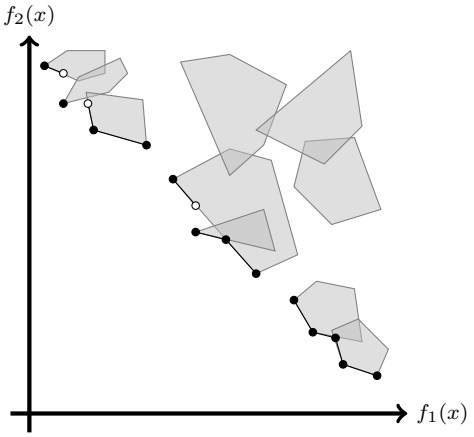


Figure 1.2: Image set $\mathcal{Y}$ for BOMIP. The ND set for minimization is darkened.

view of the value of each remaining image.

1.1 Preliminaries

A *multiobjective optimization problem (MOP)*, with $p \in \mathbb{N}, p \geq 2$ objectives can be stated as

$$\begin{aligned} \min f(x) = \min \left(f_1(x), \ldots, f_p(x)\right)^T \qquad \text{(MOP)} \\ \text{s.t. } x \in \mathcal{X}, \end{aligned}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$, for $n \in \mathbb{N}$, is called the *feasible set*, and $f = (f_1, \ldots, f_p)^T : \mathbb{R}^n \to \mathbb{R}^p$ is the (vector-valued) *objective function*. We denote by $\mathcal{Y} := f(\mathcal{X}) := \{y \in \mathbb{R}^p : y = f(x),\ x \in \mathcal{X}\}$ the *image set* and call $\mathbb{R}^n$ and $\mathbb{R}^p$ the *decision space* and the *image space* (or *criterion* or *objective* space), respectively.

Methods organized by the decision space include branch-and-bound methods, which are not covered in this thesis. Alternatively, a *criterion space search method* is organized by the image space to discover new images, which are found by existing, highly efficient black box solvers.

This thesis focuses on two subclasses of (MOP), which are distinguished by structural

properties. First, a *biobjective mixed integer linear program (BOMIP)* can be stated as (MOP) with the added restriction that $p = 2$ and $\mathcal{X} \subseteq \mathbb{Z}^{n_I} \times \mathbb{R}^{n_C}$ for finite n_I and n_C. In this case, note that vector $x \in \mathcal{X}$ has n_I integer components and n_C continuous components, and when it is convenient to differentiate between the components, we use the convention $x = (x_I, x_C)$ where $x_I \in \mathbb{Z}^{n_I}$ and $x_C \in \mathbb{R}^{n_C}$. Let the projections of $\mathcal{X}$ onto the set of integer and real vectors be defined as $\mathcal{X}_I := \{x_I \in \mathbb{Z}^{n_I} : (x_I, x_C) \in \mathcal{X} \text{ for some } x_C \in \mathbb{R}^{n_C}\}$ and $\mathcal{X}_C := \{x_C \in \mathbb{R}^{n_C} : (x_I, x_C) \in \mathcal{X} \text{ for some } x_I \in \mathbb{Z}^{n_I}\}$, respectively. Chapter 2 presents a criterion space search method for BOMIP.

Second, a *multiobjective discrete optimization problem (MODO)* can be stated as (MOP) with the added restriction that $\mathcal{X} \subseteq \mathbb{Z}^n$ for finite n. Chapters 3 and 4 focus on MODOs with $p \geq 3$ with some results that are specific to $p = 3$.

For $y, \bar{y} \in \mathbb{R}^p$, denote the following *component-wise orderings*:

- $y \leqq \bar{y}$ is defined by $y_i \leq \bar{y}_i$ for all $i = 1, \ldots, p$;
- $y \leq \bar{y}$ is defined by $y \leqq \bar{y}$ and $y \neq \bar{y}$; and
- $y < \bar{y}$ is defined by $y_i < \bar{y}_i$ for all $i = 1, \ldots, p$.

The *nonnegative orthant* is defined by $\mathbb{R}^p_{\leqq} := \{t \in \mathbb{R}^p : t \leqq 0\}$. The sets $\mathbb{R}^p_{\leq}$ and $\mathbb{R}^p_{<}$ are defined analogously. For minimization, a feasible solution x *dominates (strictly dominates)* another solution x' if $f(x) \leq f(x')$ ($f(x) < f(x')$). A feasible solution x^* is *efficient (weakly efficient)* if there does not exist another feasible solution x such that $f(x) \leq f(x^*)$ ($f(x) < f(x^*)$). An image $y = f(x)$ is *nondominated (ND)* if x is efficient, and let $\mathcal{Y}_N$ denote the set of ND images, also called the *nondominated frontier*. An image $y = f(x)$ is *weakly nondominated* if x is weakly efficient, and let $\mathcal{Y}_{wN}$ denote the set of weakly ND images. For a more detailed and thorough introduction on multiobjective optimization, see [17].

We use the relative sense of the "interior" and "boundary" points of a set. Let $B(n, y, r)$ be the unit ball with dimension n centered at y with radius r, and suppose $Y \subset \mathbb{R}^n$ where

$\dim(Y) \leq n$. Now we say y belongs to the *relative interior* of set Y, denoted $\text{rint}(Y)$, if there exists $\epsilon > 0$ such that $B(\dim(Y), y, \epsilon) \subseteq Y$. Otherwise, if no such ϵ exists, we say y belongs to the *relative boundary* of of y, denoted $\text{rbound}(Y)$.

1.2 Scalarizations

Scalarizations transform (MOP) into a single-objective problem with the help of additional parameters, such as weights or reference points. Under the proper conditions, such as nonnegativity of the weight vector, solving the resulting single-objective problem yields a single ND image. We use the term *integer program (IP)* to refer to any single-objective problem that has integer variables, including mixed integer linear programs. Here we summarize the scalarizations commonly used in the following chapters.

The *lexicographic scalarization* hierarchically minimizes the objectives in turn. We define only the biobjective case: the case of minimizing $f_1(x)$ and then $f_2(x)$ is denoted by

$$\eta = \text{lexmin}\{(f_1(x), f_2(x)) : \ x \in \mathcal{X}\}. \tag{1.1}$$

Computationally, solving a lexicographic scalarization IP requires solving p IPs in sequence. In this case,

$$\eta_1 = \min\{f_1(x) : \ x \in \mathcal{X}\} \ \text{ and then} \tag{1.2}$$

$$\eta_2 = \min\{f_2(x) : \ f_1(x) \leq \eta_1, x \in \mathcal{X}\} \tag{1.3}$$

are solved. Then $\eta = (\eta_1, \eta_2)$ is an ND image of (MOP). We note that in practice the second IP tends to solve very quickly. In general, lexicographic scalarization can be defined for any permutation of the objectives.

For vector $\lambda \in \mathbb{R}^p_{\geq}$, we refer to

$$\min\{\lambda^T f(x) : \ x \in \mathcal{X}\} \tag{1.4}$$

as the *weighted sum scalarization with respect to* λ [18]. The image of an optimal solution is (weakly) ND if $\lambda \in \mathbb{R}^p_>$ ($\lambda \in \mathbb{R}^p_\geq$) [19]. By varying the weights, alternative ND images can be found. Weighted sum scalarizations yield *supported* ND images. These are located on the convex hull of the image set. ND images that are also extreme points of the convex hull of $\mathcal{Y}$ are called *extreme supported nondominated (ESND) images*.

Let $\|y\|^\lambda_\infty := \max_{i=1,\dots,p}\{|\lambda_i\ y_i|\}$. With a *reference point* $s \in \mathbb{R}^p$ and a given weight vector $\lambda \in \mathbb{R}^p_\geq$, the *weighted Tchebychev scalarization with respect to* λ is defined as

$$\min\ \{\|f(x) - s\|^\lambda_\infty\ :\ x \in \mathcal{X}\}. \tag{1.5}$$

In practice, (1.5) is modeled with an IP by introducing an auxiliary variable. Usually, the reference point is chosen to be the *ideal point*, defined as $y^I_i := \min_{x\in\mathcal{X}} f_i(x)$ for $i = 1, \dots, p$, or to be a *utopia point* $y^U < y^I$. For weight $\lambda \in \mathbb{R}^p_>$ and reference point $s \leqq y^I$, every optimal solution to $\Pi^{TS}(\lambda)$ is at least weakly efficient for (MODO). If the solution is unique, its image is ND [20]. Conversely, there exists a positive weight vector for each ND image y' such that an optimal solution x' for the weighted Tchebychev scalarization problem satisfies $y' = f(x')$ [20].

Given weighted scalarizations (1.4) and (1.5), the *weight space* can be the organizing principle for algorithm design, much like criterion space search. By restricting weights to be nonnegative and have unit sum, the weight space is one dimension lower than the image space, i.e., $\mathbb{R}^{p-1}_\geq$. Chapters 3 and 4 present such a weight space technique.

A glossary and list of acronyms may be found in the end materials of this dissertation.

1.3 Dichotomic Search

We present the classic method of dichotomic search as an instructive example to highlight a procedure for (MOP) that is relevant throughout this dissertation. This procedure was co-discovered and published by Cohon [21] and Aneja and Nair [22] in the late 1970s. See

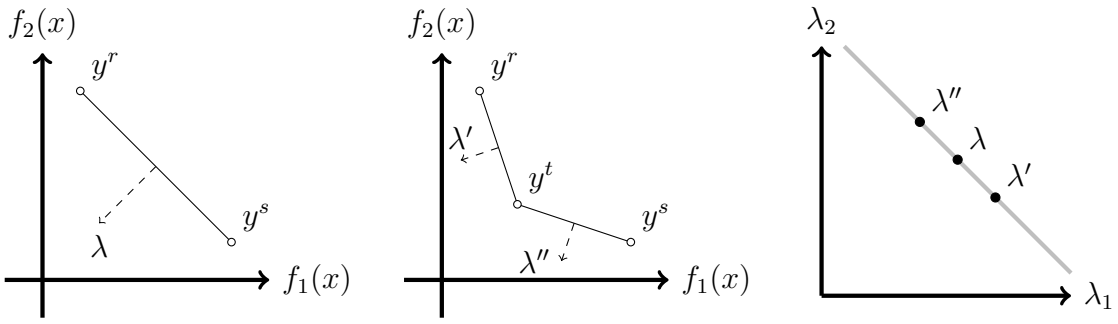


Figure 1.3: Dichotomic search for biobjective (MOP), adapted from [23]. The gradient vectors are illustrated in the weight space in (c).

[23] for a detailed summary of the procedure, which is illustrated here and in Figure 1.3.

Given a biobjective (MOP) (e.g., Figure 1.1), *dichotomic search* iteratively solves the weighted sum scalarization (1.4) to discover all ESND images. At each step of the procedure, a pair of ESND images are compared: either they are certified as adjacent on the convex hull of $\mathcal{Y}$, or a new ESND image is found which proves them to be non-adjacent. The procedure is initialized with the two "extreme" ND images, say $y^r = \text{lexmin}(f_1(x), f_2(x)) : x \in \mathcal{X}$ and $y^s = \text{lexmin}(f_2(x), f_1(x)) : x \in \mathcal{X}$. Then the (positive) gradient vector for the line between y^r and y^s is computed, denoted by λ, and used as the weight vector for the weighted sum scalarization (1.4). Figure 1.3(a) illustrates this subproblem. One of two cases occur:

First, if the optimal value of (1.4) with respect to λ is less than $\lambda^T y^r = \lambda^T y^s$, then a new ESND image has been found. Denote the image by y^t. Now, two new pairs of images must be tested for adjacency: (y^r, y^t) and (y^s, y^t); see Figure 1.3(b). Second, if the optimal value of (1.4) with respect to λ is equal to $\lambda^T y^r = \lambda^T y^s$, then no new ESND image has been found. The two images, y^r and y^s, are certified as adjacent. Dichotomic search is generally implemented recursively, and it terminates when the full set of ESND images are found and all adjacencies are certified.

For a multiobjective linear program, it is well-known that every efficient solution is supported [24]. The ND frontier for the biobjective linear program can then be described

as a set of line segments between adjacent ESND images; see Figure 1.1a. Therefore, we say that the dichotomic search procedure completely "solves" biobjective linear programs. The following two observations are relevant for the chapters to come.

There are various reasons one may want to terminate the dichotomic search procedure before all ESND images are found or certified. In particular, if not all line segments from $\mathrm{conv}(\mathcal{Y})$ (each segment defined by a pair of ESND images) are necessary, then many of the subproblems may be skipped. This "restricted" dichotomic search is the basic premise of the line segment generation routine presented in Chapter 2.

The gradient vectors, while used as a means to compute new ESND images or certify adjacent pairs, may be viewed on their own for extra insights. The set of nonnegative weight vectors with unit sum is shown in gray in Figure 1.3(c), and the weight vectors $\lambda, \lambda', \lambda''$ are illustrated in black. If we assume adjacency between image pairs (y^r, y^t) and (y^s, y^t), as shown in Figure 1.3(b), then we can conclude that for any weight vector $\lambda \in \mathrm{conv}\,(\lambda', \lambda'')$, the optimal value for (1.4) with respect to λ is $\lambda^T y^t$. This perspective from the weight space and the partitioning of the weight set are fully explored in Chapters 3 and 4.

1.4 Motivating Applications

The following examples describe multiobjective applications which motivate the classes of (MOP) we study.

Example 1 (BOMIP)**.** *A recent study [25] examines a biobjective hub-airport location problem, which analyzes the network containing larger airports ("hubs") and smaller airports ("nodes"), specifically in Turkey. The decision makers are investors, and the objectives are to minimize the total transportation cost and the total cost of using 2-stop routes. The former objective reflects the preference of the airline companies, which is to prioritize the efficient routing through hubs, while the latter reflects the preference of the customers, which is to avoid unnecessary stops by flying directly between nodes. The resulting model is a BOMIP. For small instances, the authors apply an exact BOMIP algorithm [26] that*

preceeded the Boxed Line Method presented in Chapter 2 (wherein we show our algorithm performs better). For larger instances, the authors utilized a heuristic approach.

Example 2 (MODO)**.** *Recent work [27] studies urban freight distribution. Introducing pollution related objectives into traditional vehicle routing problems is becoming more common and is referred to as the pollution-routing problem. These problems are more sophisticated because they incorporate time-varying traffic conditions, vehicle payload, and fuel consumption. This work studes the Steiner pollution-routing problem variant. The triobjective model aims to minimize (1) vehicle hiring cost, (2) total amount of fuel consumed, and (3) total duration of the routes. The authors note that while this is a triobjective mixed integer linear program, since only one objective function contains continuous variables, then the ND frontier is a discrete set just as in MODO.*[1] *They use the Quadrant Shrinking Method [29], a criterion space search method, to solve for the ND set.*

Example 3 (Multiple decision makers)**.** *The McRow method proposed by [30] is defined for multiobjective linear programs with multiple experts or decision makers. Given a set of weights, the McRow formulation returns a single solution to a min-max problem which minimizes the "dissatisfaction" of the most dissatisfied expert. They primarily apply the weighted sum scalarization but also note that this can be replaced by the weighted Tchebychev scalarization. As an example, they implement their methodology to a disaster relief application for a volcano in Indonesia.*

This final example, in particular, implicitly uses the weight space for (MOP) algorithm design. Unlike the image space, the weight space gives a more intuitive perspective of preference weights, including *dissatisfaction* and *compromises*. However, the McRow returns a single solution, whereas we propose to return all the solutions associated with the set of expert weights. Chapter 4 formalizes this as an application of the novel weight space decomposition.

[1] Similar to the conclusion made in [28].

1.5 Timeline of Algorithms

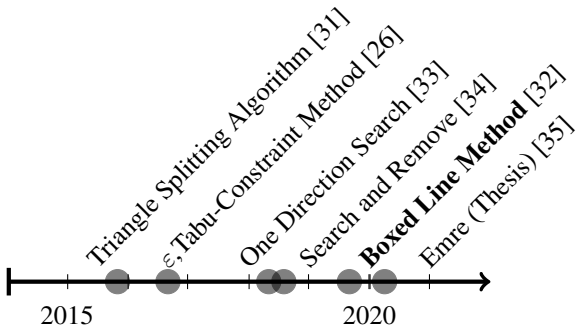
We have so far discussed subclasses of (MOP), including one that has been solvable since the 1970s (i.e., biobjective linear programs by dichotomic search). We briefly and visually summarize algorithms for the other relevant subclasses studied in this thesis. See Figure 1.4.

Figure 1.4a includes only exact criterion space search algorithms for BOMIP; to the best of our knowledge, the timeline is complete. In 2015, the Triangle Splitting Algorithm [31] was the first to be published.[2] As seen from the timeline, its publication was followed by a flurry of attention to this class of problems. From 2015 to 2020, at least one new BOMIP algorithm was published each year. The Boxed Line Method [32] is presented in Chapter 2.

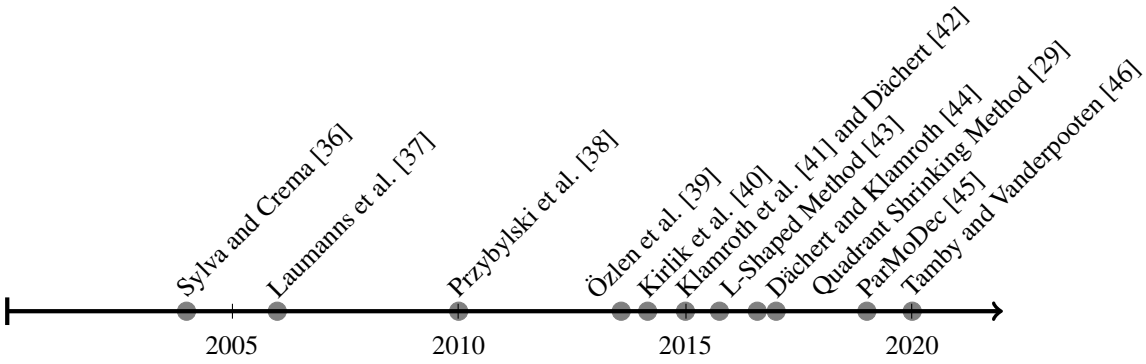
For MODOs with $p = 3$, Figure 1.4b gives only a representative sample of the published criterion space search algorithms. Indeed, the true number of studies is massive. Over time, the algorithms have become more and more sophisticated in the decomposition of the image space and the subproblems solved. Rather than contributing to this class of algorithms, we instead adapt the algorithm from Klamroth et al. [41] to contribute a new weight space decomposition algorithm (see Chapter 4).

In contrast to criterion space search algorithms, the history of weight space search algorithms for MODOs is much more sparse. Figure 1.4c highlights only the algorithms that explicitly compute the weight set decomposition (or an approximation thereof). All of these methods were defined for weighted sum scalarization. This motivated research into a new weighted scalarization (Chapters 3 and 4).

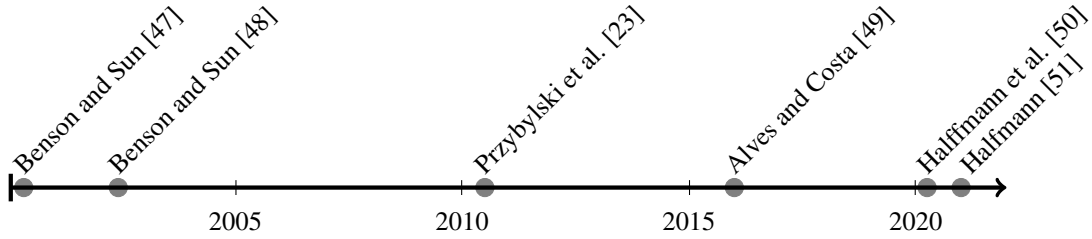
[2]Since its publication, the Triangle Splitting Algorithm has since been cited 56 times. source: Google Scholar.



(a) Timeline of criterion space search algorithms for BOMIPs. The Boxed Line Method is presented in Chapter 2.



(b) Timeline of (a sample of) criterion space search algorithms for MODOs with $p = 3$.



(c) Timeline of weight space decomposition algorithms for MOPs.

Figure 1.4: Timelines of related MOP algorithms.

1.6 Contributions

Chapter 2 presents a criterion space search algorithm developed for BOMIP called the Boxed Line Method. This algorithm generalizes a previous algorithm for biobjective integer programs to include continuous variables. The algorithm improves upon previous BOMIP algorithms in multiple ways: BLM provides a parsimonious representation of the

ND set (rather than splitting line segments into smaller segments), better approximates the ND set in partial run-time, and shows superior performance in computational experiments. The algorithm has two variants that permit analysis of the number of single-objective IPs that they require to be solved: a basic, iterative version and a recursive version. For both variants, we provide upper bounds on the number of single-objective IPs required to produce the ND frontier. These were the first analytic results of this type for multiobjective mixed integer problems. In addition, a new class of instances was developed for study, and thorough computational results were presented. This research was conducted in collaboration with Diego Pecin, Natashia Boland, and Martin Savelsbergh: Tyler Perini developed the algorithm design, complexity analyses, and instance generation, and Diego Pecin performed the computational study. The research has been published in INFORMS Journal of Computing [32] and received the 2020 INFORMS Computing Society Student Paper Prize.

Chapter 3 applies the weight set decomposition approach to the weighted Tchebychev scalarization of MODOs, which has not been done before for more than two objectives. The weighted Tchebychev scalarization implies a more sophisticated and rich structure in comparison to the weighted sum scalarization. The primary contribution of this work is a theoretical study of this structure and its properties, which provides a foundation for the development of new algorithms to compute the weighted Tchebychev weight set decomposition, which in particular includes the enumeration of all ND images of a MODOs. The weight set components are shown to be nonconvex but have convexity related properties: they are star-shaped as well as convex along rays emanating from a vertex of the weight set. The intersections of weight set components are shown to coincide with weakly ND images, and hence all convexity-related properties also apply, even though intersections of star-shaped sets are, in general, not star-shaped. Weight set components are described as unions of finitely many polytopes, which lays a well-defined foundation of the dimensional analysis, including an adjacency structure for weight set components. This research was conducted in collaboration with Stephan Helfrich, Pascal Halffmann, and Stefan Ruzika

at Technical University of Kaiserslautern in addition to Natashia Boland. Tyler Perini and Stephan Helfrich co-discovered many of the geometric properties of this weight space decomposition in their PhD research: Tyler Perini contributed mainly to the study of intersection sets, and Stephan Helfrich contributed mainly to the convexity and polytopal analysis.

Chapter 4 extends the work of Chapter 3, maintaining focus on the weighted Tchebychev scalarization. The concept of local nadir points are generalized and fully developed, and an existing method to compute them is adapted to satisfy the needs of the weight space decomposition. These methods are proven to be sufficient to compute weight set components and cover the weight set. A triangularization method is presented which eliminates redundancy for computing exact area of weight set components. Furthermore, this research explores the tight relationship between the hyperrectangular level set of the weighted Tchebychev scalarization and the "boxes" used by algorithms to decompose the image space for MODOs. Insights from this weight space decomposition technique are used to modify classic MODO algorithmic approaches. By doing so, inner and outer approximations of the weight set components can be produced during run-time. Additionally, computation can be reduced by solving for ND images associated with a subset of the weight set which represents the compromise between a set of weights. Simulated computational experiments are presented. This research was also conducted in collaboration with Stephan Helfrich, Pascal Halffmann, and Stefan Ruzika at Technical University of Kaiserslautern in addition to Natashia Boland. Tyler Perini contributed to the applications to primal algorithms and the computational study; Stephan Helfrich contributed to the proof of coverage; and both contributed equally to the design of algorithms and theoretical results.

1.7 Outline

This dissertation discusses techniques for MOP with discrete variables. Table 1.1 summarizes the topics covered by each chapter. Chapter 2 presents the Boxed Line Method in the context of BOMIP. Chapter 3 explores fundamental geometric properties of the weighted

Tchebychev weight space decomposition method for MODOs with three or more objectives. Chapter 4 applies the weight space decomposition with algorithms adapted to computing the weight set components exactly, computing approximations of the components, and restricting criterion space search algorithms via the weight space. Chapter 5 concludes with special computational considerations and motivations for future work.

Table 1.1: Chapter contents.

Chapter 2	**Chapter 3**	**Chapter 4**
$p = 2$	$p \geq 3$	$p = 3$
Mixed Integer	Pure Integer	Pure Integer
Criterion Space	Weight Space	Weight Space
Algorithm, Theoretical Results	Theoretical Results	Algorithm, Theoretical Results

CHAPTER 2

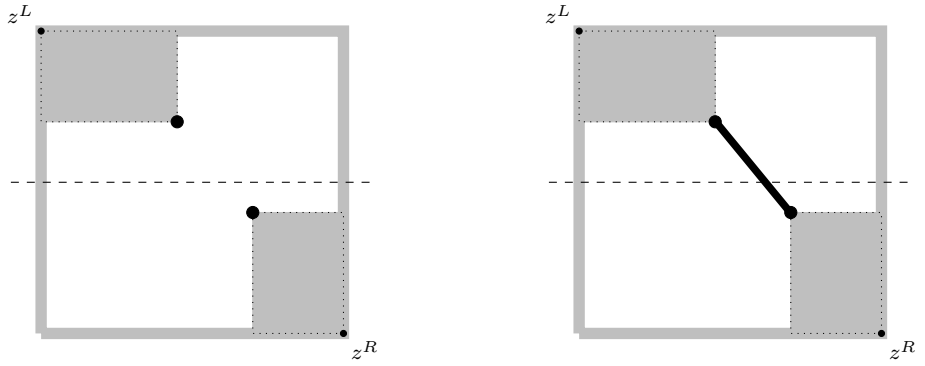
BOXED LINE METHOD: AN EXACT METHOD FOR BIOBJECTIVE MIXED INTEGER PROGRAMS

Abstract. *Despite recent interest in multiobjective integer programming, few algorithms exist for solving biobjective mixed integer programs. We present such an algorithm: the Boxed Line Method. For one of its variants, we prove that the number of single-objective integer programs solved is bounded by a linear function of the number of nondominated line segments in the nondominated frontier; this is the first such complexity result. An extensive computational study demonstrates that the Boxed Line Method is also efficient in practice, and that it outperforms existing algorithms on a diverse set of instances.*[1]

Recently, criterion space methods, in which the search for the nondominated (ND) frontier operates over the space of the vector of objective function values, known as the *criterion space*, have emerged. Such methods have the advantage of being able to exploit advances in single-objective solver software, since these methods repeatedly solve single-objective problems, both linear programs (LPs) and mixed integer linear programs (which we will generically refer to as *IPs*), treating the single-objective solver as a "black box". Single-objective problems, either LPs or IPs, are the main "workhorses" of these algorithms, which differ mainly in the structure and number of such problems that need to be solved before the ND frontier is completely identified.

This chapter focuses on criterion space search algorithms for biobjective mixed integer linear programs (BOMIPs). The first of such algorithms to be published was the *Triangle Splitting Algorithm (TSA)* [31]. The TSA first identifies all extreme supported nondominated (ESND) images using dichotomic search [21, 22], which allows the remaining search region to be divided into right-angled triangles. Each triangle is then split, either horizon-

[1]This work was published in *INFORMS Journal on Computing* (2019) [32], coauthored with Natashia Boland, Diego Pecin, and Martin Savelsbergh, with Tyler Perini as first author. This work received the INFORMS Computing Society Student Paper Prize in 2020.



(a) ND images are filled. (b) ND line segments are darkened.

Figure 2.1: The key step in the Balanced Box Method (a) and the Boxed Line Method (b). The remaining search regions (boxes) after the step are shaded.

tally or vertically, with each half searched for a ND image so as to reduce the remaining search region within the triangle to two rectangles. Within each rectangle, all ESND images are found (these are extreme and supported only in the local sense, i.e., within the rectangle), and the process repeats. Line segments in the ND frontier are identified when they form part of the hypotenuse of a triangle. TSA may split a line segment in the ND frontier. This can occur even if the line segment is part of the frontier of a unique *slice problem*, which is a biobjective LP defined by fixing the integer part of the solution. The TSA thus requires a post-processing procedure to provide a parsimonious description of the ND frontier.

The second algorithm published was the *ε,Tabu-Constraint Method (εTCM)* of [26], which uses "tabu", or no-good, constraints to identify line segments in the ND frontier, combined with ε-constraints to progressively generate the frontier from right-to-left (or vice versa).

In this chapter, the following are key contributions.

1. We propose a new criterion space search method for solving the BOMIP: the Boxed Line Method (BLM). The method is designed to generalize the *Balanced Box Method (BBM)* [52], which is a computationally effective method for MODO limited to two

objectives. The BLM defaults to BBM in the absence of continuous variables. The key step of the BBM is illustrated in Figure 2.1a: the rectangular search region (box) is split, each half is searched for a ND image, which reduces the remaining search region to two boxes having total area less than half that of the original box. To apply this idea to BOMIP, we observe that when the split line passes through a line segment of the frontier, the ND image found when the first half of the box is searched will lie on the split line. In the BLM, we seek to extend this ND image to the line segment in the ND frontier which contains it. Using the endpoints of this line segment gives a remaining search region consisting of two boxes, with combined area less than half that of the original box. Figure 2.1b illustrates this idea. As a consequence of the idea, the resulting algorithm is amenable to analysis (discussed next) and produces a parsimonious description of the ND frontier.

2. The algorithm has two variants that permit analysis of the number of single-objective IPs that they require to be solved: a basic, iterative version and a recursive version. For both variants, we provide upper bounds on the number of single-objective IPs required to produce the ND frontier. These are the first analytic results of this type for mixed integer multiobjective problems.

3. We design an enhancement that takes advantage of a property of the ND frontier encountered in many BOMIP instances, which can provide a significant improvement in algorithm runtime.

4. The benchmark instances originally proposed by [12], which have been used to test recent methods, have a structure that is very sensitive to the numerical tolerances used in algorithms. In these instances, the slice problems often have many short line segments in their frontiers and include line segments that are close to horizontal or close to vertical. The instances also have frontiers represented by a relatively small number of slice problems, which can bias comparisons of algorithms if one of the

algorithms is designed to exploit this structure. Here, we propose a new approach to creating instances, in a controlled way. The approach facilitates validation of BOMIP algorithms, by producing instances for which the frontiers are known precisely, *a priori*. It also supplements the existing suite of test instances by providing instances having different characteristics, such as having many slice problems contributing to the frontier.

5. We provide computational results that demonstrate the relative strengths and weaknesses of the BLM variants on two classes of the new instances proposed here, as well as on instances from [12]. The results in the latter case are compared with the results of the TSA [31] and in both cases with the εTCM [26].

This chapter is structured as follows. In Section 2.1, we introduce unique structures of the ND frontier for a BOMIP. The basic variant of the BLM is described in Section 2.2. In Section 2.2.5, we prove that the method has an upper bound for the number of IPs. We continue by presenting the recursive extension to the method in Section 2.3 along with its better upper bound for the number of IPs in Section 2.3.4. The enhancement designed for a structure common to many BOMIP instances is discussed in Section 2.4. In Section 2.5, we discuss numerical issues in implementation and how they relate to the structure of existing benchmark instances. Then in Section 2.6, we present our new instance generator method. Finally, we give our computational study in Section 2.7 and summarize our findings.

2.1 Structures of a BOMIP Frontier

The ND frontier of a BOMIP can be described by *nondominated line segments, vertical gaps*, and *horizontal gaps*. Define $L(z^1, z^2)$ to be the line segment connecting endpoints $z^1, z^2 \in \mathbb{R}^2$, i.e. $L(z^1, z^2) := \{\xi z^1 + (1-\xi)z^2 : 0 \leq \xi \leq 1\}$ where the endpoints are ordered so that $z^1_1 \leq z^2_1$. As defined, $L(z^1, z^2)$ is a *closed* line segment. When $z^1 \neq z^2$, we can also have an open line segment (neither z^1 nor z^2 are part of the line segment) and a

half-open line segment (either z^1 or z^2 is not part of the line segment); we refer to z^1 and z^2 as *closed*, if they belong to the line segment, and as *open* otherwise. A line segment $L(z^1, z^2)$ is a point when $z^1 = z^2$.

A gap may appear in the ND frontier between two ND images in the vertical or horizontal direction, or both; this is obvious for MODOs for which all ND images are isolated points, but it is not a given fact when given mixed continuous and integer variables. Between a ND image and an open endpoint of a line segment in the ND frontier there must appear either a vertical or a horizontal gap. We define a *vertical gap* as an interval $(y^-, y^+) \subset \mathbb{R}$ such that no ND image p exists with $p_2 \in (y^-, y^+)$ but where there does exist a ND image p^- with $p_2^- = y^-$ and either a ND image p^+ with $p_2^+ = y^+$ or a sequence of ND images $\{p^0, p^1, ...\}$ with $\lim_{n\to\infty} p_2^n = y^+$ (or both). A *horizontal gap* can be defined similarly.

Given $x_I \in \mathcal{X}_I$, the biobjective LP (BOLP) obtained from fixing the integer variables to x_I is called the *slice problem for* x_I [53]. The ND frontier of a BOLP slice problem consists of a (connected) set of (closed) line segments; we call this a *slice*. The ND frontier of a slice problem for x_I is called the *slice for* x_I[2].

We now develop a proper definition for nondominated line segments, including how they should be enumerated for our complexity results. Define S to be the index set of all feasible integer solutions in $\mathcal{X}_I$. For a slice problem with index $s \in S$, we denote its slice by N^s. If a slice problem with index $s \in S$ contributes to the ND frontier of the BOMIP, so $N^s \cap \mathcal{N}$ is nonempty, then we write $N^s \cap \mathcal{N} = \{L_1^s, L_2^s, ..., L_{n(s)}^s\}$, where $n(s)$ is the number of line segments contributed to the ND frontier by the slice problem. (If $N^s \cap \mathcal{N}$ is empty, then $n(s) = 0$.) Each of the line segments, L_i^s for some $s \in S$ and $i = 1, \ldots, n(s)$, is a *nondominated line segment (NDLS)*. Note that a single maximal line segment in N^s may contribute several NDLSs to the ND frontier, consisting of non-overlapping sections of the line segment. We make the natural assumption that the set $N^s \cap \mathcal{N}$ consists of maximal

[2]Note that our definition of a slice differs from the original definition by [53], where it is defined as the *feasible set* for the slice problem as opposed to the resulting ND frontier.

line segments, where maximal refers to set inclusion, so $N^s \cap \mathcal{N}$ provides the minimum cardinality set of line segments that describes the slice's intersection with the ND frontier of the BOMIP. For a given BOMIP, we define the total number of NDLSs in the ND frontier as

$$n^* := \sum_{s \in S} n(s). \tag{2.1}$$

2.2 Basic Method

Here we present the fundamental principles of the BLM. The four main components of the algorithm are the initialization, *outer loop, inner loop*, and *line segment generation subroutine*. In this chapter, we assume exact arithmetic. For example, we describe some constraints as strict inequalities. These are implemented, in practice, as inequalities with the right-hand side adjusted using the desired accuracy, $\epsilon > 0$. The pseudocode provided in the appendix makes this explicit. In practice, the numerical issues naturally encountered in solving integer programs, which are even more pressing for mixed integer programs, make it unreasonable to expect to determine the ND frontier exactly. In Section 2.5, we define an approximation – an ϵ-nondominated frontier – and discuss the numerical challenges that arise in finding it.

2.2.1 Initialization

The first stage, initialization, solves two lexicographic scalarization IPs of the form (1.1) to determine the upper-leftmost ND image, $z^L = \text{lexmin}\{(f_1(x), f_2(x)) : x \in \mathcal{X}\}$, and the lower-rightmost ND image, $z^R = \text{lexmin}\{(f_2(x), f_1(x)) : x \in \mathcal{X}\}$. In this chapter, we refer to a rectangular region in criterion space as a "box" and describe it by its upper-leftmost and lower-rightmost corner points. The entire ND frontier must lie in the box with z^L and z^R as its two corner points, which we denote by $B(z^L, z^R)$. Note that if $z^L = z^R$, then $B(z^L, z^R)$ is a point. We call a box with $z^L_i = z^R_i$ for $i = 1$ or 2 (or both) *trivially small*. If the box $B(z^L, z^R)$ is trivially small, then the method terminates and one point is

returned as the full ND frontier. Otherwise, two points are added to the current subset of the ND frontier, which we call $\tilde{\mathcal{N}}$, and the box $B(z^L, z^R)$ is added to the queue, which we call $\mathcal{Q}$, for further processing by the outer loop. In general, the basic method only adds boxes to the queue if both corner points are the closed endpoints of a NDLS to which they belong (we discuss this further in Section 2.2.5).

2.2.2 Outer Loop

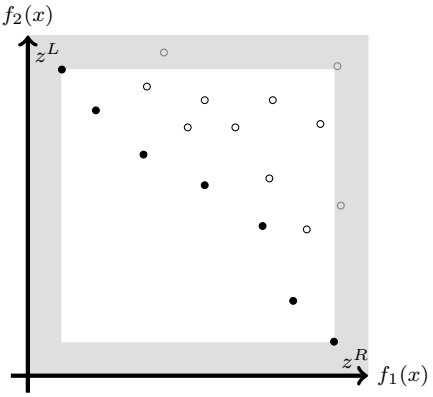
To introduce the outer loop of the basic version of the BLM, we first describe the process for the Balanced Box Method [52] because, in the first case we consider, the two algorithms follow the same procedure. The steps are visually summarized in Figure 2.2.

The outer loop is defined as a while loop that ends once $\mathcal{Q}$ is empty. The main roles of the outer loop are to remove boxes from $\mathcal{Q}$ for processing, split the boxes to begin processing, (if necessary) call the inner loop to complete processing, and update $\tilde{\mathcal{N}}$ with any found ND images and $\mathcal{Q}$ with remaining boxes for future processing. Suppose a (nontrivial) box $B(z^L, z^R)$ is arbitrarily chosen from $\mathcal{Q}$ where z^L and z^R now represent the corner points of the new box (not necessarily the ND images found by initialization). The outer loop begins processing by choosing an arbitrary horizontal split line $z_2 = \mu$ where $\mu \in (z_2^R, z_2^L)$. We solve a lexicographic IP for a ND image on or below the split line, namely

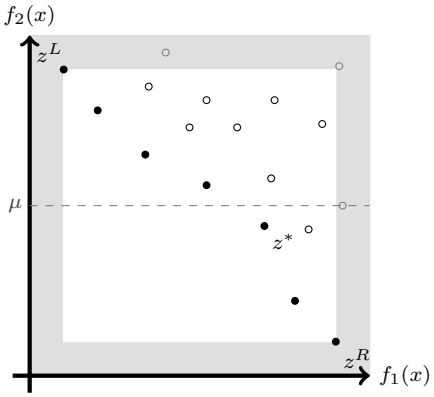
$$z^* = \text{lexmin}\{(f_1(x), f_2(x)) : f_2(x) \leq \mu, x \in \mathcal{X}\}. \tag{2.2}$$

We note that the solution $x^R \in \mathcal{X}$, which maps to $z^R = f(x^R)$, is feasible for (2.2), so in practice we provide x^R as an initial feasible solution to the solver. The next step is to form a second split line, this time vertically at z_1^* and solve a second lexicographic minimization[3]

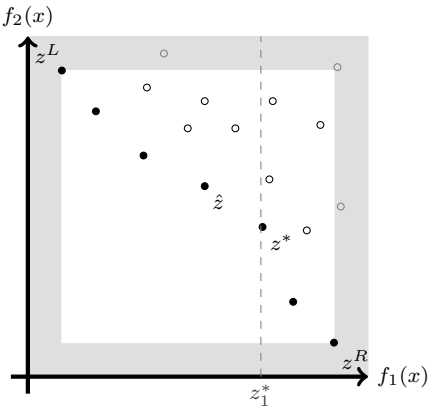
[3]The strict inequality used is a convenient shorthand: it is meant to be interpreted and implemented as $f_1(x) \leq z_1^* - \epsilon$ for some $\epsilon > 0$.



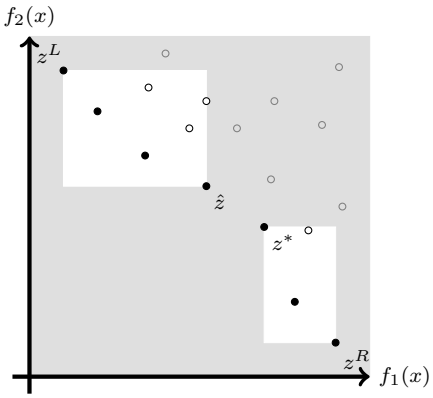
(a) Initialization identifies z^L and z^R and defines the white box to the queue. The outer region in grey is ignored.



(b) A horizontal split line $z_2 = \mu$ is chosen such that $\mu \in (z_2^R, z_2^L)$. Lexicographically minimizing $f_1(x)$ then $f_2(x)$ below the split line yields z^* where $z_2^* < \mu$.



(c) A vertical split line is chosen at z_1^*. Lexicographically minimizing $f_2(x)$ then $f_1(x)$ (strictly) to the left of the split line yields $\hat{z}$.



(d) The approximation of the ND frontier, $\tilde{\mathcal{N}}$, is updated with $\hat{z}$ and z^*, and the two new boxes (in white) are added to the queue, $\mathcal{Q}$.

Figure 2.2: Outer loop procedure for the Balanced Box Method and the BLM when $z_2^* < \mu$.

for the next ND image,

$$\hat{z} = \text{lexmin}\{(f_2(x), f_1(x)) : f_1(x) < z_1^*, x \in \mathcal{X}\}. \tag{2.3}$$

The solution $x^L \in \mathcal{X}$ that maps to $z^L = f(x^L)$ is feasible for (2.3), so in practice, x^L is provided as an initial feasible solution to the solver.

Finally, the outer loop updates $\tilde{\mathcal{N}}$ and $\mathcal{Q}$, accordingly. First, z^* and $\hat{z}$ are added to $\tilde{\mathcal{N}}$. Then, z^* and $\hat{z}$ are used respectively with z^R and z^L to define two new boxes, $B(z^L, \hat{z})$ and

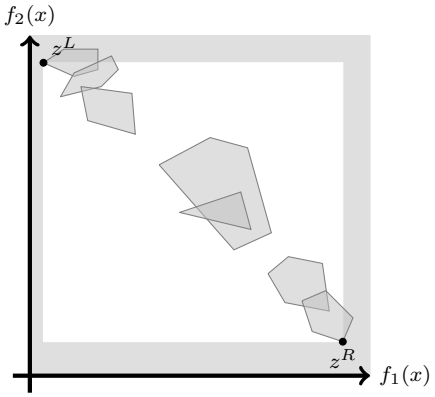
$B(z^*, z^R)$, each of which is added to the queue if it is not trivially small. This concludes the processing of one box by the outer loop when $z_2^* < \mu$.

The BLM follows the above procedure when the first lexicographic minimization (2.2) returns ND image z^* such that $z_2^* < \mu$. Note that when there are discrete variables, it is possible for an arbitrary horizontal split line $z_2 = \mu$ to not intersect any ND image, which results in $z_2^* < \mu$. In this case, we say that the outer loop has *identified a vertical gap* in the ND frontier. Once a vertical gap is identified by the outer loop, the boxes resulting from processing are such that this vertical gap is excluded from any subsequent boxes added to the queue (we prove this formally in Section 2.2.5).

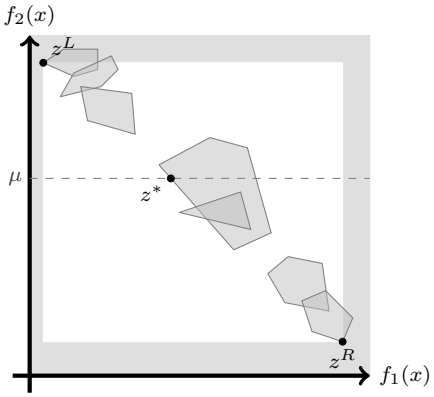
For a BOMIP with continuous variables, it is likely that the horizontal split line will intersect an NDLS. In this case, the first lexicographic minimization (2.2) returns ND image z^* such that $z_2^* = \mu$. The rest of the procedure is designed to *identify the NDLS* that contains z^*, i.e., to find $L(z^1, z^2) \subset B(z^L, z^R)$ such that $z^* \in L(z^1, z^2)$. This concept motivates the name for the BLM.

The outer loop calls the inner loop, which returns $L(z^1, z^2)$ (details on how this is done are given in the Section 2.2.3). The inner loop returns endpoints z^1 and z^2, as well as the ND image that dominates each open endpoint, if any. The outer loop updates the current approximation of the ND frontier by adding $L(z^1, z^2)$ to $\tilde{\mathcal{N}}$ and updates the queue by adding up to two nontrivial boxes. Since BLM only creates boxes for which both corner points are ND, we initialize the boxes depending on the openness of the endpoints as follows. If z^1 is closed, then $B(z^L, z^1)$ is added to $\mathcal{Q}$. Otherwise the inner loop has returned a ND image, $\hat{z}^1$, that dominates z^1, and the box $B(z^L, \hat{z}^1)$ is added to $\mathcal{Q}$. Similarly, if z^2 is closed, then $B(z^2, z^R)$ is added to $\mathcal{Q}$. Otherwise, the inner loop has returned a ND image, $\hat{z}^2$, that dominates z^2, and the box $B(\hat{z}^2, z^R)$ is added to $\mathcal{Q}$.

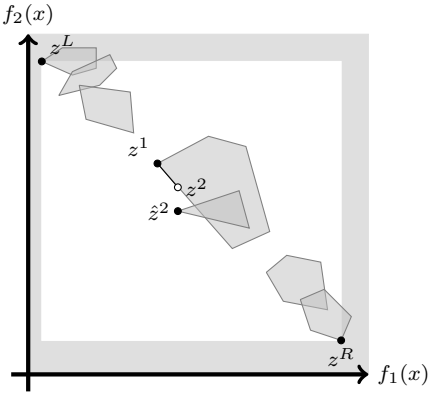
This concludes the processing of one box by the outer loop when $z_2^* = \mu$, which is summarized in Figure 2.3. The pseudocode is included in the Appendix, as Algorithm 4.



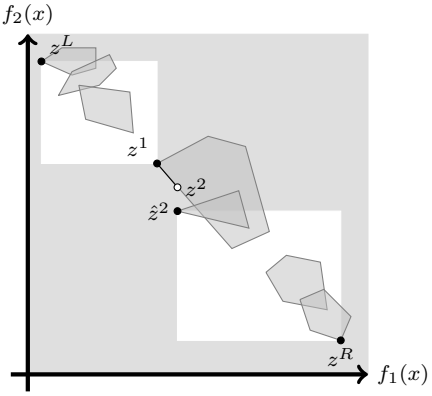
(a) Suppose the box $B(z^L, z^R)$ is chosen from $\mathcal{Q}$.



(b) Minimizing $f_1(x)$ then $f_2(x)$ below the split line μ yields z^* where $z_2^* = \mu$.



(c) The Inner Loop returns the NDLS containing z^*, indicating whether each endpoint z^1 and z^2 are either open or closed, and the point(s) that dominates its open endpoint(s) (e.g. $\hat{z}^2$ dominates z^2).



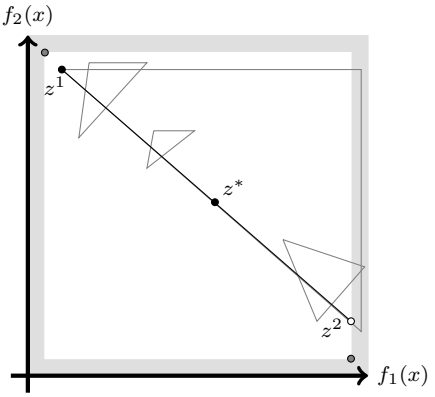
(d) The approximation of the ND frontier is updated with line segment $L(z^1, z^2)$, and two new boxes are added to the queue, e.g., $B(z^L, z^1)$ and $B(\hat{z}^2, z^R)$ (assuming they are nontrivial).

Figure 2.3: Outer loop procedure for the BLM when $z_2^* = \mu$.

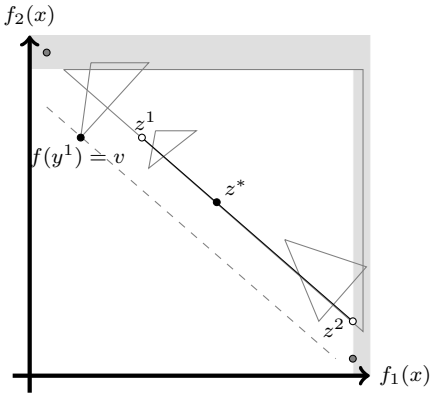
2.2.3 Inner Loop

The inner loop is called when the outer loop is processing box $B(z^L, z^R)$ and the split line contains a ND image, z^*, i.e. when $z_2^* = \mu$. The inner loop finds the NDLS, $L(z^1, z^2)$, that contains z^*. Its steps are visually summarized in Figure 2.4, and the pseudocode is presented in the Appendix as Algorithm 5.

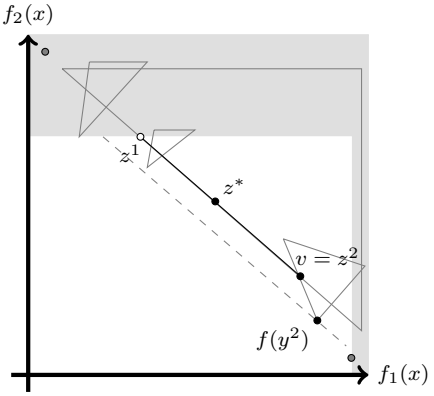
Let $x^* = (x_I^*, x_C^*) \in \mathcal{X}$ be an optimal solution to (2.2) that maps to $z^* = f(x^*)$. The first step in the inner loop is to provide an *overestimation* for the NDLS by generating



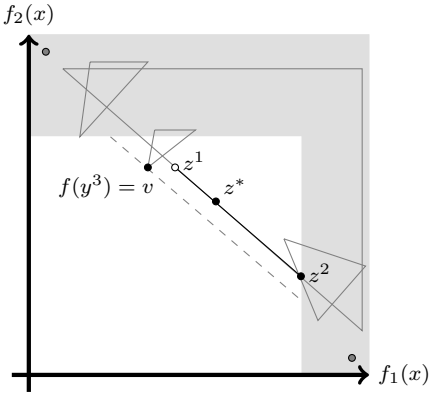
(a) The gray corner points of the white box are z^L and z^R. The darkened line is the approximation of the line segment immediately after line generation subroutine. Note that z^2 is open because it lies directly above z^R: $z_1^2 = z_1^R$ and $z_2^2 > z_2^R$.



(b) Solving (2.5) (within the white region bounded by the previous z^1,z^2) identifies $f(y^1)$, which is also equal to v. We update z^1 accordingly, so it is now open.



(c) Solving (2.5) again (within the white region bounded by the previous z^1,z^2) identifies $f(y^2)$. Since v is at the intersection of the slice containing z^* with the line segment, v is equal to the new z^2, which is now closed.



(d) Solving (2.5) yields $f(y^3)$, then solve for v. Update z^1, and one final weighted sum IP confirms that $L(z^1, z^2)$ is ND. Although z^1 is open, in this case the ND image that dominates it is already known, i.e. $\hat{z}^1 = f(y^3)$.

Figure 2.4: The inner loop takes as input ND images z^L, z^R, and z^*. The output is the (maximal) NDLS containing z^*, $L(z^1, z^2)$, and the ND image that dominates each open endpoint (if any).

the line segment containing z^* in the slice for x_I^*, restricted to $B(z^L, z^R)$. We discuss the details of the line segment generation subroutine in the Section 2.2.4. Here we assume the following is given: the maximal line segment $L(z^1, z^2)$ such that $z^* \in L(z^1, z^2)$, $z_1^L \leq z_1^1 \leq z_1^2 \leq z_1^R$, $z_2^L \geq z_2^1 \geq z_2^2 \geq z_2^R$, and z^1, z^2, and z^* are all in the slice for x_I^*.

Since $L(z^1, z^2)$ is part of the slice for x_I^*, it must be that either (i) $z^1 = z^2$ or (ii) $z_1^1 < z_1^2$ and $z_2^1 > z_2^2$. In the case that $z^1 = z^* = z^2$, the slice consists of the isolated ND image z^*, so the inner loop ends and returns $z^1 = z^*$ and $z^2 = z^*$ (both closed).

Otherwise, $z^1 \neq z^2$, and the gradient vector of the line segment, $L(z^1, z^2)$, normalized to have unit length, is defined as

$$\vec{w} = (z_2^1 - z_2^2, z_1^2 - z_1^1)/||(z_2^1 - z_2^2, z_1^2 - z_1^1)||. \tag{2.4}$$

In what follows, each weighted sum IP (1.4) is computed with $\lambda = \vec{w}$.

Next, the endpoints of $L(z^1, z^2)$ are restricted, iteratively, until only the ND portion of it remains. Either or both ends are "trimmed" while always maintaining z^* as the central point between z^1 and z^2, ensuring that the final line segment contains z^*. Until the inner loop terminates, each iteration updates the endpoints of the line segment, z^1 or z^2 (both their coordinates and their open/closed flag), with any newfound knowledge. For instance, immediately after the line generation subroutine provides $L(z^1, z^2)$, both endpoints are flagged as closed unless the following cases occur: (i) $z_1^R = z_1^2$ and $z_2^R < z_2^2$, illustrated in Figure 2.4a, in which case the endpoint z^2 is dominated by z^R, so z^2 cannot be closed and is flagged as open, and (ii) $z_2^L = z_2^1$ and $z_1^L < z_1^1$, in which case z^1 is flagged as open.

The inner loop is a while loop that continues until the entirety of the current line segment, $L(z^1, z^2)$, is ND. To check this stopping condition, the following weighted sum IP,

$$f(y^*) = \min\{\vec{w}^T f(x) : f_1(x) \leq z_1^2, \;\; f_2(x) \leq z_2^1, \;\; x \in \mathcal{X}\}, \tag{2.5}$$

is solved, where the first inequality is strict if z^2 is open and the second inequality is strict if z^1 is open. These *conditionally strict* inequalities are required to avoid cycling. Now $L(z^1, z^2)$ is ND if and only if (2.5) yields an optimal solution, y^*, with $\vec{w}^T f(y^*) = \vec{w}^T z^*$ (a formal proof of this is provided in the Appendix). Hence, $\vec{w}^T f(y^*) = \vec{w}^T z^*$ is the stopping criterion for the while loop.

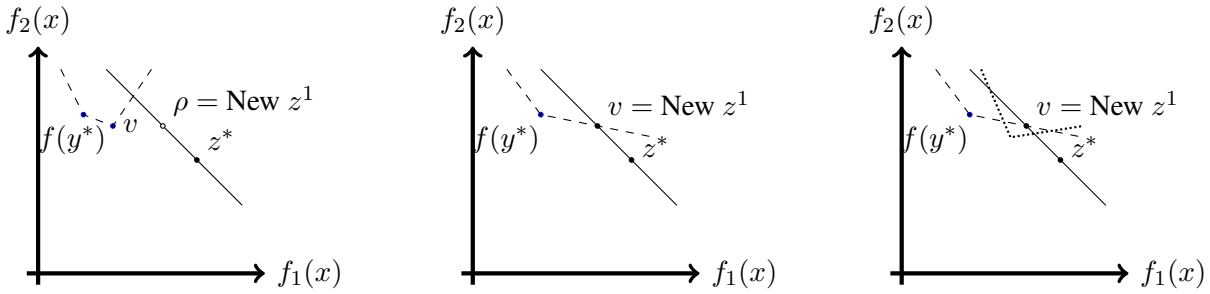
In the case that $\vec{w}^T f(y^*) \neq \vec{w}^T z^*$ it must be that $\vec{w}^T f(y^*) < \vec{w}^T z^*$, since z^* in $L(z^1, z^2)$ ensures that the feasible solution mapping to z^* is feasible for (2.5). Thus, the ND image $f(y^*)$ dominates a portion of the current line segment. There are two cases: either $f(y^*)$ is to the left or it is to the right of z^*. Without loss of generality, assume $f_1(y^*) < z_1^*$ (the other case follows from a symmetric argument). There may be many other points from the slice for y_I^* that also dominate part of $L(z^1, z^2)$: z^1 is updated to exclude all such points within the box by finding the point with integer solution y_I^* that dominates part of $L(z^1, z^2)$ and has minimal z_2-coordinate. Hence, we solve the LP

$$\min\{f_2(x) : \ \vec{w}^T f(x) \leq \vec{w}^T z^*, \ \ x_I = y_I^*, \ \ x \in \mathcal{X}\}. \tag{2.6}$$

Let y' be an optimal solution to (2.6) and define $v = f(y')$. We highlight some observations about the LP defined in (2.6).

- Minimizing $f_2(x)$ ensures all points on the slice for y_I^* that can dominate the current line segment are excluded when z_2^1 is updated. The idea is illustrated in Figure 2.5a.
- Even though the point v may lie on the line segment $L(z^1, z^2)$, it will never be *dominated* by any point from $L(z^1, z^2)$ because of the first constraint, $\vec{w}^T f(x) \leq \vec{w}^T z^*$. Consider Figure 2.5b.
- While the first scalarization returns a ND image $f(y^*)$, the point v may be dominated. Consider Figure 2.5c. This issue will be discussed later.
- As it is an LP, (2.6) can be solved efficiently. Furthermore, since the solution y^* is feasible for (2.6), in implementation we provide this initial feasible solution to the solver, which yields even greater efficiency.

Next, v is used to update z^1. First, suppose $v \notin L(z^1, z^2)$. Point $\rho \in L(z^1, z^2)$ such that $\rho_2 = v_2$ is computed and then z^1 is updated: $z^1 = \rho$ and flagged to be open (see Figure 2.5a). Otherwise, $v \in L(z^1, z^2)$ and z^1 is updated to $z^1 = v$. Before updating the



(a) The slice for y_I^* is dashed. Minimizing $f_2(x)$ on this slice finds v. Updating the upper endpoint z^1 to the point ρ (as open) will prevent y_I^* recurring when solving (2.5).

(b) The minimization to find v along the slice for y_I^* cannot surpass $L(z^1, z^2)$ because of the constraint $\vec{w}^T f(x) \leq \vec{w}^T z^*$. Since $v \in L(z^1, z^2)$, z^1 is updated as $z^1 = v$ and flagged to be closed.

(c) The point v in this example is dominated by a point from another slice (dotted)
.

Figure 2.5: Observations about the construction of LP (2.6).

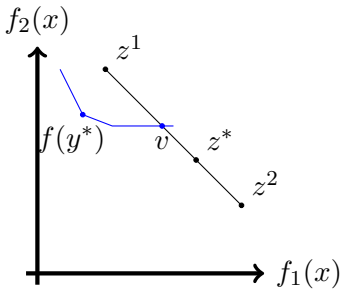
open/closed flag for z^1, we need to check for the possibility that v is a weakly ND point on the slice for y_I^*, as in Figure 2.6. If so, then the new z^1 will be open and otherwise it will be closed. To check this, the algorithm solves a second LP,

$$\min\{f_1(x) : \ f_2(x) = v_2, \ \ x_I = y_I^*, \ \ x \in \mathcal{X}\}. \tag{2.7}$$

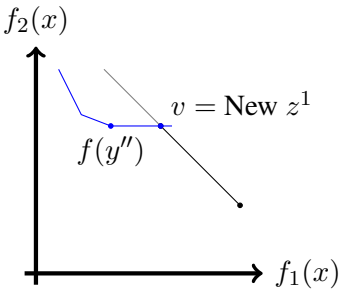
Let y'' be an optimal solution of (2.7) and update $v = f(y'')$. (Note that even this updated v is still not necessarily a ND image of the BOMIP). If $v_1 < z_1^1$, then the algorithm flags z^1 as open, and otherwise it is flagged as closed.

After updating z^1 as described, then, by convexity, it is guaranteed that no point from the slice for y_I^* can again be found. Since $\mathcal{X}$ is bounded, which implies that $\mathcal{X}_I$ is finite, the inner loop will terminate in a finite number of iterations. The resulting endpoints z^1 and z^2 will then define the NDLS containing z^*, $L(z^1, z^2)$.

As a final step, the inner loop finds each ND image that dominates an endpoint that is open. Let $\hat{z}^1$ denote the ND image that dominates z^1 when z^1 is open, and let $\hat{z}^2$ denote the ND image that dominates z^2 when z^2 is open. To determine $\hat{z}^i$ for $i \in \{1, 2\}$, two special



(a) $f(y^*)$ dominates points from the line segment $L(z^1, z^2)$. The LP (2.6) has multiple optimal solutions. Finding v on the current line segment requires checking for an alternative optimum before updating the open/closed flag for z^1.



(b) Solving (2.7) yields y'' with $f_1(y'') < v_1$, so the updated z^1 is flagged to be open.

Figure 2.6: The case that v lies on the current line segment and is a weak ND image of the slice problem for y^*.

cases are checked first. (1) If $z_2^1 = z_2^L$, then clearly $\hat{z}^1 = z^L$ (and similarly for z^2 and z^R). (2) If $z_2^1 = f_2(y^*)$ then $\hat{z}^1 = f(y^*)$ (see Figure 2.4d; and similarly if $z_1^2 = f_1(y^*)$, then $\hat{z}^2 = f(y^*)$). Otherwise, the inner loop identifies these ND images by solving the IP

$$\min\{f_1(x) : f_2(x) \leq z_2^1,\ x \in \mathcal{X}\}, \tag{2.8}$$

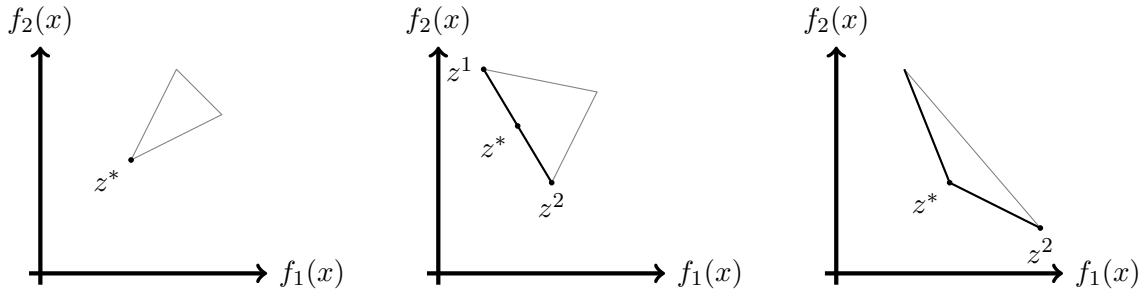
if z^1 is open, and by solving the IP

$$\min\{f_2(x) : f_1(x) \leq z_1^2,\ x \in \mathcal{X}\}, \tag{2.9}$$

if z^2 is open. If z^1 is open, $\hat{z}^1 = f(\hat{x}^1)$, where $\hat{x}^1$ is an optimal solution to (2.8), and if z^2 is open, $\hat{z}^2 = f(\hat{x}^2)$, where $\hat{x}^2$ is an optimal solution to (2.9).

2.2.4 Line Segment Generation Subroutine

The purpose of the line segment generation subroutine is to provide the inner loop with a single line segment that can be reduced to an NDLS by simply updating its endpoints. The subroutine determines an overestimate of the NDLS containing a given ND image,



(a) The ND image z^* dominates all other solutions with the same integer solution: it is an isolated ND image and $z^1 = z^2 = z^*$.

(b) The ND image belongs to exactly one line segment of its slice (darkened): $L(z^1, z^2)$.

(c) The ND image belongs to exactly two line segments of the slice (darkened). The line segment generation subroutine finds the one to the lower right: $L(z^*, z^2)$.

Figure 2.7: Three possible cases for finding a line segment of the slice containing z^*.

$z^* = f(x^*)$, within $B(z^L, z^R)$. Specifically, it finds a maximal line segment, $L(z^1, z^2)$, of the slice problem for given integer solution x_I^* with $z^* \in L(z^1, z^2)$. Such a segment is an NDLS of the slice problem $\min\{(f_1(x), f_2(x)) : x_I = x_I^*, x \in \mathcal{X}\}$. Figure 2.7 illustrates the three possible cases for z^*: z^* may be an isolated point or it may belong to one or two line segments of the slice. When z^* belongs to two line segments, the subroutine finds the segment to the lower right, i.e., it takes $z^1 = z^*$ (see Figure 2.7c).

Finding the maximal line segment, $L(z^1, z^2)$, of the slice problem for given integer solution x_I^* with $z^* = f(x^*) \in L(z^1, z^2)$ can be done, in theory, by accessing the values of the dual variables at the optimal branch-and-bound node of the IP that found z^* and applying LP duality theory. However, we are not confident that, in practice, the IP solver will provide the LP dual variables needed, since it may have found z^* with a primal heuristic, eliminated variables and/or constraints or modified coefficients with preprocessing, or added constraints. Therefore, we simply use a variant of dichotomic search [21, 22], restricted to a box around z^* of size sufficiently large to ensure that the gradient of any line segment that contains z^* and lies within the box can be calculated accurately.

The BLM requires the line generation subroutine to produce three interrelated forms of

output. Primarily, it seeks the two endpoints of the line segment, z^1 and z^2. In addition, as long as $z^1 \neq z^2$, it seeks the gradient vector, $\vec{w}$, of the line segment $L(z^1, z^2)$. When $z^1 = z^2$, we say that $\vec{w}$ does not exist; we note that when the gradient vector $\vec{w}$ exists, it is normalized before it is returned. Obviously, if we have z^1 and $z^2 \neq z^1$, then it is trivial to compute $\vec{w}$. However, if $\vec{w}$ is discovered first, it is also possible to find each endpoint by solving the following LPs:

$$x^1 = \arg\min\{f_1(x) : \vec{w}^T f(x) \leq \vec{w}^T z^*, f_2(x) \leq z_2^L, x_I = x_I^*, x \in \mathcal{X}\}, \text{ and} \tag{2.10}$$

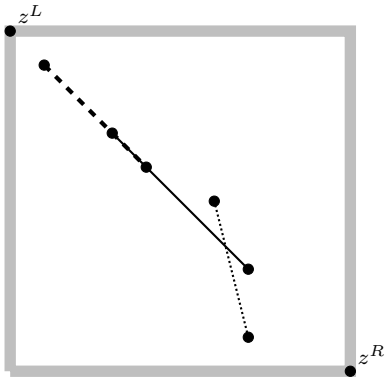
$$x^2 = \arg\min\{f_2(x) : \vec{w}^T f(x) \leq \vec{w}^T z^*, f_1(x) \leq z_1^R, x_I = x_I^*, x \in \mathcal{X}\}. \tag{2.11}$$

Then $z^1 = f(x^1)$ and $z^2 = f(x^2)$. (By construction, z^1 or z^2 should not be outside the box $B(z^L, z^R)$, since the box's corner points are ND images.) Therefore, the subroutine either finds *both* endpoints *or* the gradient of the line segment, and then the remaining output can be computed accordingly. Because of this, Algorithm 8, giving details of the subroutine, is flexible in the order in which these outputs are discovered. The algorithm generally searches for z^2 first, and in the case that neither z^2 nor $\vec{w}$ are found, then it assumes $z^2 = z^*$ and searches for z^1.

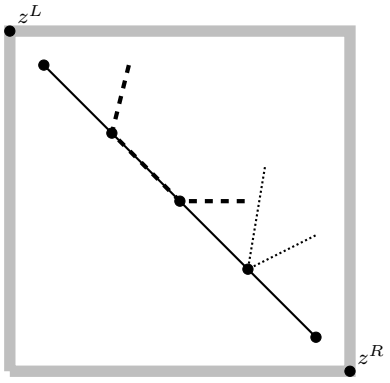
2.2.5 Complexity of the Basic Method

In this section, we prove a worst-case upper bound on the number of IPs solved in order to generate the entire ND frontier. We do so by describing the number of IPs solved as a function of the number of NDLSs in the ND frontier.

Consider a BOMIP where the feasible set, $\mathcal{X}$, is nonempty and bounded (see (MOP)). Recall that $\mathcal{X}_I$ is the projection of $\mathcal{X}$ onto the set of integer vectors, and S is the index set of $\mathcal{X}_I$. Let $S_{\mathcal{N}}$ be the index set of feasible integer solutions whose slice has nonempty intersection with the ND frontier, $\mathcal{N}$, i.e. $S_{\mathcal{N}} = \{s \in S : N^s \cap \mathcal{N} \neq \emptyset\}$. Recall that for all $s \in S_{\mathcal{N}}$, we write $N^s \cap \mathcal{N} = \{L_1^s, L_2^s, ..., L_{n(s)}^s\}$, where, for each $i = 1, \ldots, n(s)$,



(a) The cases above are "allowed": overlapping and intersecting NDLSs from distinct slices, resulting in $n = 3$.



(b) The cases above are assumed to not occur: a NDLS from one slice contained within the NDLS from another. Without the dashed and dotted slices, $n = 1$ and $g = 2$.

Figure 2.8: Examples of ND frontiers where we assume there is no containment between NDLS of distinct slices, and how to count the number of NDLS, n, using (2.12).

L_i^s is a (distinct, maximal) line segment. We make one technical assumption for ease of exposition: for all distinct pairs of feasible integer solutions, $s, t \in S_{\mathcal{N}}$, we assume that $L_i^s \not\subseteq L_j^t$ and $L_j^t \not\subseteq L_i^s$ for all $i = 1, \ldots, n(s)$ and $j = 1, \ldots, n(t)$). We note that in benchmark problems, (point) intersections between distinct slices in the ND frontier are relatively common, but *containment* of an NDLS from one slice inside that from another is relatively rare. Such containments do not prevent the algorithm from functioning correctly and returning the entire ND frontier, but the counting of NDLSs becomes much more complicated. This assumption does not rule out intersections or even *overlap* between slices; compare Figures 2.8a and 2.8b.

In (2.1), we defined the total number of line segments by $n^* = \sum_{s \in S_{\mathcal{N}}} n(s)$. For the following proofs, we use $n(Y)$ to represent the total number of line segments in a given (nontrivial) box, $Y := B(z^L, z^R)$, where z^L and z^R are ND images. We take $n(Y)$ to be the number of line segments (including isolated ND images) in the strict interior of Y, which excludes z^L, z^R:

$$n(Y) := \sum_{s \in S} |N^s \cap \mathcal{N} \cap Y \setminus \{z^L, z^R\}|. \tag{2.12}$$

See Figure 2.8 as an example of counting NDLS with this new definition. In addition, let $g(Y) \geq 0$ be the number of vertical gaps in the ND frontier within Y. Note that since a vertical gap is defined by its two incident ND images, it must be that $g(Y) \leq n(Y) + 1$ for all Y.

For simplicity of exposition, we drop the notation for dependence on Y, i.e., we use n and g instead of $n(Y)$ and $g(Y)$, respectively. We will compute the specific bounds on the number of IPs solved for different cases of Y with given n and g. Let $\ell(n, g)$ be the *worst case* number of lexicographic optimization IPs solved by the basic method in *completely* processing an arbitrary box $Y = B(z^L, z^R)$ with n NDLSs in the strict interior of Y and g vertical gaps in the ND frontier. By "completely" processing, we mean processing any resulting boxes added to the queue after processing Y, processing any resulting boxes added to the queue after that, and so on until no more (nontrivial) boxes remain in the queue. Similarly, let $s(n, g)$ be the *worst case* number of single-objective optimization IPs (e.g., scalarized IPs) solved to completely process such a box Y. In general, we assume that no trivial regions Y would appear from the queue because trivial boxes are not added to the queue. Therefore, we define $\ell(0, 0) = s(0, 0) = 0$.

Example 4. *For $n = 0$ and $g = 1$, $\ell(0, 1) = 2$ and $s(0, 1) = 0$.*

Proof of Example 4. For any split line $z_2 = \mu$ where $z_2^R < \mu < z_2^L$, we have that the first lexicographic optimization IP (5) returns z^R. Since we have $z_2^R < \mu$, the algorithm continues by solving a second lexicographic optimization IP (6), which yields z^L. Note that no new regions are added to the queue (because $B(z^L, z^L)$ and $B(z^R, z^R)$ are trivially small), so we have that the iteration of the outer loop terminates having solved $\ell(0, 1) = 2$ lexicographic IPs and $s(0, 1) = 0$ scalarized IPs. □

Example 5. *For $n = 1$ and $g = 0$, $\ell(1, 0) = 1$ and $s(1, 0) = 1$.*

Proof of Example 5. Since a single NDLS traverses Y, solving the first lexicographic minimization IP (5) with any split line $z_2 = \mu$ where $z_2^R < \mu < z_2^L$ will result in z^* on the

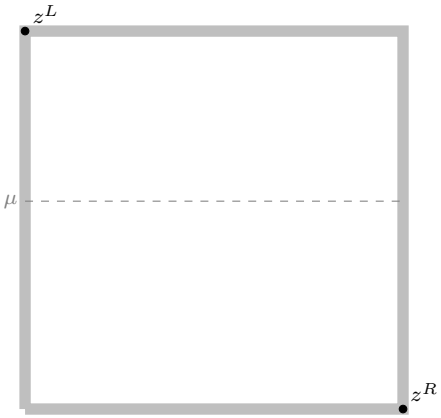


Figure 2.9: A region Y with $n = 0$ NDLSs in its interior and $g = 1$ vertical gaps. An arbitrary horizontal split line is shown as a dashed line.

split line, i.e., $z_2^* = \mu$. The line generation subroutine will generate the entire line segment within Y, and because there are no vertical gaps, we must have that $z_2^1 = z_2^L$ and $z_2^2 = z_2^R$. Since z^R is nondominated, the latter implies $z^2 = z^R$, and so z^2 is *closed*. However, z^1 may be open *or* closed; namely, if $z_1^L < z_1^1$, z^1 will be flagged as open. Regardless of whether it is open or closed, the inner loop will solve the single-objective IP (8) once to confirm that the line segment is indeed nondominated (recall that the IP formulation includes a conditionally strict inequality for the open endpoint which makes z^L infeasible). By our assumption that $n = 1$, an optimal solution must exist on the line segment. At the end of the inner loop, since $z_2^1 = z_2^L$, no other single-objective IPs will be solved by the inner loop since it is known that z^L is the ND image that dominates z^1. After the inner loop concludes, the outer loop would not add any box to the queue, since they would both be trivially small. Thus, $\ell(1, 0) = 1$ and $s(1, 0) = 1$. □

As we described while illustrating the basic method, every iteration of the outer loop identifies either a NDLS or a vertical gap. The following theorem proves that the algorithm eliminates that line segment or gap *completely* before adding two new boxes to the queue. This is an important characteristic which, for example, does not apply to the Triangle Splitting Algorithm since it may divide a single NDLS into several smaller segments

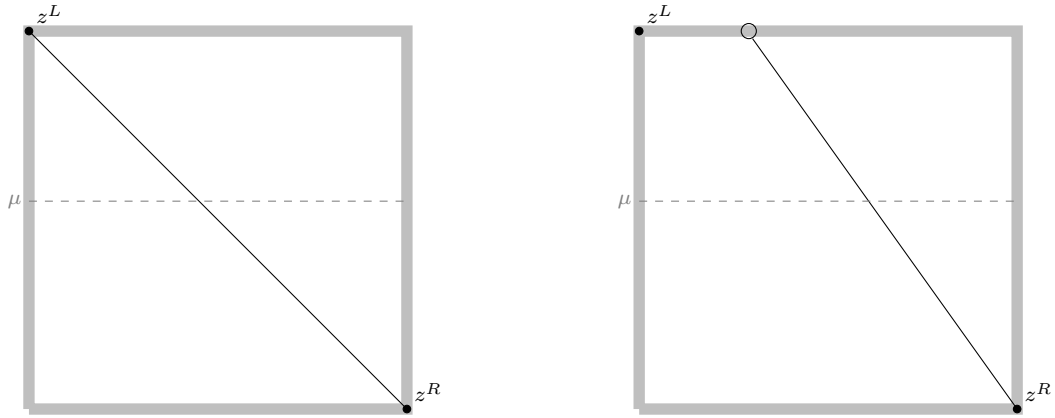


Figure 2.10: The only two cases for $n = 1$, $g = 0$: The NDLS must be incident to z^R to ensure no vertical gap, but the endpoint z^1 may be either open or closed. Both cases have the same worst-case: $\ell(1, 0) = 1$ and $s(1, 0) = 1$.

over more than one iteration. Proposition 1 proves this unique characteristic, which will allow us to provide an upper bound for the number of iterations of the outer loop as well as an appropriate equation for $\ell(n, g)$.

Proposition 1. *Suppose an iteration of the outer loop begins with a region Y having $n \geq 0$ NDLSs in its strict interior and $g \geq 0$ vertical gaps where $n + g \geq 1$.*

1. *At the end of the iteration, (up to) two nontrivial regions Y', Y'' are added to the queue, say with n', n'' numbers of NDLSs in their interior and g', g'' vertical gaps, respectively. Then we have $n' + n'' \leq n$ and $g' + g'' \leq g$ where **at least one** inequality is strict. That is, after each iteration of the outer loop, one NDLS or one vertical gap is eliminated completely from the total unexplored region(s).*

2. *The number of iterations of the outer loop required to finish completely processing Y (including all subsequent boxes) is bounded above by $n + g$.*

3. *$\ell(n, g) = n + 2g$.*

Proof of Proposition 1 (1). In one iteration of the outer loop, there are two cases for the split line $z_2 = \mu$: either it intersects a vertical gap or it intersects a NDLS of the ND frontier.

In the first case, when the split line intersects a vertical gap of the ND frontier, we have by construction that the two lexicographic minimizations (2.2) and (2.3) identify the two ND images incident to the vertical gap, $\hat{z}$ (above) and z^* (below). Let the two resulting regions be $Y' = B(z^L, \hat{z})$ and $Y'' = B(z^*, z^R)$. We have of course decreased the total number of vertical gaps in the ND frontier(s), so $g' + g'' < g$. However, the total number of NDLSs in the interiors of Y' and Y'' is no greater than n, i.e. $n' + n'' \leq n$. We have satisfied the claim in first case.

In the second case, z^* belongs to some NDLS $L(z^1, z^2)$. Then we initialize regions Y' and Y'' based on ND images – either the closed endpoints (i.e., z^1, z^2) or the ND images that dominate the open endpoints (i.e., $\hat{z}^1, \hat{z}^2$) – discovered by the inner loop in such a way that guarantees the single NDLS $L(z^1, z^2)$ is not split. Hence, we must have one fewer NDLS in the remaining regions, i.e. $n' + n'' < n$. We also have $g' + g'' \leq g$, which satisfies the claim in the second case. □

Proof of Proposition 1 (2). This result follows trivially from Proposition 1 (1). Since every iteration of the outer loop eliminates an NDLS or a vertical gap from the unexplored region(s), then after $n + g$ iterations, there will only be trivial unexplored regions remaining. □

Proof of Proposition 1 (3). This result follows from the proof of Proposition 1 (1). Only the outer loop solves lexicographic IPs. In every iteration of the outer loop, the horizontal split line $z_2 = \mu$ either intersects a vertical gap in the ND frontier or it intersects a NDLS. If the split line intersects a NDLS, one lexicographic IP is solved for z^*, and if the split line intersects a vertical gap, two lexicographic IPs are solved for z^* and $\hat{z}$. Then it takes at most $\ell(n, g) = n + 2g$ lexicographic IP solves to process Y (at most one for each NDLS and at most two for each vertical gaps). Also note that Examples 4 and 5 satisfy this upper bound. □

Lemma 1. *For all* $g \in \{0, 1, 2\}$, $s(1, g) = 1$.

Proof. Note that we have already shown $s(1,0) = 1$ by Example 5. Therefore, suppose $g = 1$ or $g = 2$.

First, consider when the split line $z_2 = \mu$ intersects the NDLS. Then solving one lexicographic IP (5) finds z^* such that $z_2^* = \mu$ (we do not count lexicographic IPs for s). Once the line segment containing z^* is generated, the single-objective IP (8) is solved to determine if the line segment is nondominated. As mentioned previously, the algorithm checks if $z_2^1 = z_2^L$ and $z_1^L < z_1^1$, in which case endpoint z^1 is flagged as open, and it is known that z^L dominates z^1 (and similarly for z^2 and z^R). Note that by assumption, $n = 1$ implies there are no other ND images in the interior of $B(z^1, z^2)$ that dominate the line segment, and by construction we have the conditionally strict inequality that prevents finding z^L (or z^R) in the case that z^L dominates z^1 (or z^R dominates z^2). Hence, an optimal solution to the single-objective IP (8) will certainly map to a point on the line segment, which will terminate the while loop within the inner loop. Since the ND images that dominate the open endpoints are known, no additional single-objective IPs will be solved to discover what points dominate z^1 or z^2, and so the inner loop terminates.

Once the inner loop has generated the NDLS within the given region, the outer loop will add at most two nontrivial boxes, whose interiors contain $n' = n'' = 0$ NDLSs, to the queue. So we have for some nonnegative integers $(g', g'') \in \mathbb{Z}_+^2$ where $g' + g'' = g$ but $g', g'' \leq 1$ (this follows from $g' \leq n' + 1$ and $g'' \leq n'' + 1$ where $n' = n'' = 0$). Therefore, $s(1, g) = 1 + s(0, g') + s(0, g'') = 1$ because $s(0, 0) = 0$ by definition, and $s(0, 1) = 0$ by Example 4. Thus, $s(1, g) = 1$ for $g \in \{0, 1, 2\}$.

Second, consider when the split line crosses at a vertical gap in the ND frontier. Note that a single-objective IP is not solved until the inner loop is initiated, which respectively does not occur until a ND image is found on the split line. Therefore there can be at most g iterations of the outer loop in which $z_2^* < \mu$, wherein each iteration is only performing lexicographic IP solves as the split line identifies a vertical gap in the ND frontier which is then removed by initiating new boxes to the queue (note we do not count any of these

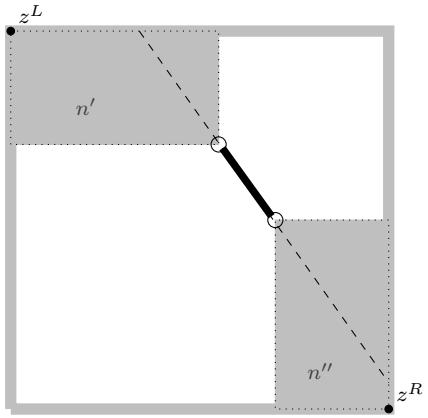


Figure 2.11: General ND frontier where there are $n \geq 1$ NDLSs. In the worst case for counting single-objective IP solves, both endpoints are open.

lexicographic IP solves). Therefore, in at most g iterations the lexicographic optimization IP will return z^* such that $z_2^* = \mu$. At such point, the process would follow the process argued above, so again we have $s(1, g) = 1$ for $g \in \{0, 1, 2\}$. □

Since $g \leq n + 1$, we can therefore define the worst case number of IP solves as a function of only n, the number of NDLS in the strict interior of Y:

$$\hat{\ell}(n) = \max_{g=1..n+1} \ell(n, g), \tag{2.13}$$

$$\hat{s}(n) = \max_{g=1..n+1} s(n, g). \tag{2.14}$$

We so far have $\hat{s}(0) = 0$, $\hat{s}(1) = 1$, and $\hat{\ell}(n) = 3n + 2$ for all $n \geq 0$ (since $g \leq n + 1$ gives $\ell(n, g) = n + 2g \leq n + 2(n + 1) = 3n + 2$).

Lemma 2. *For all* $n \geq 2$, $\hat{s}(n) = n + 2 + \hat{s}(n - 1)$.

Proof of LEMMA *2.* Assume there are $n \geq 2$ NDLSs. Without loss of generality, we suppose the number of vertical gaps in the ND frontier is arbitrary and only consider the case that the split line $z_2 = \mu$ intersects a NDLS; otherwise, when the split line intersects a vertical gap, the box is processed in the outer loop without solving any scalarized or single-objective IPs. Once the lexicographic IP (2.2) (which is not counted for $\hat{s}$) finds ND image

z^* on the split line, the inner loop is initiated. Since there are $n-1$ other NDLSs in the box, there are at worst $n-1$ scalarized IPs (2.5) solved in updating the endpoints z^1 and z^2. The nth solve of scalarized IP (2.5) confirms that the final endpoints on the line segment indeed define a NDLS. The worst case is when both z^1 and z^2 are open, $z_2^1 < z_2^L$, and $z_1^2 < z_1^R$ since two additional single-objective IP solves are required to find the ND images that dominate z^1 and z^2 (a figure illustrating this is given in the online supplement). Therefore, we have for some $n' + n'' = n-1$, $\hat{s}(n) = n+2+\hat{s}(n')+\hat{s}(n'')$. So it follows that the worst case is $\hat{s}(n) = n+2+\max_{i=0,..,n-1}\{\hat{s}(i)+\hat{s}(n-1-i)\}$.

We now use induction to complete the proof. First observe that for $n=2$, as required,

$$\hat{s}(n) = n+2+\max_{i=0,1}\{\hat{s}(i)+\hat{s}(2-1-i)\} = n+2+\hat{s}(1)+\hat{s}(0) = n+2+\hat{s}(1),$$

since $\hat{s}(0)=0$. Now make the inductive assumption that for some $m \geq 2$, $\hat{s}(n) = n+2+\hat{s}(n-1)$ for all $n = 2,\ldots,m$, and consider $\hat{s}(m+1) = m+3+\max_{i=0,..,m}\{\hat{s}(i)+\hat{s}(m-i)\}$. The case that $i=0$ (and $i=m$) in the max term gives $\hat{s}(0)+\hat{s}(m) = \hat{s}(m)$ since $\hat{s}(0)=0$. The case that $i=1$ (and $i=m-1$) in the max term gives $\hat{s}(1)+\hat{s}(m-1) = 1+\hat{s}(m-1)$ since $\hat{s}(1)=1$. But, by the inductive assumption $\hat{s}(m) = m+2+\hat{s}(m-1) > 1+\hat{s}(m-1)$, since $m \geq 2$. So the case $i=1$ (and $i=m-1$) cannot achieve the maximum. Finally, again by the inductive assumption,

$$\begin{aligned}
\max_{i=2,\ldots,m-2}\{\hat{s}(i)+\hat{s}(m-i)\} &= \max_{i=2,\ldots,m-2}\{i+2+\hat{s}(i-1)+m-i+2+\hat{s}(m-i-1)\} \\
&= m+4+\max_{i=1,\ldots,m-3}\{\hat{s}(i)+\hat{s}(m-2-i)\} \\
&\leq m+4+\max_{i=0,1,\ldots,m-3,m-2}\{\hat{s}(i)+\hat{s}(m-2-i)\} \\
&= m+4+\hat{s}(m-1)-(m-1+2) \\
&= 3+\hat{s}(m-1).
\end{aligned}$$

But $\hat{s}(m) = m+2+\hat{s}(m-1) > 3+\hat{s}(m-1)$ since $m \geq 2$, so none of the cases $i =$

$2, \ldots, m-2$ can achieve the maximum. We conclude that $\max_{i=0,..,m} \{\hat{s}(i) + \hat{s}(m-i)\} = \hat{s}(m)$ and so $\hat{s}(m+1) = m + 3 + \hat{s}(m)$, as required. □

Theorem 1. *For all* $n \geq 1$, $\hat{s}(n) = \frac{n(n+1)}{2} + 2(n-1)$.

Proof of Theorem 1. We use induction. The case $n = 1$ follows since $\hat{s}(1) = 1 = \frac{1(1+1)}{2} + 2(1-1)$. Now assume that, for some $n \geq 1$, $\hat{s}(n) = \frac{n(n+1)}{2} + 2(n-1)$, and consider $\hat{s}(n+1)$. By Lemma 2 and the inductive assumption,

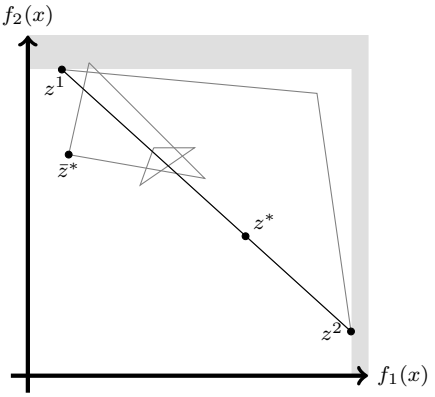
$$\begin{aligned}\hat{s}(n+1) = n + 3 + \hat{s}(n) = n + 3 + \frac{n(n+1)}{2} + 2(n-1) = \frac{1}{2}n^2 + \frac{7}{2}n + 1 \\ = \frac{1}{2}n^2 + \frac{3}{2}n + 1 + 2n = \frac{(n+1)(n+2)}{2} + 2n,\end{aligned}$$

as required. □

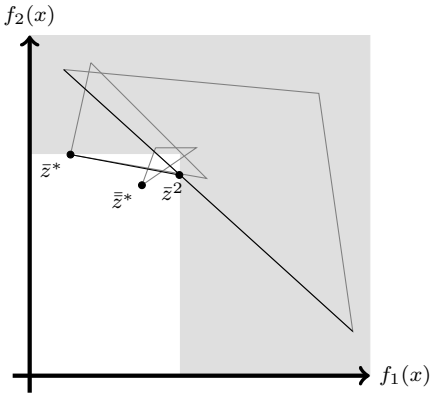
2.3 Recursive Method

There is a natural way to make the BLM a recursive method, which has substantially better worst-case complexity. The fundamental concept is to transform the inner loop into a recursive procedure that either confirms that a line segment is nondominated or identifies a ND image that dominates a portion of it. Whenever such a ND image is found, the inner loop is called recursively to find an NDLS containing it. In addition to modifying the inner loop, we must introduce a line segment *trimming* subroutine and modify the outer loop to fit the recursive paradigm. The recursive method is summarized in Figure 2.12.

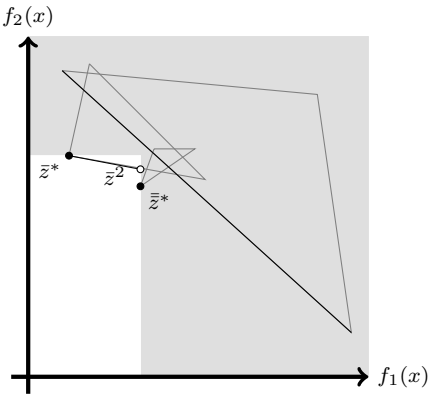
The initialization stage remains the same, and the outer loop follows the same procedure as the basic method before calling the inner loop, i.e. it selects boxes from the queue, chooses a split line $z_2 = \mu$, solves lexicographic minimization (2.2), and follows the BBM procedure when the ND image found is below the split line. When the outer loop has chosen box $B(z^L, z^R)$ from the queue and finds ND image z^* on the split line, i.e. $z_2^* = \mu$, the outer loop then calls the recursive inner loop for the first time, which we call *depth level 0*.



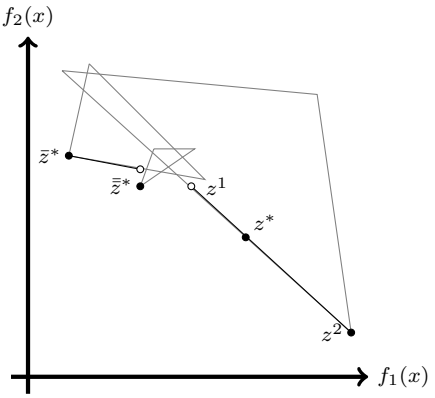
(a) Level 0: The original ND image is z^*, and line generation returns $L_0 = L(z^1, z^2)$, which is not trimmed. Solving a scalarized IP over the white region identifies ND image $\bar{z}^*$.



(b) Level 1: The line generated for $\bar{z}^*$ is trimmed by its intersection with L_0, resulting in $L_1 = L(\bar{z}^*, \bar{z}^2)$ with $\bar{z}^2$ closed. ND image $\bar{\bar{z}}^*$ solves the next scalarized IP.
Level 2: Returns isolated ND image $\bar{\bar{z}}^*$.



(c) Level 1: Update endpoint $\bar{z}^2$, which is now open. One more scalarization over the white region identifies that $L(\bar{z}^*, \bar{z}^2)$ is nondominated.



(d) Level 0: Update endpoint z^1 with respect to $p = \bar{\bar{z}}^*$, so z^1 is now open. Solving one final scalarized IP confirms that $L(z^1, z^2)$ is nondominated.

Figure 2.12: The recursive inner loop applied to ND image $\bar{z}^*$ returns NDLSs $L(z^1, z^2)$ and $L(\bar{z}^*, \bar{z}^2)$ and the isolated ND image $\bar{\bar{z}}^*$.

2.3.1 Recursive Inner Loop

The recursive inner loop takes as input: a box bounded by two ND images, $B(z^L, z^R)$; the ND image z^* whose NDLS it seeks to generate; and the *set* of all line segments "inherited" from its parent calls, which we call $\mathcal{L}$. On the first call of the inner loop (depth level 0), the set $\mathcal{L}$ is empty; for greater depth levels, this set will be nonempty and will be used for

the line segment trimming subroutine. The output from the recursive function is a set of NDLSs, $\mathcal{M}$, including the one that contains z^*.

We now describe the steps of the recursive function. It first calls the line segment generation subroutine, as in the basic method. If z^* is an isolated ND image, then the recursive function terminates and returns the isolated ND image. Otherwise, the subroutine provides the line segment $L(z^1, z^2)$ and its gradient vector $\vec{w}$.

The second step is to call the line segment trimming subroutine, which "trims" the current line segment so that its endpoints do not exceed any of the previously-found line segments. This step prevents cycling. We postpone further discussion of this feature to Section 2.3.2. For now, we note that there is no trimming to be done at depth level 0 since $L(z^1, z^2)$ is the first line segment that has been generated in the box $B(z^L, z^R)$. Once trimming is complete, the same checks as in the basic method's inner loop are made, i.e., to see if either endpoint is dominated by z^L or z^R (and, if so, to update the endpoint as open, accordingly). The resulting line segment is then added to the set of line segments, $\mathcal{L}$, for future line trimming.

The next step in the recursive inner loop is to solve the same scalarized IP with respect to $\vec{w}$ as in the basic method, i.e., (2.5), including the conditionally strict inequalities. Say the optimal solution is y^*. There are two cases. (1) If $L(z^1, z^2)$ is nondominated, then $f(y^*)$ lies on the line segment, i.e. $\vec{w}^T f(y^*) = \vec{w}^T z^*$. This is the stopping criterion for the while loop. When it is satisfied, the recursive function terminates and returns the NDLS, together with all others it found, stored in $\mathcal{M}$. (2) The more interesting case is when $\vec{w}^T f(y^*) < \vec{w}^T z^*$, i.e., when a point from $L(z^1, z^2)$ is *dominated* by $f(y^*)$. In this case, the function recurses and calls itself, at the next depth level. Assume $f(y^*)$ is to the left of z^*; the case of $f(y^*)$ to the right can be handled similarly. Then the following is the input to the recursive call:

- $B(\chi, z^*)$, the new box, where χ is the ND image from $\mathcal{M} \cup \{z^L\}$ closest to $f(y^*)$ on the left (note that z^* is the closest known ND image to the right of $f(y^*)$);

- $f(y^*)$, the new ND image (it is the NDLS containing it that the function seeks); and
- $\mathcal{L}$, the set of line segments, used for line segment trimming.

The choice of $B(\chi, z^*)$ as the new box prevents re-discovery of line segments already found, and ensures that only the portion of the line segment containing $f(y^*)$ currently not known to be dominated is explored in the recursive call. Note that when $\mathcal{M}$ is empty, $\chi = z^L$.

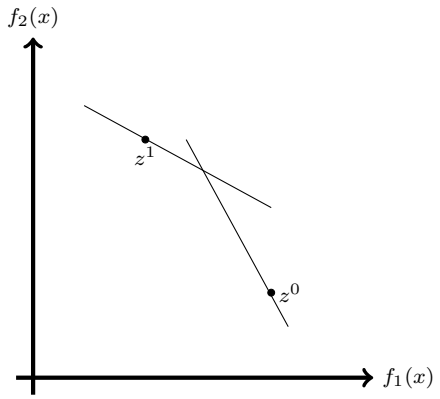
After all deeper levels of recursive processing are complete, a set of NDLSs, $\mathcal{M}$, is returned to the current depth level call of the recursive inner loop. Then the recursive function updates the line segment $L(z^1, z^2)$. In doing so, it follows the same rules as in the basic method's inner loop. However, instead of v found from the LP (2.6), it chooses the ND image from $\mathcal{M}$ that is the nearest to and strictly to the left of z^*. That is, it chooses

$$\sigma = \text{argmin}\{z_2 : z \in \mathcal{M}, z_1 < z_1^*\}. \tag{2.15}$$

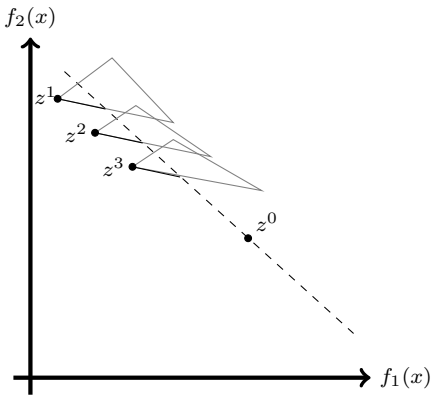
The coordinate σ_2 gives the updated z_2^1; the updated z_1^1 value is calculated to ensure z^1 remains on its original line segment. Whether it is closed or open is determined by whether or not σ lies on the line segment. Since σ is guaranteed to be a ND image, there is no need to solve a separate LP to discover the ND image that dominates z^1 when it is open.

Once the endpoint z^1 (or z^2, if exploring to the right of z^*) is updated, the while loop again solves the scalarized IP (2.5). If the updated line segment is nondominated, then the stopping criterion is met and the line segment $L(z^1, z^2)$ is returned, along with any others collected in $\mathcal{M}$. Otherwise, the while loop continues.

The recursive inner loop accumulates all NDLSs identified throughout all depths of recursion. This accumulated set, $\mathcal{M}$, is returned to the outer loop, which updates $\tilde{\mathcal{N}}$ and $\mathcal{Q}$ accordingly (details are in Section 2.3.3).



(a) Without the line segment trimming, the recursive method would cycle between finding z^0 and z^1.



(b) The recursive method first finds ND image z^0 and its line segment (dashed). By recursing, it finds z^1, z^2, and z^3 in that order, where the line segments after trimming are darkened.

Figure 2.13: The line segment trimming subroutine prevents the recursive method from cycling.

2.3.2 Line Segment Trimming Subroutine

To motivate this subroutine, we give an example of how – without it – cycling can occur. Consider two intersecting slices from distinct integer solutions, as in Figure 2.13a. Suppose z^0 were first found by the split line and lexicographic minimization. Then line generation would generate its full line segment, L_0, and scalarization by its gradient $\vec{w}^0$ would result in finding z^1 (recall that the scalarized IP (2.5) is constrained by its endpoints). For the next level of recursion, line generation would generate the full line segment containing z^1, L_1, and its gradient $\vec{w}^1$. Note, however, that L_1 intersects L_0, and in fact the lower right endpoint of L_1 is dominated by z^0. Thus scalarization by $\vec{w}^1$ would yield z^0 again, and the algorithm would cycle.

In general, there is a risk of cycling whenever a call to the recursive inner loop generates a line segment that intersects the line segment of a parent call. We developed a simple routine to detect such intersections and update the child's line segment so as to prevent cycling. This is done by solving a linear system and updating one endpoint of the child's line segment, as necessary; see Appendix Section A.1 for details.

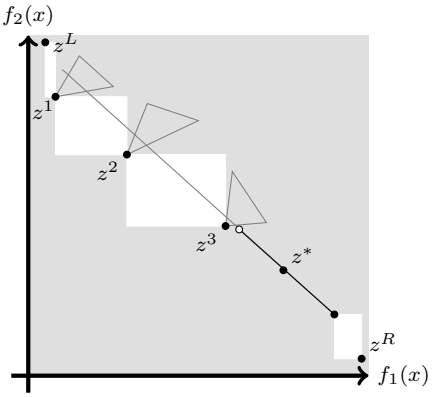


Figure 2.14: The recursive variant may return more than one continuous portion of the ND frontier (e.g., the NDLS containing z^* and 3 isolated ND images in the example above), in which case the algorithm adds more than two boxes to the queue (in white).

The line trimming process must be repeated for all line segments in $\mathcal{L}$ that have been inherited by the current call. For instance, observe that in Figure 2.13b, if all subsequent line segments were not trimmed by the line segment containing z^0 then a cycle would occur.

2.3.3 Outer Loop Modification

The only major modification in the outer loop is the handling of the output from the recursive inner loop, which may return more than one continuous portion of the ND frontier (see Figure 2.14). Therefore, we must allow the outer loop to add more than one NDLS to the ND frontier and more than two boxes to the queue, as necessary.

The latter requires a simple check of neighboring NDLSs. Consider checking $L_1 := (z^1, z^2)$ which is immediately to the left of $L_2 := (z^3, z^4)$. When $z_1^2 < z_1^3$ and $z_2^2 > z_2^3$, the outer loop should add the box $B(z^2, z^3)$ to the queue. Note that we must also consider the boundary ND images, z^L and z^R, while doing this check. In Figure 2.14, four boxes are added to the queue because four pairs of adjacent NDLSs satisfy the criteria. One pair of NDLSs does not satisfy the criteria: the isolated ND image z^3 and the NDLS containing z^*.

2.3.4 Complexity of the Recursive Method

Let $\hat{\ell}_R(n)$ be the worst case number of lexicographic IPs solved by the *recursive* method in completely processing an arbitrary box $B(z^L, z^R)$ with n NDLSs (and an arbitrary number of vertical gaps), and let $\hat{s}_R(n)$ be the same but for scalarized IPs. Recall that the outer loop only solves lexicographic IPs, and the recursive inner loop only solves scalarized IPs.

The initialization stage and most of the functionality of the outer loop is unchanged, i.e., in every iteration the outer loop solves one lexicographic IP to identify a NDLS, or it solves two lexicographic IPs to identify a vertical gap. What has changed is the updating procedure for the ND frontier and queue after the recursive inner loop has terminated. Regardless, the upper bound for the basic method's number of lexicographic IP solves is valid for the recursive method, i.e. $\hat{\ell}_R(n) = \hat{\ell}(n) = 3n + 2$. However, we must reconsider an upper bound for the number of scalarized IP solves.

Proposition 2. *For all* $n \geq 1$, $\hat{s}_R(n) = 2n - 1$.

Proof of Proposition 2. Let z^L, z^R be the corner points for a box with $n \geq 1$ NDLSs. Assume that the ND frontier in the box contains no vertical gaps. This assumption can be made without loss of generality, since vertical gaps in the ND frontier are only discovered in the outer loop, by a lexicographic IP solve that yields a ND image not on the split line; they do not change the number of scalarized IPs that need to be solved.

Since there are no vertical gaps, any split line intersects an NDLS, containing the ND image z^* found by the lexicographic IP. Then the line segment in a slice containing z^* is determined (using LPs) and the recursive inner loop is initiated.

In the recursive inner loop, there are two roles for scalarized IPs. Type I confirms that the current line segment is nondominated, and every nontrivial line segment requires exactly one of these IPs to be solved. Isolated ND images require none, so in the worst case, all NDLSs have dimension 1, and one Type I scarized IP is solved per NDLS in the ND frontier. Type II identifies an NDLS for the first time: it shows that a portion of the

current line segment is dominated, by returning a new ND image, and the recursive call returns the NDLS containing the new ND image. Within the recursive call, it is impossible to rediscover the same NDLS. Therefore, no more than $n-1$ Type II scalarized IPs will be required to discover all of the NDLSs (the first NDLS must be found by a lexicographic IP). Since all n NDLSs require solving a Type I scalarized IP to confirm nondominance, a total of $\hat{s}_R(n) = n + (n-1) = 2n-1$ scalarized IPs will be solved, in the worst case. □

2.4 Same Integer Solution Enhancement

We propose an enhancement to the basic method (given in Section 2.2) that has the potential to improve its computational performance. The enhancement exploits the observation that when both corner points of a box have the same integer part of their solution, (they belong to the same slice), there is a good chance that the ND frontier within the box is precisely the part of the slice within the box. We call this the *Same Integer Solution (SIS)* variant.

This enhancement was motivated by a common structure encountered in the ND frontiers of benchmark instances used in previous work on BOMIP algorithms [31, 26]. These instances have ND frontiers consisting of a relatively small number of continuous sections, generated by relatively few slices, each having many small NDLSs. Figure 2.15 gives plots of two such instances. In one, only 5 slices contribute to the ND frontier, which consists of only 4 continuous sections. In the other, 15 slices yield the ND frontier consisting of 10 continuous sections. Each continuous section has several NDLSs from the same slice. These two instances are typical of the benchmark instances used in prior work.

During the BLM's processing of these instances, when both corner points of a box have the same integer solution, i.e., when $z^L = f(x^L)$ and $z^R = f(x^R)$ where $x^L_I = x^R_I =: x^*_I$, it is very likely that the ND frontier between z^L and z^R is a subset of the slice for x^*_I. The SIS variant exploits this observation. In the case that the slice problem does yield the ND frontier in the current box, the SIS variant may need to solve only one scalarized IP, having one no-good (sometimes called "tabu") constraint, instead of one scalarized IP for each

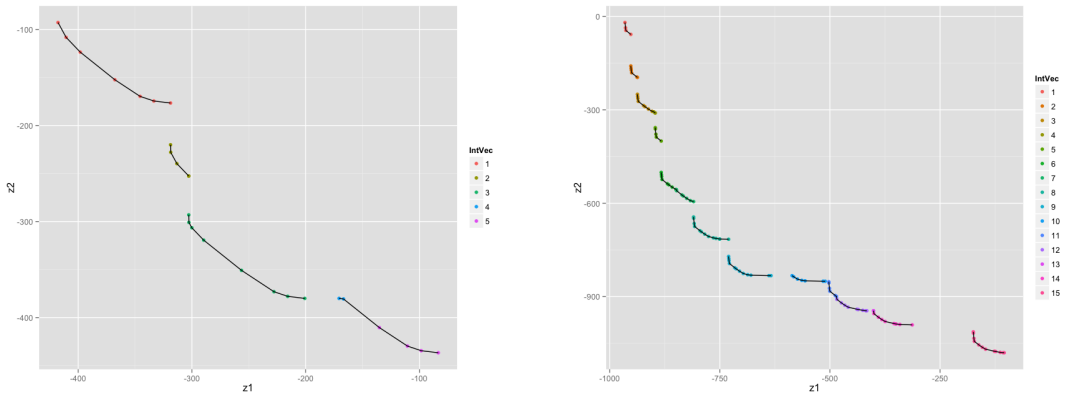


Figure 2.15: The ND frontiers for two benchmark instances. The keys indicate the integer vector associated with each ND image. Notice the similar structure in both: each integer vector contributes several small and continuous NDLSs to the ND frontier.

NDLS; once it is determined that the slice problem yields the ND frontier, LPs, rather than IPs, may be used to find all NDLSs.

When the corner points of a box are generated by a single integer solution, x_I^*, the slice must be entirely on or below the line segment $L(z^L, z^R)$. This follows from convexity of the image of the LP feasible set (integer part of the solution is fixed) in criterion space.

Let $\vec{w}$ denote the gradient vector of $L(z^L, z^R)$. The following scalarized IP with respect to $\vec{w}$ includes a no-good constraint, $x_I \neq x_I^*$, which makes all points from the slice for x_I^* infeasible:

$$\min\{\vec{w}^T f(x) : f_1(x) \leq z_1^R, \;\; f_2(x) \leq z_2^L, \;\; x_I \neq x_I^*, \;\; x \in \mathcal{X}\}. \tag{2.16}$$

We assume that the no-good constraint is implemented linearly in the usual way. Let y^* be an optimal solution to (2.16). Because of the no-good constraint, $f(y^*)$ is not necessarily nondominated. However, we make a simple observation: if the point $f(y^*)$ is on or above the line segment $L(z^L, z^R)$, i.e. $\vec{w}^T z^L \leq \vec{w}^T f(y^*)$, then $f(y^*)$ must be dominated by a point from the slice for x_I^*, as shown in Figure 2.16a. Furthermore, the entire ND frontier within $B(z^L, z^R)$ is given by the slice for x_I^*. In this case, we solve the slice problem for

x_I^* using dichotomic search [21, 22], which solves a series of LPs[4]. Therefore, if $f(y^*)$ is on or above the line segment $L(z^L, z^R)$, the SIS variant will generate the entire ND frontier within $B(z^L, z^R)$ by solving just one scalarized IP (with one no-good constraint) and a sequence of LPs. Contrast this with the basic BLM: in the same scenario with n such NDLSs in the ND frontier (all from the same slice), it would solve n lexicographic IPs and n scalarized IPs. Thus the SIS variant has the potential to provide significant savings in the number of IPs solved.

Now consider the other case, that $f(y^*)$ is below the line segment $L(z^L, z^R)$, i.e., that $\vec{w}^T f(y^*) < \vec{w}^T z^L$. There are two sub-cases: either $f(y^*)$ is a ND image, or $f(y^*)$ is dominated by some point on the (so far unknown) slice for x_I^*. These subcases are illustrated in Figures 2.16b and 2.16c, respectively.

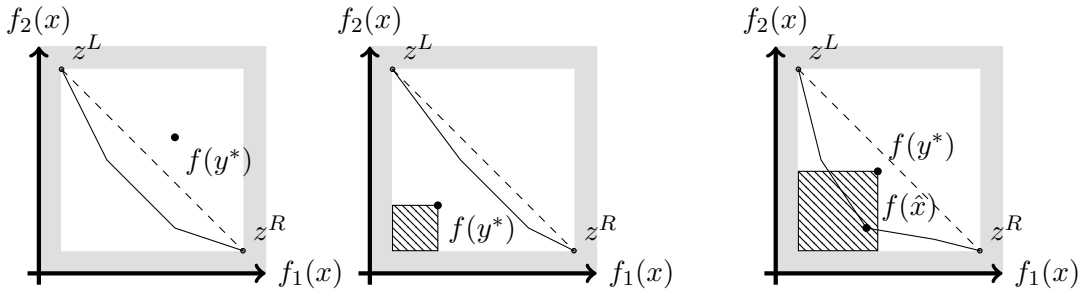
When $f(y^*)$ is below $L(z^L, z^R)$, the SIS variant solves the LP

$$\min\{\vec{w}^T f(x) : f_1(x) \leq f_1(y^*),\ \ f_2(x) \leq f_2(y^*),\ \ x_I = x_I^*,\ \ x \in \mathcal{X}\}. \tag{2.17}$$

If (2.17) is infeasible, then $f(y^*)$ is a ND image. This follows since $z_i(x) \leq z_i(y^*)$ for $i = 1, 2$ and $x \in \mathcal{X}$ implies, by the definition of y^*, that either $f(x) = f(y^*)$ or $x_I = x_I^*$. The latter would imply x_I is feasible for (2.17), which is impossible. Hence $f(x) = f(y^*)$; $f(y^*)$ must be a ND image. Otherwise, for similar reasons, any feasible solution of (2.17), $\hat{x}$ say, generates a ND image $f(\hat{x})$. In either case, a new ND image has been found, and then the inner loop is called with the box $B(z^L, z^R)$ and this new ND image as z^*.

In this case, the SIS variant solves one scalarized IP with a no-good constraint and one LP before calling the inner loop, where the output will be a single NDLS (and some number of boxes added to the queue). On the other hand, the basic BLM would solve one lexicographic IP before calling the inner loop, with similar output. Therefore, the computational effort (and return) is comparable, and so there is not much benefit from the SIS variant in this case.

[4] An alternative method for solving the slice problem is parametric simplex [17].



(a) If $f(y^*)$ is found on or above $L(z^L, z^R)$ (dashed), then the ND frontier is generated entirely by x_I^*. The slice (solid) is determined by solving LPs.

(b) If $f(y^*)$ is found below $L(z^L, z^R)$, an LP over the hatched region is solved. Here it determines that the slice for x_I^* does *not* dominate $f(y^*)$, and the inner loop is called with ND image $z^* = f(y^*)$.

(c) In this case, the LP over the hatched region finds the point $f(\hat{x})$ on the slice for x_I^*, which dominates $f(y^*)$, so the inner loop is called with ND image $z^* = f(\hat{x})$.

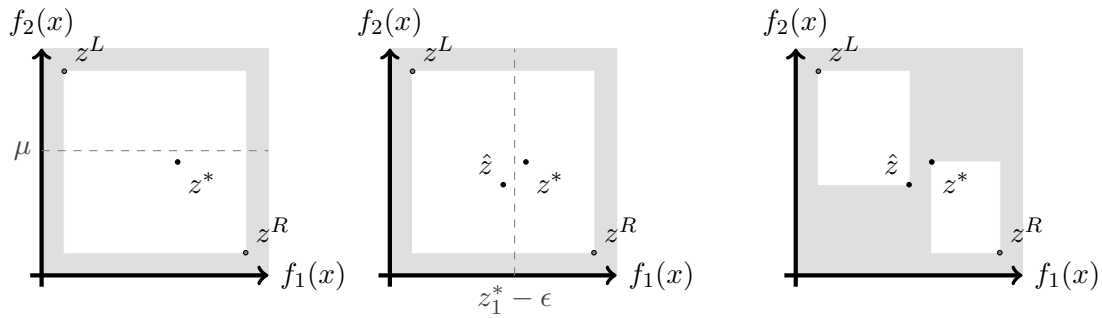
Figure 2.16: The SIS variant applied when z^L and z^R are both generated by integer solution x_I^*. The scalarization (2.16) with no-good constraint $x_I \neq x_I^*$ will find a point $f(y^*)$ either above or below the line segment $L(z^L, z^R)$, proceeding with one of the three cases illustrated in (a), (b) and (c) above.

Computational experiments with the SIS variant show enough improvement to indicate that the first case is much more likely than the second case, and so, on balance, the enhancement is useful.

2.5 Implementation Issues

Because computers use finite precision arithmetic, it is necessary to introduce numerical tolerances. We use a value $\epsilon > 0$ to indicate the accuracy at which we expect the BLM to provide output (defined more precisely in Section 2.5.1). Note that because the choice of ϵ impacts (the accuracy of) the ND frontier generated by an algorithm, it also impacts the time required by the algorithm to find the ND frontier. In Section 2.5.2, we show that the ND frontiers generated for the same instance for different values of ϵ can vary drastically.

Single-objective IP solvers use tolerances as well, e.g., feasibility and optimality tolerances, and their choice also impacts the performance. In fact, certain choices of IP solver tolerances may lead to unexpected behavior in the algorithm. An illustration is provided



(a) In $B(z^L, z^R)$, the first lexicographic IP finds ND image z^* below the split line $z_2 = \mu$.

(b) In $B(z^L, z^R)$, the second lexicographic IP finds ND image $\hat{z}$ with $\hat{z}_1 \leq z_1^*$, but also with $\hat{z}_2 \leq z_2^* < \mu$, i.e., $\hat{z}$ dominates z^*.

(c) The two boxes in white would be added to the queue. However, a cornerpoint being dominated violates one of the most basic assumptions of our algorithms.

Figure 2.17: If IP solver tolerances are not set appropriately, i.e., strictly less than the algorithm's ϵ, then IPs (especially lexicographic IPs) may return points that are not non-dominated.

in Figure 2.17. This is just one example of the numerical issues that we encountered during the implementation of our proposed algorithms. We found that it is critical that the IP solver tolerances are set to values strictly smaller than ϵ.

The black box nature of (commercial) IP solvers also makes it difficult to intuit how redundant criterion space constraints impact its performance. For instance, each box, $B(z^1, z^2)$, can be represented by four constraints, i.e., $z_1^1 \leq f_1(x) \leq z_1^2$ and $z_2^2 \leq f_2(x) \leq z_2^1$, or by just two constraints, i.e., $f_1(x) \leq z_1^2$ and $f_2(x) \leq z_2^1$ (the fact that z^1 and z^2 are ND implies the remaining constraints). Our computational experiments with CPLEX indicated that the former performs better. However, this behavior may be different with other solvers.

2.5.1 Epsilon Frontier

Recognizing that computers use finite precision arithmetic, we define an *approximation of the ND frontier* $\mathcal{N}$, or simply an *ϵ-frontier*, $\mathcal{N}_\epsilon$, to be a set of points in criteria space such that, for fixed $\epsilon > 0$, (1) for all $z \in \mathcal{N}$, there exists $\bar{z} \in \mathcal{N}_\epsilon$ with $||z - \bar{z}||_2 \leq \epsilon$, and (2) for

all $\bar{z} \in \mathcal{N}_\epsilon$, there exists $z \in \mathcal{N}$ with $||z - \bar{z}||_2 \leq \epsilon$.[5] Note that there may be many distinct sets that satisfy the conditions for being an ϵ-frontier. The BLM (when run to completion) produces an ϵ-frontier for any given BOMIP having nonempty and bounded feasible set. To see how ϵ is used in the method to produce the ϵ-frontier, see the algorithms given in the Appendix.

This definition of an ϵ-frontier motivated specific design choices in our implementation. For example, when the inner loop solves scalarization IPs, e.g. (2.5), to find ND images that dominate a line segment, our definition implies that we are only interested in ND images whose distance from the line segment are greater than ϵ. Therefore, we designed the criterion for entering the inner loop's while loop to be $\vec{w}^T f(y^*) < \vec{w}^T z^* - \epsilon||\vec{w}^T||_2$ where y^* is the optimal solution to the scalarized IP. By normalizing the gradient vector $\vec{w}$ with respect to the 2-norm at the end of the line generation subroutine, i.e., by requiring $||\vec{w}^T||_2 = 1$, we can simplify the criterion to $\vec{w}^T f(y^*) < \vec{w}^T z^* - \epsilon$.

2.5.2 Epsilon Sensitivity of Historical Instances

The benchmark instances used in previous computational studies on algorithms for BOMIPs [31, 26] are based on the scheme proposed by [12]. The ND frontiers for the 20 instances, ranging in size from 20 to 320 decision variables, have a structure that was described in Section 2.4: much of the ND frontier consists of continuous line segments from the same slice (Figure 2.15). This structure not only poses numerical challenges when the line segments are extremely small, i.e., when their lengths are smaller than ϵ, it also means that changes in the value of ϵ result in drastically different ND frontiers being generated by our algorithms, as shown in Table 2.1.

The fact that the ND frontier output by an algorithm for solving BOMIPs can be considerably different when a different tolerance value is used makes it very difficult to compare the performance of such algorithms. The issue is compounded by the fact that an algo-

[5] The concept of an ϵ-frontier is not new; however, as opposed to the definitions in papers such as [54] and [55], which rely on the notion of relative error, our definition uses the notion of absolute error.

Table 2.1: Number of NDLSs found for different values ϵ. All five instances are from the C160 class of instances and are solved by the basic method.

Instance	10^{-3}	10^{-4}	10^{-5}	10^{-6}
1	2570	2752	2783	2786
2	2643	2916	2958	2970
3	2578	2723	2750	2753
4	5608	6121	6164	6174
5	2824	3054	3096	3096

rithm, such as TSA, for example, may output several small line segments whose union is equivalent to a single line segment output by another algorithm, such as εTCM. This has motivated us to generate new instances for which the exact ND frontier is known and the line segments in the frontier all have length greater than a pre-specified value, e.g., 10^{-4}.

2.6 Instance Generation

We now provide a method for generating a BOMIP having a ND frontier controlled by parameters and known, *a priori*. In doing so, we provide an approach to "reverse engineer" a BOMIP from the slices[6].

Each instance's ND frontier includes some sections of the line segment $L_k = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 + x_2 = 0, -k \leq x_i \leq k \text{ for } i = 1, 2\}$ where $k \in (0, \infty)$ is a parameter. This line segment will be one slice in the instance. The instance has π other slices, where π is a parameter, all of which have their image in criterion space given by a pointed cone. The vertices of each cone lie on a line segment parallel to L_k but shifted vertically down. The width of the cone is randomly chosen in a controlled manner. Figure A.1 in the Appendix illustrates this structure, showing the image of the feasible set in criterion space for 4 slices: L_k and three pointed cones. All instances are constructed to have the property that no two cones overlap on or below L_k. This means that L_k alternates between a section that is part of the ND frontier and a section dominated by one cone.

Details of a method to generate two different classes of instances having the structure

[6]The generated instances are publicly available at https://github.com/t-perini/BOMIPresearch.

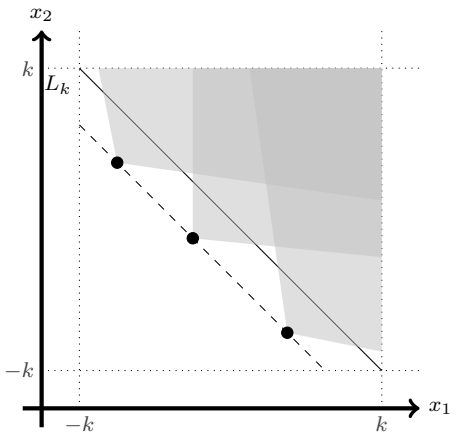


Figure 2.18: An example of a generated randomized cone-width instance with four slices, including L_k and the boundary of three cones.

described above are provided in the Appendix. One class, the *fixed cone-width instances*, is preferred for assessment of the *accuracy* of BOMILP algorithms because the ND frontier is known exactly, *a priori*. The other class, the *randomized cone-width instances*, do not have known ND frontiers, and this class of instances is more difficult to solve, in practice, because of the high frequency of intersecting slices in the ND frontier. All instances are carefully designed to ensure that each NDLS has length at least ϵ. We also force a proportion of the cones in a randomized cone-width instance to be orthogonal[7]; this induces open endpoints.

Given π, the resulting BOMIP has a number of variables and a number of constraints that is linear in π. The ND frontiers associated with the BOMIP will have no more than $3\pi + 1$ line segments ($\pi + 1$ from L_k and at most 2 per cone), including open endpoints induced by any orthogonal cones and closed endpoints induced by intersecting slices.

These structured sets of instances provide a useful way to study the accuracy and robustness of a BOMIP algorithm.

[7] A cone is chosen to be orthogonal with probability ϕ, a parameter. In all our instances, we used $\phi = 0.05$.

2.7 Computational Study

This section provides computational results obtained by the BLM (BLM) on the following sets of instances:

- the ten largest instances from the benchmark instances for BOMIPs proposed by [12], which we refer to as the *historical instances*, and
- ten new randomized cone-width instances, five obtained by setting $\pi = 5000$ and five obtained by setting $\pi = 7500$, which we refer to as the *new instances*.

All variations of BLM are coded in C++ and solve the linear and integer programs using IBM CPLEX Optimizer 12.6. All experiments were conducted in a single thread of a dedicated Intel Xeon ES-2630 2.3GHz with 50GB RAM, running Red Hat Enterprise Linux Server 7.4.

All variations of BLM use tolerances $\epsilon = 10^{-5}$ and CPLEX tolerances 10^{-7} for historic instances and tolerances $\epsilon = 10^{-4}$ and CPLEX tolerances 10^{-6} for the new instances. The reason for using greater accuracy for the historical instances is that many of the ND frontiers have very small line segments as well as nearly horizontal or vertical line segments, which makes the accurate calculation of the gradients critical.

We structure the discussion of the computational experiments as follows. In Section 2.7.1 we present a comparison of the variants of BLM, i.e., basic, SIS, and recursive, and in Section 2.7.2, we compare (variants of) BLM with TSA and εTCM.

2.7.1 Comparison between BLM variants

Table 2.2 shows the results for the largest historical instances, i.e., the class C320 instances (results for the next largest instances, i.e., the class C160 instances, can be found in the online supplement). Table 2.3 summarizes the results for the new instances by presenting the mean of the performance metrics (individual results can be found in the online supplement). For each combination of algorithm variant and instance, we provide the following

statistics: **nNDP**, the number of points output by the algorithm (described in more detail below); **nIPF**, the number of different integer part solutions (i.e., the number of different slices appearing in the ND frontier); **TT**, the total time in seconds to discover the ND frontier; **IPT**, the total time in seconds spent in IP solves; **LPT**, the total time in seconds spent in LP solves; **nIP**, the total number of IPs solved; **nLex**, the total number of lexicographic IPs; **nMin**, the total number of single-objective IPs solved to find the ND image that dominates an open endpoint, i.e. (2.8) and (2.9); **nScal**, the total number of scalarized IPs, e.g., (2.5); **nGood**, the total number of IPs with the no-good constraint; **nLP**, the total number of LPs solved; **nRec**, the total number of boxes processed; **nSIS**, the total number of boxes processed where the corner points are generated by the same integer solution; and, finally, **nZL**, the number of boxes with z^* on the horizontal split line. When not applicable, an entry in the tables is marked with "-".

Note that when we report the number of ND images (**nNDP**), we report the number of *distinct endpoints of line segments* in the ND frontier. (Note that each non-degenerate line segment in the ND frontier, regardless of whether it is closed, half-open, or open, results in two points being output by the algorithm, and that each isolated ND image, results in one point being output by the algorithm.) We use **nNDP** to denote the number of points output by the algorithm, and in the remainder of this section, refer to these points as ND images, even though in some cases, e.g., the open endpoints of line segments, they are not.

The value reported in the column **nNDP** for a given instance should ideally be the same regardless of the algorithm used. However, because of the different numerical issues that a particular algorithm may encounter, they rarely produce exactly the same number, at least for the historical instances.

Note too that the total number of IPs solved (**nIP**) satisfies $nIP = 2 \times nLex + nMin + nScal + nGood$.

First, we discuss the results from Table 2.2, starting with the basic variant. As shown in Figure 2.15, the ND frontier of a historical instance is characterized by continuous portions

of many small NDLSs from the same slice. In fact, the basic variant of BLM reports an average ratio of 47.5 ND images per integer solution (i.e., **nNDP/nIPF**). The total time to solve a historical instance reported for the basic variant is approximately the sum of the time for solving IPs and LPs (i.e., **TT** $\approx$ **IPT + LPT**). The average percentage of the total time spent solving IPs is 88.2% We note that the number of lexicographic IPs is quite close to the number of ND images. This is to be expected since in the basic variant of BLM each line segment of the ND frontier is discovered by first solving a lexicographic IP to obtain a ND image on this line segment. The number of scalarized IPs does not differ much from the number of ND images as the majority of the line segments produced by the line generation subroutine can usually be proven to be part of the ND frontier by solving a single scalarized IP; the line segments produced by the line generation subroutine are proven to be nondominated 98.1% of the time (on average). The number of boxes processed is also quite close to the number of ND images as each line segment of the ND frontier is discovered by processing exactly one box. The number of boxes with corner points with the same integer solution (**nSIS**) is large: 87.8% of the boxes (on average) Finally, note that the vast majority of the ND images found by solving a lexicographic IP in the outer loop fall on the horizontal split line: 99.9% of the ND images (on average). The high likelihood that the split line intersects some line segment of the ND frontier is a result of the fact that these instances have ND frontiers that contain few vertical gaps.

The SIS variant, as expected, greatly improves upon the basic variant. Given a box with corner points having the same integer solution, the SIS variant is able to prove that the slice for the integer solution inside the box is part of the ND frontier by solving a single IP with a no-good constraint most of the time. In that case, the ND frontier inside the box is generated by solving LPs only. The improvement of the SIS variant over the basic variant is 81.2% for total time (on average) and 85.6% for number of IP solved (on average). This is the best variant of BLM for the historical instances.

We want to draw attention to the fact that the SIS variant produces fewer ND images

for the historical instances than the other variants. We have observed that this is due to inaccuracies that can occur in the calculation of the gradients of a line segment in the line generation subroutine. If the calculated gradient for a line segment is not very accurate, then the LPs solved to find the endpoints z_1 and z_2 of the line segment can find incorrect endpoints. As a consequence, rather than finding the entire line segment, only a portion may be found, effectively splitting the line segment into smaller line segments. The SIS variant relies less on the line generation subroutine. When the SIS variant establishes that the ND frontier inside the box belongs to a single slice, the ND images of that slice problem are found by solving LPs without the need to calculate the individual gradient of each line segment.

The recursive variant does not improve upon the basic variant because it does not recurse very often (the maximum recorded depth was 3 for all instances), because most of the time the full line segment obtained from the line generation subroutine is part of the ND frontier. Since the recursive variant rarely recurses, it proceeds similarly to the basic variant for these historical instances.

Next, we focus on the results for the new instances. The basic variant reports an average ratio of 3 ND images per integer solution for both sets ($n = 5000$ and $n = 7500$). This is expected due to their structure. The percentage of time spent solving IPs is 44.9% for set $n = 5000$ and 46.3% for set $n = 7500$. This is comparable to the time spent solving LPs. As expected, for the basic variant, the number of lexicographic IPs is close to the number of ND images because there is a one-to-one correspondence between a lexicographic IP and a NDLS. The number of scalarized IPs is approximately 3 per 2 ND images. This shows that more scalarized IPs per ND image are solved in the new instances compared to the historical instances. This is due to the characterization of the ND frontier, where the while-loop within the inner loop must usually iterate more than once in a significant portion of the calls. Indeed, the inner loop solves a single scalarized IP 50.3% of the time (on average) for set $n = 5000$ and 59.3% of the time (on average) for set $n = 7500$. A reasonable number

of the boxes processed have corner points with the same integer solution, on average of 21.8% for set $n = 5000$ and 37.4% for set $n = 7500$. This is not surprising given the way the new instances are constructed, i.e., with line segment L_k extending across the ND frontier and contributing many distinct NDLSs. However, *unlike* the historical instances, the ND frontier within these boxes is less likely to be generated entirely by that single integer solution.

As expected, the SIS variant does not improve upon the basic variant for the new instances; in fact, it is a bit slower. First, in the new instances, by design, the solution of the slice problem is not as helpful (in terms of finding multiple ND images of the ND frontier associated with the integer solution of the corner points) because (1) the corner points of the box have the same integer solution of the line segment L_k, which is partially dominated by the pointed cones or (2) the corner points of the box have the integer solution associated to some cone (and each cone produces at most 3 ND images). Second, we found that solving scalarized IPs in which the objective function has the same gradient as the line segment L_k can require excessive amounts of time, most likely because there are multiple optimal solutions to the scalarized IP (due to the way the corner points of the cones are generated). The very first scalarized IP solved can take up to 35% of the total time to find the entire ND frontier for an instance!

Finally, the recursive variant is very efficient for the new instances. Compared to the basic variant, the recursive variant takes, on average, only 71.9% of the total time and solves, on average, 57.2% of the number of IPs for set the $n = 5000$ and 71.5% of the total time and 55.6% of the number of IPs for set the $n = 7500$. It is more efficient because it recurses frequently with the new instances; the maximum depth level ranges from 13 to 17 for instances in the set $n = 5000$ and from 15 to 20 for instances in the set $n = 7500$. As the algorithm recurses, the number of lexicographic IPs drops significantly because many of the line segments of the ND frontier can be discovered and proved optimal by solving only two scalarized IPs throughout the recursion. Furthermore, since the recursive inner

loop does not solve as many LPs, e.g., it does not solve (2.6) to find v, the total number of LPs drops noticeably compared to the basic variant.

Algorithm	Ins	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	nMin	nScal	nGood	nLP	nBox	nSIS	nZL
Basic	21	16850	294	37665.5	33307.6	4339.7	53514	17790	40	17894	-	171370	17803	15926	17766
	22	19778	410	42045.3	36927.4	5095.7	61766	20540	23	20663	-	191611	20510	18074	20476
	23	17319	343	38995.1	34570.3	4409.4	53859	17895	26	18043	-	171514	17884	15776	17871
	24	19898	460	46967.1	41632.8	5312.7	62338	20706	34	20892	-	200619	20696	17867	20682
	25	13682	337	24450.1	21268.9	3171.0	42196	14024	27	14121	-	130934	13994	12130	13964
Avg.		17505.4	368.8	38024.6	33541.4	4465.7	54734.6	18191	30	18322.6	-	173209.6	18177.4	15954.6	18151.8
SIS	21	15699	294	6106.2	4890.5	1211.5	6598	1741	40	1849	1227	43443	2956	1227	1722
	22	18840	410	7788.7	6297.5	1485.3	8613	2287	23	2421	1595	52662	3850	1595	2233
	23	16449	343	6977.4	5607.1	1366.8	7459	1982	26	2131	1338	46627	3309	1338	1961
	24	18546	460	9535.6	7899.2	1630.3	10070	2672	34	2884	1808	56219	4468	1808	2679
	25	13239	337	5329.1	4296.6	1029.2	6578	1753	26	1853	1193	37911	2916	1193	1700
Avg.		16554.6	368.8	7147.4	5798.2	1344.6	7863.6	2087	29.8	2227.6	1432.2	47372.4	3499.8	1432.2	2059
Recursive	21	16831	294	37201.4	32767.3	4417.0	52297	16979	-	18339	-	170040	16971	15611	16955
	22	19763	410	41650	36417.7	5210.1	60312	19600	-	21112	-	189879	19568	17830	19536
	23	17315	343	38684.6	34149.4	4521.7	52585	17042	-	18501	-	170082	17030	15546	17018
	24	19890	460	46196.8	40725.8	5449.5	60571	19576	-	21419	-	198351	19566	17509	19552
	25	13667	337	24635	21314.5	3310.4	40899	13143	-	14613	-	129295	13113	11834	13083
Avg.		17493.2	368.8	37673.6	33074.9	4581.7	53332.8	17268	-	18796.8	-	171529.4	17249.6	15666	17228.8

Table 2.2: Comparison between the different algorithms for historical instances, class C320. Times are reported in seconds.

Algorithm	n	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	nMin	nScal	nGood	nLP	nBox	nSIS	nZL
Basic	5000	15002.0	5001	3924.3	1761.6	1846.8	51786.4	14586.6	466.4	22146.8	-	104131	14546.4	3175.0	14502.6
SIS	5000	15002.0	5001	4522.0	2245.0	1954.0	53575.4	16191.2	379.6	20732.6	80.8	112810.4	16251.4	80.8	16073.2
Recursive	5000	15002.4	5001	2823.6	1341.8	1187.8	29628.2	3633.6	-	22361	-	72896.4	3612.0	1.0	3590.4
Basic	7500	22502.0	7501	8881.4	4109.9	3914.8	81398.0	21846.8	706.4	36998	-	157289.4	21795.4	8148	21740.0
SIS	7500	22502.0	7501	12060.1	7146.5	4017.5	79738.8	24021.0	606.0	30979.8	111.0	168228.0	24098.8	111.0	23888.8
Recursive	7500	22502.6	7501	6346.7	3002.0	2538.4	44452.4	5272.8	-	33906.8	-	109628.6	5240.4	1.2	5208.0

Table 2.3: Comparison between the different algorithms for generated instances with $n = 5000$ and $n = 75000$. All metrics are averaged over five instances. Times are reported in seconds.

2.7.2 Comparison with existing algorithms

Comparing the performance of different algorithms is always challenging, but it is especially difficult for algorithms solving multiobjective mixed integer programs, given that it is non-trivial, in practice, to characterize an optimal solution, i.e., the ND frontier. A ND frontier can contain isolated points as well as open, half-open, and closed segments, and because computers employ finite-precision arithmetic, algorithms have to use tolerances to decide whether two values are equal or different. Changing the tolerance(s) used in an algorithm for finding the ND frontier of a multiobjectiove integer program, as we have seen in Section 2.5, can have a noticeable effect on the resulting ND frontier.

Unfortunately, the use of tolerances in algorithms for finding the ND frontier of a multiobjectiove integer program also makes it more likely that the execution of an algorithm on a different hardware platform exhibits a different behavior, even to the point where it finds the ND frontier for an instance on one hardware platform but fails to do so for the same instance on another hardware platform.

In addition, to compare the performance of algorithms, it is preferable to run the algorithms on the same hardware platform, and, if at all possible, for an algorithm to be the only computationally intensive process running on the platform when computing times are recorded.

We were fortunate in that the developers of the Triangle Splitting Algorithm (TSA) and the ε,Tabu-Constraint Method (εTCM) both made the source code of the implementation of their algorithms available to us. Thus, a comparison of the performance of our proposed algorithms to the performance of these two algorithms could be conducted on the same hardware platform and therefore be fair.

This worked well (mostly) for the instances used in previous computational studies on algorithms for biobjective mixed integer programming, i.e., the historical instances generated according to the scheme proposed by [12], but not so well for the new instances we created. The implementation of the TSA uses an "instance reader" that was customized to

the historical instances, and it was not obvious how to adapt it to the new instances. The implementation of εTCM reads the instances correctly, but terminates after the first integer program is solved.

Furthermore, the implementation of TSA that we had access to uses a *relative* tolerance and as a result only finds an approximation of the ND frontier. More specifically, averaged over the instances in the set C160 and C320, TSA finds 1,519 and 3,140 ND images, respectively, whereas the SIS variant of our proposed algorithms finds, averaged over the instances in the sets C160 and C320, 3,548 and 16,555 ND images, respectively. Therefore, TSA finds 42.8% of the ND images found by the SIS variant of our proposed algorithm for the instances in the set C160 and only 19.0% for the instances in set C320. As a result, it is not meaningful to compare solution times[8].

The implementation of εTCM that we had access to produced almost identical ND frontiers, differing by only a few ND images, but for a few historical instances, it failed to produce a ND frontier (it cycled). We expect that this is due to the use of a different version of CPLEX and the use of a different hardware platform. However, because we felt it was important to perform a thorough and fair comparison on all instances, new as well as historic, we implemented our own version of εTCM, using, whenever possible, data structures and subroutines common to BLM. We validated our implementation of εTCM by confirming that it produced a nearly identical ND frontier to that produced by the original implementation, in very similar computing time, on all instances for which this was possible, i.e, for which the original implementation ran on our platform.

Our overall comparison of algorithms can be found in Table 2.4, which reports on the performance of the SIS variant of BLM and εTCM on the historic instances, and the recursive variant of BLM and εTCM on the new instances.

We observe that the SIS variant of BLM solves fewer IPs than εTCM on the historic instances, which results in faster solution times, 297.9 vs. 345.6, respectively, averaged

[8]For the C160 instances, the total run time of BLM was 7% longer, on average, than the published results for TSA, and for the C320 instances, BLM's run time is 85% longer.

over the C160 instances, and 7,147.4 vs. 10,354.8, respectively, averaged over the C320 instances.

Even though the recursive variant of BLM solves more IPs than εTCM, and far more LPs than εTCM, on the new instances, it still ends up having significantly faster solution times, 2,823.6 vs. 12,156.3, respectively, averaged over the $n = 5000$ instances, and 6,346.7 vs. 37,922.2, respectively, averaged over the $n = 7500$ instances.

In summary, variants of the BLM not only have desirable theoretical properties (in terms of the number of IPs that need to be solved to generate the ND frontier), but are fast in practice and outperform existing algorithms for solving BOMIPs.

Instance	εTCM							BLM						
	nNDP	**nIPF**	**TT**	**IPT**	**LPT**	**nIP**	**nLP**	**nNDP**	**nIPF**	**TT**	**IPT**	**LPT**	**nIP**	**nLP**
16	2762	115	216.6	145.8	70.3	2870	10520	2764	115	206.0	148.6	57.1	1934	8287
17	2949	110	324.9	194.1	130.2	3054	16205	2952	110	257.7	185.6	71.7	1960	8915
18	2739	101	239.3	156.4	82.4	2834	11257	2738	101	232.8	170.5	61.9	1909	8172
19	6151	181	674.1	494.9	177.1	6319	23086	6166	181	577.1	433.1	143.1	3805	18212
20	3125	100	273.0	202.4	69.9	3216	9080	3122	100	216.1	148.8	66.9	1748	8698
Avg.	3545.2	121.4	345.6	238.7	106.0	3658.6	14029.6	3548.4	121.4	297.9	217.3	80.2	2271.2	10456.8
21	15636	295	8836.8	6659.8	2165.8	15923	86957	15699	294	6106.2	4890.5	1211.5	6598	43443
22	18825	410	11671.6	9055.6	2598.7	19205	98134	18840	410	7788.7	6297.5	1485.3	8613	52662
23	16420	343	10437.3	7743.2	2682.4	16752	102411	16449	343	6977.4	5607.1	1366.8	7459	46627
24	18471	457	13028.6	10187.5	2825.8	18968	101218	18546	460	9535.6	7899.2	1630.3	10070	56219
25	13216	337	7799.5	5698.3	2092.7	13518	79133	13239	337	5329.1	4296.6	1029.2	6578	37911
Avg.	16513.6	368.4	10354.8	7868.9	2473.1	16873.2	93570.6	16554.6	368.8	7147.4	5798.2	1344.6	7863.6	47372.4
5000.A	15002	5001	12152.0	11513.9	337.8	25229	24345	15002	5001	2861.8	1383.9	1184.6	29711	72937
5000.B	15002	5001	12185.2	11565.4	337.3	25281	24201	15002	5001	2788.4	1301.3	1192.8	29587	72826
5000.C	15002	5001	12218.9	11586.8	338.4	25292	24156	15004	5001	2827.3	1350.4	1184.6	29538	72824
5000.D	15002	5001	12196.0	11563.4	338.2	25256	24272	15002	5001	2788.5	1306.2	1187.8	29664	72986
5000.E	15002	5001	12029.3	11415.1	334.1	25263	24243	15002	5001	2851.8	1367.3	1189.4	29641	72909
Avg.	15002	5001	12156.3	11528.9	337.2	25264.2	24243.4	15002.4	5001	2823.6	1341.8	1187.8	29628.2	72896.4
7500.A	22502	7501	38542.3	37022.1	701.1	37875	36407	22502	7501	6544.4	3209.7	2535.2	44466	109695
7500.B	22502	7501	38528.6	37024.5	700.7	37889	36369	22502	7501	6329.9	3016.2	2517.2	44485	109677
7500.C	22502	7501	38532.5	37004.0	701.5	37890	36374	22502	7501	6341.6	2964.6	2562.0	44445	109532
7500.D	22502	7501	36466.7	34998.8	671.7	37907	36311	22502	7501	6172.0	2822.3	2538.8	44415	109698
7500.E	22502	7501	37540.8	36076.9	671.3	37893	36369	22505	7501	6345.6	2997.1	2538.9	44451	109541
Avg.	22502	7501	37922.2	36425.3	689.3	37890.8	36366.0	22502.6	7501	6346.7	3002.0	2538.4	44452.4	109628.6

Table 2.4: Comparison between εTCM and BLM (SIS variant for historic and recursive variant for new instances). Times are reported in seconds.

Comments on recently published algorithms for solving BOMIPs

The growing interest in solving BOMIPs is illustrated by two very recent papers (both published while this paper was under review): [33] introduce an algorithm, called One Direction Search (ODS), based on ideas similar to εTCM, and [34] introduces an algorithm, called Search-and-Remove (SaR), which cleverly combines dichotomic search, solving slice problems, and no-good constraints. For the sake of completeness, we provide some comments on the computational results reported in these papers.

[33] and [34] provide a comparison with published results for TSA on historic instances, classes C160 and C320, acknowledging the challenges associated with such a comparison as the their results were obtained on different hardware platforms and using with different versions of CPLEX.

The results for ODS with relative error $\xi = 10e^{-5}$ [33] show that ODS takes more time to produce the ND frontier for class C160 instances and about the same time or slightly less time for class C320 instances. The main contribution, however, and the authors' primary objective, was demonstrating that more accurate ND frontier approximations can be achieved (compared to TSA).

The results for SaR with 10 subregions [34] show substantial improvement over TSA. On average, SaR produced the ND frontier 32% faster than TSA for C160 instances, and 7% faster than TSA for C320 instances, where SaR was faster in 9 out of 10 instances, by 37% on average, but struggled with one problematic instance. We note that SaR is especially well-suited for the historic instances, because their ND frontiers consist of a relatively small number of continuous sections, generated by relatively few slices, each having many (small) line segments.

CHAPTER 3

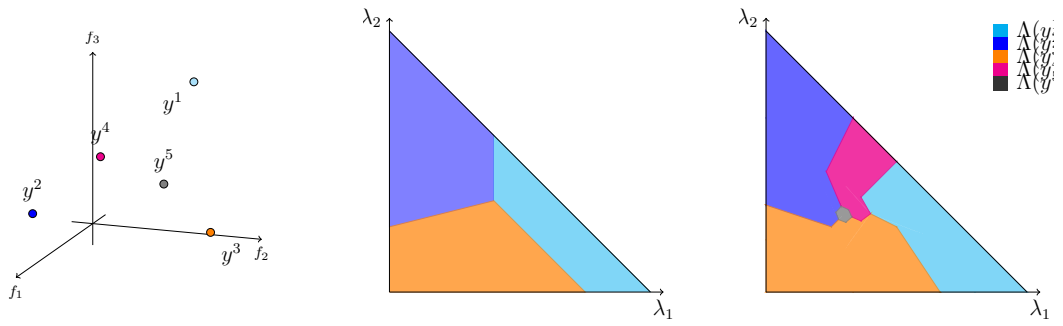
TCHEBYCHEV WEIGHT SPACE DECOMPOSITION: GEOMETRY

Abstract. *Scalarization is a common technique to transform a multiobjective programming problem into a scalar-valued optimization problem. This article deals with the weighted Tchebychev scalarization applied to multiobjective discrete optimization problems. It consists of minimizing a weighted distance of the image of a feasible solution to some desirable reference point. By choosing a suitable weight, any Pareto optimal image can be obtained. In this article, we provide a comprehensive theory of this set of eligible weights. In particular, we analyze the polyhedral and combinatorial structure of the set of all weights yielding the same (Pareto) optimal solution as well as the decomposition of the weight set as a whole. The structural insights are linked to properties of the set of (Pareto) optimal solutions, thus providing a profound insight into the weighted Tchebychev scalarization method and, as a consequence, also into all methods for multiobjective programming problems using this scalarization as a building block.*[1]

The basic idea of a weight set decomposition for a multiobjective discrete optimization problems (MODOs) is quite intuitive and has been explored extensively for the weighted sum scalarization [49, 56, 50, 57, 58]. Each nondominated (ND) image has an associated *weight set component*: the set of weight vectors for which the weighted sum scalarization yields the ND image. The *weight set decomposition* is usually taken to be a (minimal) collection of weight set components that cover the set of eligible parameters. Figure 3.1 illustrates the triangular weight set, $\Lambda = \{\lambda \in \mathbb{R}^3_{\geq} : \sum_i \lambda_i = 1\}$, and (a) represents the well-studied weight set decomposition by weighted sum scalarization.

Weight set components offer decision-makers additional insight into the ND frontier, and can be especially useful for three or more objectives, when visualization of the frontier itself is more difficult. For example, a weight set component with a comparatively large

[1]This work was submitted to *Journal on Global Optimization* (January 2021), coauthored with Stephan Helfrich, Pascal Halffmann, Natashia Boland, and Stefan Ruzika, with Tyler Perini as second author.



(a) (b) (c)

Figure 3.1: An example with image set $y^1 = (1, 3, 4)^\top$, $y^2 = (8, 2, 4)^\top$, $y^3 = (4, 8, 2)^\top$, $y^4 = (6, 4, 6)^\top, y^5 = (7, 7, 5)$ and their weight set components $\Lambda(y^r)$, $r = 1, \ldots, 5$, for both weighted sum scalarization (b) and weighted Tchebychev scalarization (c). Note that, for both scalarization, the restriction to weights contained in $\Lambda = \{\lambda \in \mathbb{R}^P_{\geq} : \sum_{i=1}^{p} \lambda_i = 1\}$ is without loss of generality. Thus, $\lambda_3 = 1 - \lambda_1 - \lambda_2$. The image y^4 is not extreme supported. The image y^5 is not supported. The images y^1 and y^2 are adjacent with respect to the weighted sum weight set decomposition though their weighted Tchebychev weight set components do not intersect.

volume is obtained from a ND image that is in some sense more "robust": there are comparatively many weight vectors which yield this image. The intersections of weight set components also embody the adjacency structure of the ND frontier: the higher the dimension of the intersection of two weight set components, the "closer" in the ND frontier are the images they are obtained from. In addition to their value to decision-makers, construction of weight set components may also form an integral part of algorithms for generating ND frontiers or approximations to them. The adjacency structure can be especially helpful in the design of interactive methods.

However, one may have *unsupported* ND images, which are attributed empty sets of weights by this weighted scalarization. Even more, it is not difficult to construct nonconvex image sets in which almost the entire ND frontier is unsupported. This motivates the use of a different scalarization:

Recall the *weighted Tchebychev scalarization*, which with a *reference point* $s \in \mathbb{R}^p$ and

given weight vector $\lambda \in \mathbb{R}^p_{\geq}$, can be stated as

$$\min \{\|f(x) - s\|^{\lambda}_{\infty} \ : \ x \in \mathcal{X}\}, \qquad (\Pi^{TS}(\lambda))$$

where $\|y\|^{\lambda}_{\infty} := \max_{i=1,\dots,p}\{|\lambda_i \ y_i|\}$. Although the weighted Tchebychev scalarization is well-known, there has been no effort to investigate its weight set decomposition prior to this work. We provide a first rigorous and comprehensive theory on the weighted Tchebychev weight set components, and we analyze the polyhedral and combinatorial structure of the sets and provide an adjacency concept of ND images.

3.1 Related Work

To the best of our knowledge, the Tchebychev norm was introduced for biobjective optimization problems by Geoffrion in 1967 [59]. Bowman [60] and Wierzbicki [61], among others, suggest using the (weighted) Tchebychev norm to generate all ND images of MOPs, even for nonlinear objective functions. To avoid weakly ND images, modifications are introduced: the lexicographic weighted Tchebychev scalarization[20] chooses among all images that are optimal the image with minimal sum-norm. The *augmented weighted Tchebychev norm* [20] adds the sum-norm scaled with a small parameter. The *modified augmented weighted Tchebychev norm* [62] also uses weights in the augmentation term.

Since the distance measure to the reference point gives useful information during the optimization process, many applications of these Tchebychev scalarization techniques can be found in the context of interactive approaches; see [63] for an overview. For example, Steuer and Choo [20] utilize the (augmented) weighted Tchebychev scalarization, while Luque et al. [64] develop such an approach for solving convex multiobjective programs using the lexicographic weighted Tchebychev scalarization. For multiobjective mixed integer linear programming, Alves and Clímaco [65] combine a branch-and-bound approach with iterative adjustments of the reference point, employing the augmented weighted Tcheby-

chev scalarization. Bozkurt et al. [66] use the weighted Tchebychev scalarization to evaluate the quality of solutions.

Weight set decomposition methods for the weighted sum scalarization date back to the work of Yu and Zeleny in 1975 [67], who introduce a generalized simplex method and link basic efficient solutions with the set of weights in the polyhedral cone defined by the corresponding basis matrix. For biobjective problems, the well-known dichotomic search [22, 21] in fact calculates all extreme supported ND (ESND) images based on the general idea of weight set decomposition: for two known ESND images, it calculates the unique weight such that the 1-norm of both coincide. Solving the related weighted sum scalarization problem leads either to a new ESND image, or justifies that one cannot find other ESND images 'between' the two. Benson and Sun [47, 48] extend this idea and establish a link between ESND images of a multiobjective linear optimization problem and a partitioning of the weight set.

Przybylski et al. [58] adapt this technique to MODOs. They state fundamental properties concerning the weight set components: Each weight set component, denoted $\Lambda^{WS}(y)$ for ND image y, is a (convex) polytope, and knowing all ESND images is sufficient for its calculation. A weight set component is full-dimensional if and only if the corresponding image is an ESND one, which implies that the set of ESND images is sufficient and necessary to cover the whole weight set. Further, two weight set components intersect in common faces, only. That is, there exists a face F of $\Lambda^{WS}(y)$ and a face F' of $\Lambda^{WS}(y')$ such that $F = F' = \Lambda^{WS}(y) \cap \Lambda^{WS}(y')$. Based on this symmetry, two ESND images are defined to be adjacent if and only if the dimension of their intersection is one less than the dimension of the weight set. Finally, they also present an algorithm for computing all ESND images for three objectives using the derived properties by iteratively shrinking supersets of the actual weight set components (a.k.a., outer approximations).

The weight set decomposition is implicitly calculated by the procedures of Özpeynirci and Köksalan [57] and Bökler and Mutzel [56]. The algorithms of Alves and Costa [68]

and Halffmann et al. [50] iteratively augment subsets of the weight set components (a.k.a., inner approximations). Recently, Schulze et al. [69] use a weight set decomposition linked with so-called arrangements of hyperplanes in the objective space to show that the number of ESND images of unconstrained MODOs is polynomially bounded.

For the weighted Tchebychev scalarization, Eswaran et al. [70] explicitly consider weight set components for biobjective problems. Based on this approach, Ralphs et al. [71] adapt the dichotomic search method to calculate all ND images of biobjective discrete optimization problems.

3.1.1 Contributions

We apply the weight set decomposition approach to the weighted Tchebychev scalarization of MODOs, which has not been done before for more than two objectives. As shown in Figure 3.1(b), the weighted Tchebychev scalarization implies a more sophisticated and rich structure in comparison to the weighted sum scalarization. The primary contribution of this work is a theoretical study of this structure and its properties, which provides a theoretical foundation for the development of new algorithms to compute the weighted Tchebychev weight set decomposition, which in particular includes the enumeration of all ND images of a MODOs.

We provide foundational properties of the weighted Tchebychev weight set decomposition in Section 3.2. We show that it is necessary and sufficient to consider only the weight set components for ND images and establish that weight set components are nonconvex but have convexity related properties: they are star-shaped as well as convex along rays emanating from a vertex of the weight set. We study the intersection of weight set components in Section 3.3. Such intersections coincide with weight set components of special weakly ND images, and hence all convexity-related properties also apply, even though intersections of star-shaped sets are, in general, not star-shaped. In Section 3.4, we describe weight set components as unions of finitely many polytopes. This lays a well-defined foundation

of the dimensional analysis presented in Section 3.5 including an adjacency structure for weight set components to reveal the "organization" of the ND frontier. We close with some concluding remarks in Section 3.6.

3.2 Foundations

In this section, we introduce fundamental concepts for analyzing the weight set decomposition according to the weighted Tchebychev scalarization. We also derive properties connecting the weight set with the ND set $\mathcal{Y}_N$ and investigate convexity properties. The following two definitions, which are rarely seen in multiobjective optimization, are fundamental for the remainder of this work.

Definition 1 (Preparata and Shamos [72])**.** *A set $S \subseteq \mathbb{R}^p$ is* star-shaped*, if there exists a $y \in S$ such that $\theta y + (1-\theta)\bar{y} \in S$ for all $\bar{y} \in S$ and all $\theta \in (0,1)$. The set of all such images y is called* kernel *of S and is denoted by* $\ker(S)$.

Definition 2 (Ziegler [73])**.** *A* polytopal complex *$\mathcal{C}$ is a finite collection of polytopes in $\mathbb{R}^d$ such that*

- *the empty polytope is in $\mathcal{C}$,*
- *if $P \in \mathcal{C}$, then all the faces of P are also in $\mathcal{C}$,*
- *the intersection $P \cap Q$ of two polytopes $P, Q \in \mathcal{C}$ is a face of both P and Q.*

The *dimension* $\dim(\mathcal{C})$ is the largest dimension of a polytope in $\mathcal{C}$. The *underlying set* of $\mathcal{C}$ is the point set $\bigcup_{P\in\mathcal{C}} P$. A *subcomplex* of a polytopal complex is a subset $\mathcal{C}' \subseteq \mathcal{C}$ that itself is a polytopal complex. A *polytopal subdivision* of a set $S \subseteq \mathbb{R}^d$ is a polytopal complex $\mathcal{C}$ with the underlying set $\bigcup_{P\in\mathcal{C}} P = S$. For example, it is easy to see that the collection of all faces of polytope P defines a polytopal subdivision of P itself.

In the remainder of this paper, we consider MODOs. That is, $\mathcal{X}$ is a finite set. Further, we make the following assumption on the reference point used in the weighted Tchebychev scalarization.

Assumption 1. *The reference point s is a utopia point. Thus, $s < y$ for all $y \in \mathcal{Y}$, and without loss of generality, we can also assume that $\mathcal{Y} \subseteq \mathbb{R}^p_>$ (by shifting) and the reference point s used in the weighted Tchebychev scalarization is the zero vector ($s = 0$).*

As a consequence of Assumption 1, $\Pi^{TS}(\lambda)$ simplifies to $\min\{\|f(x)\|^\lambda_\infty : x \in \mathcal{X}\} = \min\{\|y\|^\lambda_\infty : y \in \mathcal{Y}\}$. Furthermore, $\|y\|^\lambda_\infty > 0$ for all $y \in \mathcal{Y}$ and $\lambda \in \Lambda$, since $\lambda \geq 0$ with $\lambda \neq 0$, for all $\lambda \in \Lambda$.

The latter part of the following proposition is given as Theorem 4.5 in [20]; here we extend it to the case of weakly ND images.

Proposition 3. *For all $y \in \mathcal{Y}_{wN}$ there exists a weight $\lambda \in \mathbb{R}^p_>$ such that y minimizes $\Pi^{TS}(\lambda)$. Moreover, if $y \in \mathcal{Y}_N$, there exists a weight such that y uniquely minimizes $\Pi^{TS}(\lambda)$.*

Proof. For $y \in \mathcal{Y}_{wN}$ choose the weight λ defined by $\lambda_i = 1/y_i > 0$ for $i = 1, \ldots, p$. Suppose there exists a $\bar{y}$ such that $\|y\|^\lambda_\infty > \|\bar{y}\|^\lambda_\infty$. Then $\max_{i=1,\ldots,p} \lambda_i y_i > \max_{j=1,\ldots,p} \lambda_j \bar{y}_j$ which implies for all $j = 1, \ldots, p$, $\lambda_j y_j = 1 > \lambda_j \bar{y}_j$ and thus $y_j = 1/\lambda_j > \bar{y}_j$. This contradicts $y \in \mathcal{Y}_{wN}$. To prove the second statement, choose again the weight λ defined by $\lambda_i = 1/y_i$ for $i = 1, \ldots, p$. Then similar calculations imply that for an image $y \neq \bar{y} \in \mathcal{Y}$ with $\|y\|^\lambda_\infty \geq \|\bar{y}\|^\lambda_\infty$, it must be that $y_j \geq \bar{y}_j$ holds for all $j = 1, \ldots, p$. This is a contradiction to $y \in \mathcal{Y}_N$. □

Since $\alpha\|y\|^\lambda_\infty = \|y\|^{\alpha\lambda}_\infty$ holds for all scalars $\alpha > 0$, normalization of the weight λ does not change the optimal solution set of $\Pi^{TS}(\lambda)$. Hence, analogously to the weighted sum method, we restrict the set of eligible parameters to the (normalized) *weight set*

$$\Lambda := \left\{ \lambda \in \mathbb{R}^p_{\geqq} : \sum_{k=1}^{p} \lambda_k = 1 \right\}. \tag{3.1}$$

Observe that Λ does not contain the zero vector. Further, Λ is a $(p-1)$ dimensional polytope, and the projection/bijection $\phi : \Lambda \to \{\lambda \in \mathbb{R}^{p-1}_{\geqq} : \sum_{i=1}^{p-1} \lambda_i \leq 1\}, (\lambda_1, \ldots, \lambda_p) \mapsto (\lambda_1, \ldots, \lambda_{p-1})$ gives a particularly useful tool for the visualization of the weight sets of

MODOs with three objectives. We introduce its decomposition implied by the weighted Tchebychev scalarization:

Definition 3. *For $y \in \mathcal{Y}$, the* weight set component *of y with respect to the weighted Tchebychev scalarization is defined by*

$$\Lambda(y) := \left\{\lambda \in \Lambda : \|y\|_\infty^\lambda \leq \|\bar{y}\|_\infty^\lambda \text{ for all } \bar{y} \in \mathcal{Y}\right\}.$$

It is helpful to note that $\lambda \in \Lambda(y)$ if and only if y is optimal for $(\Pi^{TS}(\lambda))$, meaning that $y = f(x)$ for some optimal solution x of $(\Pi^{TS}(\lambda))$. Obviously, if an image is not weakly ND, then its weight set component is empty.

We introduce a notation for the normalization of the weight used in the proof of Proposition 3.

Definition 4. *For $y \in \mathcal{Y}_{wN}$ we denote the* kernel weight *or* kernel vertex *(also known as* T-vertex *[64]) of y by $\lambda(y)$ and define it by*

$$\lambda_i(y) := \frac{1}{y_i} \frac{1}{\sum_{j=1}^{p} 1/y_j} \text{ for } i = 1, \ldots, p.$$

Proposition 3 implies that if y is weakly ND then its weight set component is nonempty. Hence, an image is weakly ND if and only if its weight set component is nonempty. Moreover, if y is ND, we get the following as a corollary.

Corollary 1. *Let $B_\varepsilon(\lambda) := \{\lambda' \in \Lambda : \sum_{i=1}^{p} \lambda_i - \lambda_i' \leq \varepsilon\}$. For $y \in \mathcal{Y}_N$, there exists an $\varepsilon > 0$ such that $B_\varepsilon(\lambda(y)) \subseteq \Lambda(y)$. If ε is chosen small enough, then $B_\varepsilon(\lambda(y)) \cap \Lambda(y') = \emptyset$ for each $y' \in \mathcal{Y}_{wN} \setminus \{y\}$.*

Proof. This follows by Proposition 3, the definition of the kernel vertex, finiteness of the feasible set, and the continuity of the function $\lambda \to \|\bar{y}\|_\infty^\lambda$ in all components of λ for any given $\bar{y} \in \mathcal{Y}_{wN}$. □

The next propositions show that $\mathcal{Y}_N$ is sufficient to define the weight set components of all images.

Proposition 4. *Let $y \in \mathcal{Y}$. Then*

$$\Lambda(y) = \{\lambda \in \Lambda : \|y\|_\infty^\lambda \leq \|\bar{y}\|_\infty^\lambda \text{ for all } \bar{y} \in \mathcal{Y}_N\}.$$

Proof. Let $\bar{y} \in \mathcal{Y} \setminus \mathcal{Y}_N$. Then, since $\mathcal{Y}$ is finite, there exists an image $y' \in \mathcal{Y}_N$ such that $y' \leq \bar{y}$. This yields $\|y'\|_\infty^\lambda \leq \|\bar{y}\|_\infty^\lambda$ for all $\lambda \in \Lambda$. Then we have $\|y\|_\infty^\lambda \leq \|y'\|_\infty^\lambda \leq \|\bar{y}\|_\infty^\lambda$ for $\lambda \in \Lambda(y)$. That is, the inequality $\|y\|_\infty^\lambda \leq \|\bar{y}\|_\infty^\lambda$ is redundant. □

The following proposition shows that all weights $\lambda \in \Lambda$ map to a ND image in $\mathcal{Y}_N$ by optimizing $\Pi^{TS}(\lambda)$.

Proposition 5. $\Lambda = \bigcup_{y \in \mathcal{Y}_N} \Lambda(y)$.

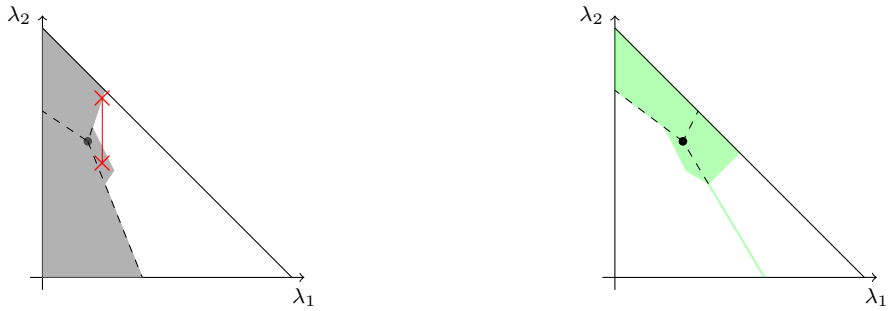
Proof. For a weight $\lambda \in \Lambda$ there exists an image $y \in \mathcal{Y}$ that is optimal for $\Pi^{TS}(\lambda)$, and hence $\lambda \in \Lambda(y)$. If $y \notin \mathcal{Y}_N$, then (by finiteness of $\mathcal{Y}$) there exists $\bar{y} \in \mathcal{Y}_N$ such that $\bar{y}_i \leq y_i$ for all $i = 1, \ldots, p$. Since $\lambda \geq 0$ this gives $\|\bar{y}\|_\infty^\lambda \leq \|y\|_\infty^\lambda$, and so $\bar{y}$ must also be optimal for $\Pi^{TS}(\lambda)$. Thus $\lambda \in \Lambda(\bar{y})$, too. Hence $\Lambda \subseteq \bigcup_{\bar{y} \in \mathcal{Y}_N} \Lambda(\bar{y})$. The reverse inclusion holds trivially. □

Proposition 3 implies another fact about the weight set components: weight set components of ND images cannot be a subset of one another. In particular, if $y \in \mathcal{Y}_N$, we have $\Lambda(y) \setminus \left(\bigcup_{\bar{y} \in \mathcal{Y}_N, \bar{y} \neq y} \Lambda(\bar{y})\right) \neq \emptyset$. Thus, Proposition 5 states a sufficient and necessary condition to decompose the weight set.

We begin with a crucial geometric property of the weight set components: in contrast to the weighted sum weight set components, the sets $\Lambda(y)$ are not necessarily convex.

Example 6. *Consider the set of ND images*

$$\mathcal{Y} = \left\{y^1 = (3,1,2)^\top, y^2 = (2,1,3)^\top, y^3 = (2,2,2)^\top, y^4 = (1,2,3)^\top\right\}.$$



(a) $\Lambda(y^1)$ (b) $\Lambda(y^2)$

(c) $\Lambda(y^3)$ (d) $\Lambda(y^4)$

Figure 3.2: The weight set components (a) - (d) of Example 6. For $r = 1, 2, 3, 4$, the colored regions represent $\Lambda(y^r)$, the dots represents the kernel vertex $\lambda(y^r)$, and the red line in (a) represents the convex combination of weights λ^1 and λ^2 investigated in Example 6. The dashed lines indicate the decomposition of the weight set components into its dimensional weight set components. Notice that the "lower dimensional parts" along these dashed lines belong to both adjacent dimensional weight set components. Indeed, such "lower dimensional parts" can only occur in the intersection of dimensional weigt set components (see Proposition 12).

Figure 3.2 shows the weight set components for Example 6. Let $\lambda^1 = (0.24, 0.72, 0.04)^\top$ *and* $\lambda^2 = (0.24, 0.46, 0.3)^\top$. *Then, simple calculations yield* $\lambda^1, \lambda^2 \in \Lambda(y^1)$ *(see Figure 3.2(a)). However, for* $\lambda^3 := {}^1\!/_2\lambda^1 + {}^1\!/_2\lambda^2 = (0.24, 0.59, 0.17)^\top$ *we have* $\|y^1\|_\infty^{\lambda^3} = 0.72 > 0.59 = \|y^2\|_\infty^{\lambda^3}$ *and therefore* $\lambda^3 \notin \Lambda(y^1)$. *Consequently,* $\Lambda(y^1)$ *is not convex.*

To gain more insights about the structure of weight set components, we divide components into smaller subsets according to which index maximizes the corresponding scalar product (i.e., defining the weighted Tchebychev norm value).

Definition 5. *Let* $y \in \mathcal{Y}_{wN}$. *We define for* $i = 1, \ldots, p$ *the* ith dimensional weight set

component *by*

$$\Lambda(y,i) := \{\lambda \in \Lambda(y) : \lambda_i y_i \geq \lambda_k y_k \textit{ for all } k = 1,\dots,p\}.$$

Clearly, $\Lambda(y,i) = \{\lambda \in \Lambda(y) : \|y\|_\infty^\lambda = \lambda_i y_i\}$ and $\bigcup_{i=1}^p \Lambda(y,i) = \Lambda(y)$. Figure 3.2 presents these sets for Example 6.

With the image set of Example 6, one can also show that both λ^1 and λ^2 are contained in $\Lambda(y^1,1)$ and thus the dimensional weight set components are not necessarily convex. However, we can derive a "convexity-related" property.

Proposition 6. *Let $y \in \mathcal{Y}_{wN}$. Then the following holds.*

1. $\bigcap_{i=1}^p \Lambda(y,i) = \{\lambda(y)\}$.
2. *For $i = 1,\dots,p$, $\Lambda(y,i)$ is a star-shaped set with $\lambda(y) \in \ker(\Lambda(y,i))$.*

Proof. Since $\lambda_i(y)y_i = {}^1\!/\!_{\sum_{j=1}^p 1/y_j}$ for all $i = 1,\dots,p$, we have $\lambda(y) \in \Lambda(y,i)$ for all $i = 1,\dots,p$. Furthermore, if a weight $\lambda \in \bigcap_{i=1}^p \Lambda(y,i)$, we have $\lambda_j y_j \geq \lambda_k y_k$ for all $j,k = 1,\dots,p$. This yields $\lambda_1 y_1 = \lambda_2 y_2 = \cdots = \lambda_p y_p = M$ if and only if $\lambda_i = {}^M\!/\!_{y_i}$ for all $i = 1,\dots,p$ for a constant $M \in \mathbb{R}$. Since $\sum_{i=1}^p \lambda_i = 1$, we get $M = (\sum_{i=1}^p {}^1\!/\!_{y_i})^{-1}$ and therefore λ is the kernel weight. This shows statement (i).

To prove (ii), fix i and let $\lambda' \in \Lambda(y,i)$. We first show that for $\theta \in (0,1)$, the convex combination $(\theta\lambda(y) + (1-\theta)\lambda') \in \Lambda(y)$. To do so we prove for all $\bar{y} \in \mathcal{Y}$ that $\|y\|_\infty^{\theta\lambda(y)+(1-\theta)\lambda'} \leq \|\bar{y}\|_\infty^{\theta\lambda(y)+(1-\theta)\lambda'}$. To begin, observe that since $\lambda(y), \lambda' \in \Lambda(y,i)$, it follows that $\|y\|_\infty^{\lambda(y)} = \lambda_i(y)y_i \geq \lambda_k(y)y_k$ and $\|y\|_\infty^{\lambda'} = \lambda'_i y_i \geq \lambda'_k y_k$ for all $k = 1,\dots,p$. Fix $\theta \in (0,1)$. It is now straightforward to show that for all $k = 1,\dots,p$,

$$\begin{aligned}(\theta\lambda_i(y) + (1-\theta)\lambda'_i)y_i &= \theta\lambda_i(y)y_i + (1-\theta)\lambda'_i y_i \\ &\geq \theta\lambda_k(y)y_k + (1-\theta)\lambda'_k y_k = (\theta\lambda_k(y) + (1-\theta)\lambda'_k)y_k,\end{aligned}$$

and so

$$\|y\|_\infty^{\theta\lambda(y)+(1-\theta)\lambda'} = (\theta\lambda_i(y) + (1-\theta)\lambda'_i)y_i. \tag{3.2}$$

Fix $\bar{y} \in \mathcal{Y}$. Now for some j, it must be that $\lambda'_j\bar{y}_j = \|\bar{y}\|_\infty^{\lambda'}$. By Assumption 1, $\|\bar{y}\|_\infty^{\lambda'} > 0$ so $\lambda'_j > 0$. Since $\lambda' \in \Lambda(y)$, we have that $\|y\|_\infty^{\lambda'} \leq \|\bar{y}\|_\infty^{\lambda'}$. So, $\lambda'_i y_i = \|y\|_\infty^{\lambda'} \leq \lambda'_j\bar{y}_j$. Furthermore, $\lambda'_j y_j \leq \lambda'_i y_i$ by the definition of $\|y\|_\infty^{\lambda'}$. Thus, $\lambda'_j y_j \leq \lambda'_j\bar{y}_j$, which implies $y_j \leq \bar{y}_j$, since $\lambda'_j > 0$. Hence, $\lambda_j(y)y_j \leq \lambda_j(y)\bar{y}_j$. But $\lambda_j(y)y_j = {}^1/_{\sum_{k=1}^p {}^1/_{y_k}} = \lambda_i(y)y_i$ so it is also the case that $\lambda_i(y)y_i \leq \lambda_j(y)\bar{y}_j$. Altogether, we find

$$\begin{aligned}\|y\|_\infty^{\theta\lambda(y)+(1-\theta)\lambda'} &= \theta\lambda_i(y)y_i + (1-\theta)\lambda'_i y_i \leq \theta\lambda_j(y)\bar{y}_j + (1-\theta)\lambda'_j\bar{y}_j \\ &= (\theta\lambda_j(y) + (1-\theta)\lambda'_j)\bar{y}_j \leq \|\bar{y}\|_\infty^{\theta\lambda(y)+(1-\theta)\lambda'}\end{aligned}$$

and thus $(\theta\lambda(y) + (1-\theta)\lambda') \in \Lambda(y)$. From (3.2) we immediately get that $(\theta\lambda(y) + (1-\theta)\lambda') \in \Lambda(y,i)$ which completes the proof. □

The first property states that the kernel weight is the only possible weight that is contained in all dimensional weight set components. The second property justifies the name kernel weight. The most interesting statement is Proposition 6.2, which combined with Proposition 6.1 gives the following.

Corollary 2. *Let $y \in \mathcal{Y}_{wN}$. Then, $\Lambda(y)$ is a star-shaped set and $\lambda(y) \in \ker(\Lambda(y))$.*

In the one dimensional weight set (i.e. for two objectives) star-shapedness is equivalent to convexity of the weight set components. This explains why the straightforward adaption of the dichotomic search approach to the weighted Tchebychev scalarization proposed in [70, 71] works in the biobjective case.

A second convexity-related property can be derived with the help of the following lemma. Notice that we fix $p-1$ components of a weight $\lambda \in \mathbb{R}^p_>$ and do not restrict ourselves to normalized weights.

Lemma 3. *For a given index k, $\lambda \in \mathbb{R}^p_{\geq}$ and a scalar $t > 0$, if $y \in \mathcal{Y}$ is optimal for both $\Pi^{TS}(\lambda)$ and $\Pi^{TS}(\lambda + te_k)$, where e_k is the kth unit vector in $\mathbb{R}^p$, then y is also optimal for $\Pi^{TS}(\lambda + \theta te_k)$ for all $\theta \in (0, 1)$.*

Proof. First observe that for any $y \in \mathcal{Y}$, $\theta \in [0, 1]$, and k, λ and t as given,

$$\|y\|_\infty^{\lambda+\theta te_k} = \max\{\lambda_i y_i, (\lambda_k + \theta t)y_k\}$$

where i is an index such that $\|y\|_\infty^\lambda = \lambda_i y_i$. Consider $y \in \mathcal{Y}$ optimal for both $\Pi^{TS}(\lambda)$ and $\Pi^{TS}(\lambda + te_k)$, fix i^* such that $\|y\|_\infty^\lambda = \lambda_{i^*} y_{i^*}$ and fix $\theta \in (0, 1)$. Let $\bar{y} \in \mathcal{Y}$ and fix i such that $\|\bar{y}\|_\infty^\lambda = \lambda_i \bar{y}_i$. Thus,

$$\|y\|_\infty^{\lambda+\theta te_k} = \max\{\lambda_{i^*} y_{i^*}, (\lambda_k + \theta t)y_k\} \text{ and } \|\bar{y}\|_\infty^{\lambda+\theta te_k} = \max\{\lambda_i \bar{y}_i, (\lambda_k + \theta t)\bar{y}_k\}.$$

We consider two cases for $\|\bar{y}\|_\infty^{\lambda+te_k}$. In each we show that $\|y\|_\infty^{\lambda+\theta te_k} \leq \|\bar{y}\|_\infty^{\lambda+\theta te_k}$. In both cases, we use the observation that $\lambda_{i^*} y_{i^*} \leq \lambda_i \bar{y}_i$ since y is optimal for $\Pi^{TS}(\lambda)$.

(i) Suppose $\|\bar{y}\|_\infty^{\lambda+te_k} = \lambda_i \bar{y}_i$. Since $\theta < 1$, $t > 0$ and $\bar{y}_k > 0$, we can conclude that

$$(\lambda_k + \theta t)\bar{y}_k < (\lambda_k + t)\bar{y}_k \leq \|\bar{y}\|_\infty^{\lambda+te_k} = \lambda_i \bar{y}_i$$

and therefore $\|\bar{y}\|_\infty^{\lambda+\theta te_k} = \lambda_i \bar{y}_i = \|\bar{y}\|_\infty^{\lambda+te_k}$. Likewise,

$$(\lambda_k + \theta t)y_k < (\lambda_k + t)y_k \leq \|y\|_\infty^{\lambda+te_k} \leq \|\bar{y}\|_\infty^{\lambda+te_k} = \|\bar{y}\|_\infty^{\lambda+\theta te_k},$$

where the last inequality follows by optimality of y for $\Pi^{TS}(\lambda + te_k)$. Recall that $\lambda_{i^*} y_{i^*} \leq \lambda_i \bar{y}_i = \|\bar{y}\|_\infty^{\lambda+\theta te_k}$. Thus

$$\|y\|_\infty^{\lambda+\theta te_k} = \max\{\lambda_{i^*} y_{i^*}, (\lambda_k + \theta t)y_k\} \leq \|\bar{y}\|_\infty^{\lambda+\theta te_k}.$$

(ii) Suppose $\|\bar{y}\|_\infty^{\lambda+te_k} = (\lambda_k + t)\bar{y}_k$. Now $\|y\|_\infty^{\lambda+te_k} \leq \|\bar{y}\|_\infty^{\lambda+te_k}$ since y is optimal for

$\Pi^{TS}(\lambda + te_k)$. So $(\lambda_k + t)y_k \leq (\lambda_k + t)\bar{y}_k$. Since $t > 0$ and $\lambda_k \geq 0$, it must be that $y_k \leq \bar{y}_k$, and hence $(\lambda_k + \theta t)y_k \leq (\lambda_k + \theta t)\bar{y}_k \leq \|\bar{y}\|_\infty^{\lambda+\theta te_k}$. Furthermore, recall that $\lambda_{i^*} y_{i^*} \leq \lambda_i \bar{y}_i \leq \|\bar{y}\|_\infty^{\lambda+\theta te_k}$. Hence we again have

$$\|y\|_\infty^{\lambda+\theta te_k} = \max\{\lambda_{i^*} y_{i^*}, (\lambda_k + \theta t)y_k\} \leq \|\bar{y}\|_\infty^{\lambda+\theta te_k}.$$

□

The above lemma shows that given λ^1 and λ^2 with $\lambda^2 - \lambda^1$ equal to a positive multiple of a unit vector: if y is optimal for both $\Pi^{TS}(\lambda^1)$ and $\Pi^{TS}(\lambda^2)$, then y is also optimal for $\Pi^{TS}(\lambda)$ where λ is any convex combination of λ^1 and λ^2.

In order to transfer this result to the weight set, for a given index $k \in \{1, \ldots, p\}$ and a vector $a \in \mathbb{R}^p$ such that $a_i > 0$ for $i \neq k$, we define the following lines:

$$H_{k,a} := \{\lambda \in \mathbb{R}^p : a_i \lambda_i = a_j \lambda_j \text{ for all } i, j \in \{1, \ldots, p\} \setminus \{k\}\} \cap \Lambda. \tag{3.3}$$

Figure 3.3 shows a selection of these lines. Visually, the lines $H_{k,a} \cap \Lambda$ emanate from one of the vertices of Λ. This can be seen by rewriting

$$H_{k,a} = \{\lambda \in \mathbb{R}^p : \lambda = e_k + (a' - e_k)t \text{ for some } t \in [0, 1]\},$$

where e_k denotes the kth unit vector and a' is defined by $a'_k := 0$ and $a'_i := \frac{1}{a_i} \frac{1}{\sum_{j \neq k} 1/a_j}$ for $i \neq k$. It is precisely along these lines that we find a new convexity-related property.

Proposition 7. *For any $k \in \{1, \ldots, p\}$ and $a \in \mathbb{R}^p$ such that $a_i > 0$ for $i \neq k$, the intersection set $\Lambda(y) \cap H_{k,a}$ is convex for all $y \in \mathcal{Y}_{wN}$.*

Proof. We prove that $\Lambda(y) \cap \Lambda^> \cap H_{k,a}$ is convex for all $y \in \mathcal{Y}$. Without loss of generality, let $k = p$. Dividing all entries by a_1 does not change the validity of any equality in (3.3).

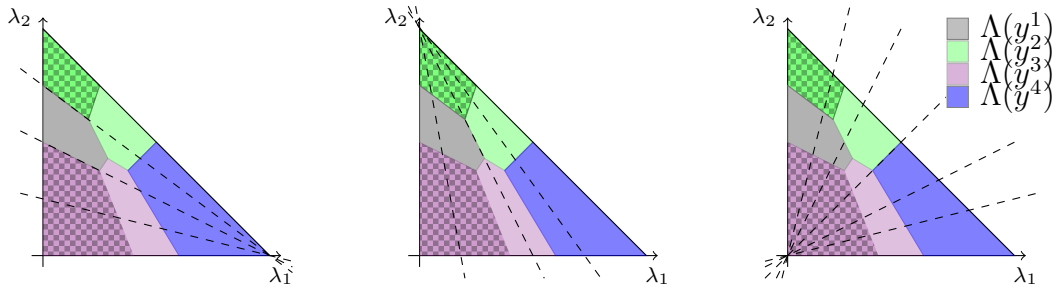


Figure 3.3: The convexity property of Proposition 7. The intersection of the planes of the form $H_{k,a}$ (dashed lines) in (3.3) and the weight set components are always convex sets. The green-gray and violet-gray checkerboard areas represent the intersection of weight set components $\Lambda(y^1) \cap \Lambda(y^2)$ and $\Lambda(y^1) \cap \Lambda(y^3)$, respectively. See Figure 3.2 for a representation of the individual weight set components.

Thus, we can also assume that the entries of a are chosen such that

$$\lambda_1 = a_i \lambda_i, \qquad i = 2, \ldots, p-1, \tag{3.4}$$

holds for all $\lambda \in H_{p,a}$. Let $\lambda^1, \lambda^2 \in \Lambda(y) \cap \Lambda^> \cap H_{p,a}$ and $\lambda = \theta\lambda^1 + (1-\theta)\lambda^2$ for some $\theta \in (0,1)$. Easy calculations yield $\lambda \in \Lambda^> \cap H_{p,a}$. Without loss of generality, let $\lambda_1^1 \leq \lambda_1^2$. By (3.4) we then get $\lambda_i^1 \leq \lambda_i^2$ for $i = 2, \ldots, p-1$, and therefore $\sum_{i=1}^p \lambda_i^1 = \sum_{i=1}^p \lambda_i^2 = 1$ implies $\lambda_p^1 \geq \lambda_p^2$. Since λ is a convex combination of λ^1 and λ^2, we summarize as follows:

$$\lambda_i^1 \leq \lambda_i \leq \lambda_i^2 \text{ for } i = 1, \ldots, p-1, \tag{3.5a}$$

$$\lambda_p^1 \geq \lambda_p \geq \lambda_p^2. \tag{3.5b}$$

Now we define scalars $\varepsilon_2, \varepsilon$ by $\varepsilon_2 := {}^{\lambda_1^2}/_{\lambda_1^1}$ and $\varepsilon := {}^{\lambda_1}/_{\lambda_1^1}$. By (3.5a) we have

$$1 \leq \varepsilon \leq \varepsilon_2. \tag{3.6}$$

Set $\bar{\lambda}^2 := {}^1/_{\varepsilon_2}\lambda^2$ and $\bar{\lambda} := {}^1/_{\varepsilon}\lambda$. Then, it follows $\bar{\lambda}_1^2 = \bar{\lambda}_1 = \lambda_1^1$ and consequently $\bar{\lambda}_i^2 = \lambda_i^1$

and $\bar{\lambda}_i = \lambda_i^1$ holds for $i = 2, \ldots, p-1$. Combining (3.5a) and (3.6) results in

$$\bar{\lambda}_p^2 = \frac{1}{\varepsilon_2}\lambda_p^2 \leq \frac{1}{\varepsilon}\lambda_p^2 \leq \frac{1}{\varepsilon}\lambda_p = \bar{\lambda}_p \text{ and } \bar{\lambda}_p = \frac{1}{\varepsilon}\lambda_p \leq \frac{1}{1}\lambda_p \leq \lambda_p^1.$$

We get $\bar{\lambda} \in \operatorname{conv}\{\lambda^1, \bar{\lambda}^2\}$. Recall that $\alpha\|y\|_\infty^\lambda = \|y\|_\infty^{\alpha\lambda}$ holds for all $y \in \mathcal{Y}$ and all scalars $\alpha > 0$. Hence, as $\lambda^2 \in \Lambda(y)$, we know that y is also optimal for $\Pi^{TS}(\bar{\lambda}^2)$. Since $\bar{\lambda} \in \operatorname{conv}\{\lambda^1, \bar{\lambda}^2\}$, y is optimal for $\Pi^{TS}(\bar{\lambda})$ by Lemma 3, and therefore y is optimal for $\Pi^{TS}(\lambda)$. The claim follows by continuity of $\|y\|_\infty^\lambda$ in λ for all $y \in \mathcal{Y}$. □

3.3 The Intersection of Weight Set Components

In this section, we state results about the intersection of two weight set components. In general, the intersection of two star-shaped sets is not guaranteed to be star-shaped. However, the intersection of two weight set components is indeed star-shaped. To prove this, we first define a (possibly artificial) image.

Definition 6. *Given a subset of weakly ND images, $\bar{Y} = \{y^1, \ldots, y^{\bar{R}}\} \subseteq \mathcal{Y}_{wN}$, we define the* local nadir point $y^N(\bar{Y})$ *by*

$$y_i^N(\bar{Y}) = \max_{r=1,\ldots,\bar{R}} y_i^r \text{ for } i = 1, \ldots, p.$$

We say an image y^r for $r \in \{1, \ldots, \bar{R}\}$ contributes *to the local nadir image if $y_i^r = y_i^N(\bar{Y})$ for some $i \in \{1, \ldots, p\}$.*

In the following, we avoid trivial cases by requiring $|\bar{Y}| \geq 2$. Further, for ease of exposition we assume that the local nadir point exists in Y. It is easy to see then that the local nadir point $y^N(\bar{Y})$ is dominated. Consequently, $\Lambda(y^N(\bar{Y})) \neq \emptyset$ implies $y^N(\bar{Y}) \in \mathcal{Y}_{wN} \setminus \mathcal{Y}_N$.

Definition 7. *We call the kernel weight of $y^N(\bar{Y})$, $\lambda^N(\bar{Y}) := \lambda(y^N(\bar{Y}))$, the* local nadir weight.

Clearly, $\lambda^N(\bar{Y}) \in \Lambda$. Observe that $\|y\|_\infty^{\lambda^N(\bar{Y})} = \max_{i=1,\dots,p} \frac{y_i}{\max_{r=1,\dots,\bar{R}} y_i^r} M \leq M$, with $M = \left(\sum_{j=1}^p \frac{1}{\max_{r=1,\dots,\bar{R}} y_j^r}\right)^{-1}$ for all $y \in \bar{Y}$. Thus, if y contributes to the local nadir image, we have $\|y\|_\infty^{\lambda^N(\bar{Y})} = M$. In particular, the weighted Tchebychev norm of $\lambda^N(\bar{Y})$ coincides for all images contributing to $y^N(\bar{Y})$.

The local nadir point is closely related to the intersection of weight set components, as shown in the following proposition.

Proposition 8. *Let $\bar{Y} \subseteq \mathcal{Y}_{wN}$. Then, $\bigcap_{y \in \bar{Y}} \Lambda(y) = \Lambda(y^N(\bar{Y}))$.*

Proof. Let $\bar{Y} \subseteq \mathcal{Y}_{wN}$ be enumerated as $\bar{Y} = \{y^1, \dots, y^{\bar{R}}\}$. We abbreviate $y^N := y^N(\bar{Y})$. If $\bigcap_{y \in \bar{Y}} \Lambda(y) = \emptyset$, the inclusion '$\subseteq$' holds trivially. Let $\lambda \in \bigcap_{y \in \bar{Y}} \Lambda(y)$. Then, there exists a constant $c > 0$ such that $\|y^r\|_\infty^\lambda = c$ for all $r = 1, \dots, \bar{R}$ and $c \leq \|y'\|_\infty^\lambda$ for all $y' \in \mathcal{Y}$. We show $\|y^N\|_\infty^\lambda = c$. Since $\max_{i=1,\dots,p} \lambda_i y_i^r = c$ for all $r = 1, \dots \bar{R}$, we have $\lambda_i y_i^r \leq c$ for all $i = 1, \dots, p$ and for all $r = 1, \dots, \bar{R}$. Thus, $\max_{r=1,\dots,\bar{R}} \lambda_i y_i^r \leq c$ and therefore $\lambda_i y_i^N = \lambda_i \cdot \max_{r=1,\dots,\bar{R}} y_i^r \leq c$ for all $i = 1, \dots, p$. Consequently, $\|y^N\|_\infty^\lambda = c$, and so $\lambda \in \Lambda(y^N)$.

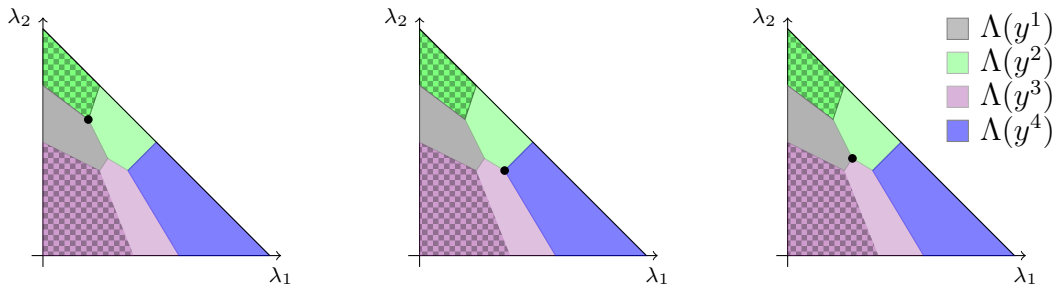
The other direction follows by $y^r \leqq y^N$ for all $r = 1, \dots, \bar{R}$ due to the definition of the local nadir weight. □

We can elaborate on the basic result of Proposition 8 in order to prove more insightful characteristics of the intersection.

Corollary 3. *Let $\bar{Y} \subseteq \mathcal{Y}_{wN}$. Then,*

1. *If $\lambda^N(\bar{Y}) \notin \bigcap_{y \in \bar{Y}} \Lambda(y)$, then $\bigcap_{y \in \bar{Y}} \Lambda(y) = \emptyset$.*
2. *The intersection $\bigcap_{y \in \bar{Y}} \Lambda(y)$ is a star-shaped set with $\lambda^N(\bar{Y})$ in its kernel.*
3. *For $k \in \{1, \dots, p\}$ and $a_i > 0$, $i \neq k$, the intersection $\bigcap_{y \in \bar{Y}} \Lambda(y) \cap H_{k,a}$ is convex.*

Proof. If $\lambda^N(\bar{Y}) \notin \bigcap_{y \in \bar{Y}} \Lambda(y)$, we have $\lambda^N(\bar{Y}) \notin \Lambda(y^N)$ by Proposition 8. Then, by the proof of Proposition 3, we have $y^N \notin \mathcal{Y}_{wN}$ and therefore the weight set component of the



(a) (b) (c)

Figure 3.4: The local nadir weights (a) $\lambda^N(\{y^1, y^2\})$, (b) $\lambda^N(\{y^2, y^3\}) = \lambda^N(\{y^3, y^4\}) = \lambda^N(\{y^2, y^3, y^4\})$ and (c) $\lambda^N(\{y^1, y^2, y^3\})$ for Example 6. The intersection sets of weight set components are always star-shaped sets. In particular, the corresponding local nadir weight is contained in the kernel.

local nadir point has to be empty. Thus we get again by Proposition 8 that $\bigcap_{y \in \bar{Y}} \Lambda(y) = \emptyset$. This gives (i). If $\bigcap_{y \in \bar{Y}} \Lambda(y) = \emptyset$, there is nothing left to prove. Otherwise, statements (ii) and (iii) follow by Corollary 2 and Proposition 7, respectively, as well as Proposition 8 since $\Lambda(y^N) \neq \emptyset$ and thus $y^N \in \mathcal{Y}_{wN}$. □

3.4 A Polytopal Subdivision of the Weight Set Components

In the following, we construct a representation of the weight set components as the union of polytopes. For this purpose, let the ND set $\mathcal{Y}_N$ be enumerated as $\mathcal{Y}_N = \{y^1, \dots, y^R\}$. Considering only the image y^1, recall its ith dimensional weight set component for $i \in \{1, \dots, p\}$ (see Definition 5):

$$\Lambda(y^1, i) = \{\lambda \in \Lambda(y^1) : \lambda_i y_i^1 \geq \lambda_k y_k^1 \text{ for all } k = 1, \dots, p\}.$$

That is, the inequalities $\lambda_i y_i^1 \geq \lambda_k y_k^1$, $k \neq i$, subdivide the weight set Λ into polytopes such that we can compute the weights yielding $\|y^1\|_\infty^\lambda = \lambda_i y_i^1$ (see Figure 3.6a). For all other images in $\mathcal{Y}_N$, similar inequalities can be defined: Given an R-tuple $(i_1, \dots, i_R) \in$

$\{1,\dots,p\}^R$, we define

$$P_{(i_1,\dots,i_R)} := \left\{ \lambda \in \Lambda : \begin{array}{ll} \lambda_{i_1} y^1_{i_1} \geq \lambda_k y^1_k & \text{for } k \neq i_1, \\ \lambda_{i_2} y^2_{i_2} \geq \lambda_k y^2_k & \text{for } k \neq i_2, \\ \vdots & \\ \lambda_{i_R} y^R_{i_R} \geq \lambda_k y^R_k & \text{for } k \neq i_R, \end{array} \right\}. \tag{3.7}$$

Obviously, each set $P_{(i_1,\dots,i_R)}$ is a polytope and the family of all polytopes subdivides the weight set. Moreover, we get

$$\Lambda(y^1) \cap P_{(i_1,\dots,i_R)} = \{\lambda \in \Lambda : \lambda_{i_1} y^1_{i_1} \leq \lambda_{i_r} y^r_{i_r} \text{ for all } r = 2,\dots,R\} \cap P_{(i_1,\dots,i_R)}. \tag{3.8}$$

with Proposition 4. Thus, $\Lambda(y^1) \cap P_{(i_1,\dots,i_R)}$ is again a polytope. We collect the polytopes defined analogously to (3.8) for all $y^r \in \mathcal{Y}_N$ in

$$\tilde{\mathcal{C}}(y^r) := \left\{ \Lambda(y^r) \cap P_{(i_1,\dots,i_R)} : (i_1,\dots,i_R) \in \{1,\dots,p\}^R \right\}. \tag{3.9}$$

Observe that many of these polytopes in $\bigcup_{r=1,\dots,R} \tilde{\mathcal{C}}(y^r)$ are empty: If $y^r_i < y^s_i$ holds for two images $y^r, y^s \in \mathcal{Y}_N$ and an index i, all polytopes $\Lambda(y^s) \cap P_{(i_1,\dots,i_R)}$ with $i_r = i_s = i$ are empty. Indeed, we can find an upper bound on the number of nonempty polytopes which is polynomial in $|\mathcal{Y}_N|$.

Lemma 4. *For fixed p, the number of full-dimensional polytopes in $\bigcup_{y \in \mathcal{Y}_N} \mathcal{C}(y)$ is asymptotically bounded by a polynomial in $|\mathcal{Y}_N|$.*

Proof. Let $R = |\mathcal{Y}_N|$. Clearly, the number of full-dimensional polytopes $P_{(i_1,\dots,i_R)}$ is bounded by the number of full-dimensional polytopes induced by the affine subspaces $G(y^r, i, j) = \{\lambda \in \Lambda : \lambda_i y^r_i = \lambda_j y^r_j\}$, see Figure 3.5. This number is bounded by $(R+1)^p$. Furthermore, $\Lambda(y) \cap P_{(i_1,\dots,i_R)}$ is nonempty if and only if we have $\lambda_i y_i \leq \lambda_i y'_i$ for all images y' such that $\lambda_i y'_i \geq \lambda_k y'_k$ for all $\lambda \in P_{(i_1,\dots,i_R)}$. Therefore, a polytope $P_{(i_1,\dots,i_R)}$ will be split

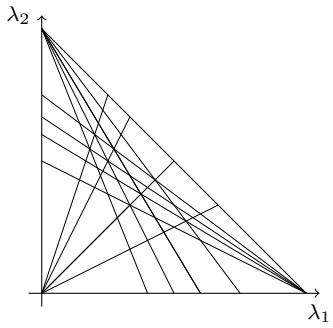


Figure 3.5: The grid for $p = 3$ which induced by affine subspaces of the form $G(y^r, i, j)$.

in at most $R \cdot p$ further full-dimensional polytopes. In summary, we get an upper bound of $R \cdot p \cdot (R+1)^p = \mathcal{O}(R^{p+1})$ polytopes. □

Note that for $p = 3$, the number of full-dimensional polytopes coincides with the number of facets of the plain graph induced by the grid. By induction over R, one can count the nodes and the implied edges. Then, Euler's formula for planar graphs implies a better upper bound of $3 \cdot (2R + 1/2(R+1)R + 1) = \mathcal{O}(R^2)$. We state some properties of the families in (3.9).

Proposition 9. *Let $\mathcal{Y}_N = \{y^1, \dots, y^R\}$. For a ND image $y^r \in \mathcal{Y}_N$. Then the following hold:*

1. *The intersection set $\Lambda(y^r) \cap P_{(i_1,\dots,i_R)}$ is a polytope for all R-tuples $(i_1, \dots, i_R) \in \{1, \dots, p\}^R$.*

2. *We have $\bigcup_{P \in \tilde{\mathcal{C}}(y^r)} P = \Lambda(y^r)$.*

3. *Let $y^r, y^s \in \mathcal{Y}_N$ be two ND images and $P_{(i_1,\dots,i_R)}, P_{(j_1,\dots,j_R)}$ be two polytopes. Then, $\big(P_{(i_1,\dots,i_R)} \cap \Lambda(y^r)\big) \cap \big(P_{(j_1,\dots,j_R)} \cap \Lambda(y^s)\big)$ is a face of both $P_{(i_1,\dots,i_R)} \cap \Lambda(y^r)$ and $P_{(j_1,\dots,j_R)} \cap \Lambda(y^s)$. In particular, it's inclusion-wise maximal.*

Proof. The statements (i) and (ii) are already proven. Without loss of generality, let $\mathcal{Y}_N$ be

enumerated such that $y^r = y^1$ and $y^s = y^2$. We define

$$\begin{aligned} H &:= \{\lambda \in \mathbb{R}^p_{\geq} : \lambda_{i_1} y^1_{i_1} = \lambda_{j_1} y^1_{j_1}, \ldots, \lambda_{i_R} y^R_{i_R} = \lambda_{j_R} y^R_{j_R}, \lambda_{i_1} y^1_{i_1} = \lambda_{j_2} y^2_{j_2}\} \\ &= \{\lambda \in \mathbb{R}^p_{\geq} : \lambda^1_{i_1} y^1_{i_1} = \lambda_{j_2} y^2_{j_2}\} \cap \left(\bigcap_{r=1}^{R} \{\lambda \in \mathbb{R}^p_{\geq} : \lambda_{i_r} y^r_{i_r} = \lambda_{j_r} y^r_{j_r}\} \right). \end{aligned}$$

On the one hand,

$$\{\lambda \in \mathbb{R}^p_{\geq} : \lambda^1_{i_1} y^1_{i_1} \leq \lambda_{j_2} y^2_{j_2}\} \cap \left(\bigcap_{r=1}^{R} \{\lambda \in \mathbb{R}^p_{\geq} : \lambda_{i_r} y^r_{i_r} \geq \lambda_{j_r} y^r_{j_r}\} \right)$$

is an intersection of valid inequalities for $P_{(i_1,\ldots,i_R)} \cap \Lambda(y^1)$, as shown in (3.8). That is, these inequalities define a face $F^1 = P_{(i_1,\ldots,i_R)} \cap \Lambda(y^1) \cap H$, if it is nonempty. On the other hand,

$$\{\lambda \in \mathbb{R}^p_{\geq} : \lambda^1_{i_1} y^1_{i_1} \geq \lambda_{j_2} y^2_{j_2}\} \cap \left(\bigcap_{r=1}^{R} \{\lambda \in \mathbb{R}^p_{\geq} : \lambda_{i_r} y^r_{i_r} \leq \lambda_{j_r} y^r_{j_r}\} \right)$$

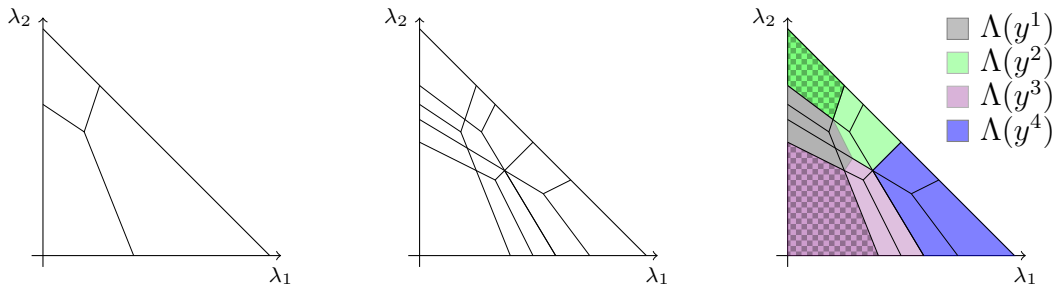
is an intersection of valid inequalities for $P_{(j_1,\ldots,j_R)} \cap \Lambda(y^2)$. Analogously, $F^2 = P_{(j_1,\ldots,j_R)} \cap \Lambda(y^2) \cap H$ is a face of $P_{(j_1,\ldots,j_R)} \cap \Lambda(y^2)$. By definition, $\lambda \in \big(P_{(i_1,\ldots,i_R)} \cap \Lambda(y^1)\big) \cap \big(P_{(j_1,\ldots,j_R)} \cap \Lambda(y^2)\big)$ satisfies for $r = 1, \ldots, R$:

$$\begin{aligned} \lambda_{i_r} y^r_{i_r} &\geq \lambda_k y^r_k \text{ for } k \neq i_r, \\ \lambda_{j_r} y^r_{j_r} &\geq \lambda_k y^r_k \text{ for } k \neq j_r. \end{aligned}$$

This implies $\lambda_{i_r} y^r_{i_r} = \lambda_{j_r} y^r_{j_r}$. Since also $\lambda_{i_1} y^1_{i_1} \leq \lambda_{i_2} y^2_{i_2}$ and $\lambda_{j_2} y^2_{j_2} \leq \lambda_{j_1} y^1_{j_1}$ hold, we get $\lambda_{i_1} y^1_{i_1} \leq \lambda_{i_2} y^2_{i_2} = \lambda_{j_2} y^1_{j_2} \leq \lambda_{j_1} y^2_{j_1} = \lambda_{i_1} y^1_{i_1}$. Thus, we have equality $\lambda_{i_1} y^1_{i_1} = \lambda_{j_2} y^2_{j_2}$. It follows that

$$\lambda \in F^1 \Leftrightarrow \lambda \in \big(P_{(i_1,\ldots,i_R)} \cap \Lambda(y^1)\big) \cap \big(P_{(j_1,\ldots,j_R)} \cap \Lambda(y^2)\big) \Leftrightarrow \lambda \in F^2$$

holds. Consequently, $F^1 = F^2 = \big(P_{(i_1,\ldots,i_R)} \cap \Lambda(y^1)\big) \cap \big(P_{(j_1,\ldots,j_R)} \cap \Lambda(y^2)\big)$. Maximality



(a) (b) (c)

Figure 3.6: The construction of the polytopal subdivision of the weight set Λ ((a) and (b)) according to (3.7) and the polytopal subdivision of the weight sets components of Example 6 (c). See Figure 3.2 for a representation of the individual weight set components.

follow also by the latter equalities. □

Since the statement of Proposition 93 remains true if y^r is chosen to be equal y^s, we can conclude that the polytopes in $\tilde{\mathcal{C}}(y^r)$ themselves always intersect in a common face. By adding all faces of the polytopes in $\tilde{\mathcal{C}}(y^r)$, Proposition 92 implies that we get a polytopal subdivision of the weight set component. Thus, we define:

Definition 8. *For $y \in \mathcal{Y}_N$, the* weight set complex *of y with respect to the weighted Tchebychev scalarization is defined by*

$$\mathcal{C}(y) := \{F : \textit{ there exists a polytope } P \in \tilde{\mathcal{C}}(y) \textit{ such that } F \textit{ is a face of } P\}.$$

$\mathcal{C}(y)$ collects all faces of the polytopes in $\tilde{\mathcal{C}}(y)$, including the polytopes themselves and the empty set, which are the trivial faces.

Corollary 4. *Let $y \in \mathcal{Y}_N$. Then, $\mathcal{C}(y)$ is a polytopal complex such that $\bigcup_{P \in \mathcal{C}(y)} P = \Lambda(y)$ holds.*

Figure 3.6c shows the subdivision for Example 6. Proposition 5 can directly be adapted such that $\Lambda = \bigcup_{y \in \mathcal{Y}_N} \bigcup_{P \in \mathcal{C}(y)} P$ is a sufficient and necessary condition.

Remark 1. *Analogously to the construction of the weight set complex, we get a polytopal subdivision of the dimensional weight set components if we collect all faces of the polytopes*

induced by the collection

$$\tilde{\mathcal{C}}(y^r, i) := \{\Lambda(y^r) \cap P_{(i_1,\dots,i_{r-1},i,i_{r+1},\dots,i_R)} : \\ (i_1, \dots, i_{r-1}, i_{r+1}, \dots, i_R) \in \{1, \dots, p\}^{R-1}\}.$$

We thus define:

Definition 9. *For $y \in \mathcal{Y}_{wN}$, $i \in \{1, \dots, p\}$, we call*

$$\mathcal{C}(y, i) := \{F : \text{ there exists a polytope } P \in \tilde{\mathcal{C}}(y, i) \text{ such that } F \text{ is a face of } P\}.$$

*the i*th dimensional weight set complex.

By construction, $\mathcal{C}(y, i)$ is indeed a subcomplex of $\mathcal{C}(y)$.
Taking images in $\mathcal{Y}_{wN} \setminus \mathcal{Y}_N$ into account, we need to refine the construction of the polytopal subdivision. This can be done by defining (3.7) and (3.8) based on an enumeration of $\mathcal{Y}_{wN}$. Then, the following holds analogously.

Corollary 5. *For $y \in \mathcal{Y}_{wN}$, there exists a polytopal subdivision of $\Lambda(y)$.*

Remark 2. *Due to the construction of these polytopal subdivisions, the intersection of weight set components induces a polytopal subdivision that uses polytopes of both subdivisions only. That is $\Lambda(y^1) \cap \Lambda(y^2) = \bigcup_{P \in \mathcal{C}(y^1) \cap \mathcal{C}(y^2)} P$. In particular, $\mathcal{C}(y^1) \cap \mathcal{C}(y^2)$ itself is also a polytopal complex. Thus, we can compare weight set components based on the polytopes in the polytopal subdivision.*
Similarly, the union of two weight set complexes is a polytopal subdivision of the union of the corresponding weight set components.

We conclude this chapter with a link of the polytopal subdivision to the line convexity in Proposition 7.

Remark 3. *Consider a polytope*

$$P = \Lambda(y^1) \cap P_{(i_1,\dots,i_R)} = \{\lambda \in \Lambda : \lambda_{i_1} y^1_{i_1} \leq \lambda_{i_r} y^r_{i_r} \text{ for all } r = 2,\dots,R\} \cap P_{(i_1,\dots,i_R)}$$

(see equation (3.8)). Observe that the inequalities defining P are all of the form $\lambda_i y_i^{r_1} \leq \lambda_j y_j^{r_2}$ or $\lambda_i y_i^{r_1} \geq \lambda_j y_j^{r_2}$ for some $i, j \in \{1,\dots,p\}, r^1, r^2 \in \{1,\dots,R\}$. In particular, in each set of such inequalities at most one is non-redundant. Conversely, in set of inequalities, one inequality is face-defining. This implies two further properties: First, the number of facets of P is bounded by $2\binom{p}{2}$, since for each combination of objectives i, j at most two can define a facet. Second, all proper faces of P are embedded in hyperplanes defined by a set of equalities of the form $\lambda_i y_i^{r_1} = \lambda_j y_j^{r_2}$, where the images y^{r_1} and y^{r_2} for some $i, j \in \{1,\dots,p\}, r^1, r^2 \in \{1,\dots,R\}$. That is, they are contained in a hyperplane intersection with an extreme point of the weight set Λ itself. In other words, for some fixed $k \in \{1,\dots,p\}$, we find a set of vectors $A \subseteq \mathbb{R}^p_>$ such that a face F of P satisfies $F \subseteq \bigcup_{a\in A} H_{k,a} \cap \Lambda \subseteq \operatorname{aff}(F)$, see (3.3) for the definition of the planes $H_{k,a}$.

In particular, for $p = 3$, edges of P are embedded in exactly one line of the form $H_{k,a} \cap \Lambda$. Therefore, the boundary of a weight set component can be described by a sequence of $Q > 2$ line segments $[\lambda^1, \lambda^2], [\lambda^2, \lambda^3], \dots, [\lambda^Q, \lambda^1]$ such that $[\lambda^q, \lambda^{q+1 \bmod Q}] \subseteq H_{k^q,a^q}$ for some $k^q \in \{1,2,3\}$ and $a^q \in \mathbb{R}^3_>$, $q = 1,\dots,Q$.

3.5 The Dimension of the Weight Set Components

In this section, we analyze the dimension of the weight set components. First, we define the dimension with respect to the associated polytopal complex. Recall that a polytopal complex is defined via a *finite* set of polytopes, and for all weight set components there exists a polytopal subdivision, see Corollary 4.

Definition 10. *For $y \in \mathcal{Y}$, we define the dimension of its weight set component by* $\dim(\Lambda(y)) := \dim(\mathcal{C}(y))$.

(Note that the dimension of the dimensional weight set components as well as the intersections or unions of weight set components can be defined analogously by Remarks 1 and 2.) In the following, we distinguish between images of $\mathcal{Y}_N$ and $\mathcal{Y}_{wN} \setminus \mathcal{Y}_N$.

We first deal with ND images. Due to the finite number of polytopes in $\mathcal{C}(y)$, Corollary 1 immediately implies that the dimension of the corresponding weight set complexes $\mathcal{C}(y)$ must be equal to $p-1$. This gives:

Corollary 6. *Let $y \in \mathcal{Y}_N$. Then,* $\dim(\Lambda(y)) = p-1$.

We refine the analysis to the dimensional weight set components. In particular, we also take a look at the intersection of these components. The proof of the following lemma is rather technical and can be found in the appendix.

Lemma 5. *Let $y \in \mathcal{Y}_{wN}$ and $I \subseteq \{1,\dots,p\}$. If $\lambda \in \bigcap\limits_{i\in I} \Lambda(y,i)$ and $\lambda \notin \bigcup\limits_{j\notin I} \Lambda(y,j)$ holds, there exists a polytope P such that*

1. $\lambda \in P$,
2. $P \subseteq \bigcap\limits_{i\in I} \Lambda(y,i)$,
3. $\dim(P) = p - |I|$.

From Lemma 5, we can deduce the dimension of the dimensional weight set complexes. The proof is mainly based on Corollary 1 and the fact that the dimensional components evenly decompose the ε-ball.

Proposition 10. *Let $y \in \mathcal{Y}_N$ and $\emptyset \neq I \subseteq \{1,\dots,p\}$. Then it holds* $\dim\left(\bigcap_{i\in I} \Lambda(y,i)\right) = p - |I|$. *In particular, we have* $\dim\left(\Lambda(y,i)\right) = p-1$.

Proof. If $I = \{1,\dots,p\}$ holds, the claim follows from Proposition 61. Thus, let $\emptyset \neq I \subsetneq \{1,\dots,p\}$.

By Corollary 1, there exists an $\varepsilon > 0$ such that $B_\varepsilon(\lambda(y)) \subseteq \Lambda(y)$. Set

$$\lambda_i := \begin{cases} (1+\frac{\varepsilon}{2})\lambda_i(y), & \text{if } i \in I, \\ \lambda_i(y) - \frac{\varepsilon}{2}\frac{1}{p-|I|}\sum_{j\in I}\lambda_j(y), & \text{if } i \notin I. \end{cases}$$

We show that λ fulfills the requirements of Lemma 5. It is easy to see that $\lambda \in \Lambda$ and in particular $\lambda \in B_\varepsilon(\lambda(y)) \subseteq \Lambda(y)$ with

$$\begin{aligned} \lambda_i y_i = (1+\frac{\varepsilon}{2})\lambda_i(y)y_i = (1+\frac{\varepsilon}{2})\lambda_k(y)y_k = \lambda_k y_k \qquad & \text{for all } k \in I, \\ \lambda_i y_i = (1+\frac{\varepsilon}{2})\lambda_i(y)y_i = (1+\frac{\varepsilon}{2})\lambda_k(y)y_k > \lambda_k(y)y_k > \lambda_k y_k \qquad & \text{for all } k \notin I. \end{aligned}$$

By Lemma 5, we have $\dim\left(\bigcap_{i\in I}\Lambda(y,i)\right) \geq p - |I|$. On the other hand, $\bigcap_{i\in I}\Lambda(y,i) \subseteq \{\lambda \in \Lambda : \lambda_i y_i = \lambda_j y_j \text{ for } i,j \in I\} =: P$. Consequently, we have $\dim\left(\bigcap_{i\in I}\mathcal{C}(y,i)\right) \leq \dim(P) \leq p - |I|$. Thus, equality holds and the proof is complete. □ □

Since a weakly ND image $y^w \in \mathcal{Y}_{wN} \setminus \mathcal{Y}_N$ is optimal for the scalarized problem $\Pi^{TS}(\lambda(y^w))$ with the central weight of y^w, the corresponding weight set component $\Lambda(y^w)$ is not empty. We have already seen that these sets also have a polytopal subdivision and fulfill the convexity properties stated in Proposition 7 and Corollary 2. Regarding their dimension, Proposition 10 does not hold due to the fact that there is no $\varepsilon > 0$ such that $B_\varepsilon(\lambda(y^w)) \subseteq \Lambda(y^w)$ holds. The next example shows this for two objectives.

Example 7. *Set* $\tilde{Y} = \{\tilde{y}^1 = (4,4)^\top,\ \tilde{y}^2 = (4,2)^\top\}$*, only. Clearly,* $\tilde{y}^2 \in \mathcal{Y}_N$ *and* $\tilde{y}^1 \in \tilde{Y}_{wN} \setminus \tilde{Y}_N$*. For the kernel vertex* $\lambda(\tilde{y}^1) = (1/2, 1/2)^\top$*, we have* $\|\tilde{y}^1\|_\infty^{\lambda(\tilde{y}^1)} = 2 = \|\tilde{y}^2\|_\infty^{\lambda(\tilde{y}^1)}$*. However, for any scalar* $\varepsilon > 0$ *and* $\lambda = (\lambda_1(\tilde{y}^1)+\varepsilon, \lambda_2(\tilde{y}^1)-\varepsilon)^\top$*, we get* $\|\tilde{y}^1\|_\infty^\lambda = 2+4\varepsilon > 1+2\varepsilon = \lambda_1\tilde{y}_1^2$ *as well as* $\|\tilde{y}^1\|_\infty^\lambda = 2+4\varepsilon > 2-4\varepsilon = \lambda_2\tilde{y}_2^2$*. Hence,* $\|\tilde{y}^1\|_\infty^\lambda > \|\tilde{y}^2\|_\infty^\lambda$*. Nevertheless, we can conclude* $\Lambda(\tilde{y}^1) = \Lambda(\tilde{y}^2) \cap \{\lambda \in \Lambda : \lambda_1 \leq \lambda_1(\tilde{y}^1)\}$*, since the weighted Tchebychev norm values are attained in the second objective for both* $\bar{y}^1$ *and* $\bar{y}^2$ *and thus*

coincide. Consequently, both weight set components have dimension $p - 1 = 1$.

This raises the question whether the claim of Proposition 10 holds for weakly ND but not ND images, too. This is not the case.

Example 8 (Example 7 cont.)**.** *We add another image to the image set:*

$$\tilde{Y}^* = \tilde{Y} \cup \{\tilde{y}^3 = (2, 4)^\top\}.$$

Analogously we get $\|\tilde{y}^1\|_\infty^\lambda > \|\tilde{y}^3\|_\infty^\lambda$, $\lambda = (\lambda_1(\tilde{y}^1) - \varepsilon, \lambda_2(\tilde{y}^1) + \varepsilon)^\top$, *for any* $\varepsilon > 0$. *Thus,* $\Lambda(\tilde{y}^1) = \{\lambda(\tilde{y}^1)\}$ *and it is* $\dim(\Lambda(\tilde{y}^1)) = 0$.

How can we characterize the dimension of the weight set components of images $y^w \in \mathcal{Y}_{wN} \setminus \mathcal{Y}_N$? We will conclude that this depends on the images dominating y^w. If $y^w \in \mathcal{Y}_{wN} \setminus \mathcal{Y}_N$, then there exists a $y \in \mathcal{Y}_N$ such that $y_i \leq y_i^w$ for all $i = 1, \ldots, p$. That is, $\Lambda(y^w, i) \subseteq \Lambda(y, i)$ for all i satisfying $y_i^w = y_i$.

Lemma 6. *Let* $y^w \in \mathcal{Y}_{wN} \setminus \mathcal{Y}_N$ *and* $y \in \mathcal{Y}_N$ *such that* $y \leq y^w$ *and* $y_{i_1} < y_{i_1}^w, \ldots, y_{i_l} < y_{i_l}^w$ *for indices* $\{i_1, \ldots, i_l\} \subseteq \{1, \ldots, p\}$. *Then,* $\Lambda(y^w) \setminus \left(\bigcup_{i \notin \{i_1, \ldots, i_l\}} \Lambda(y^w, i)\right) = \emptyset$.

Proof. Without loss of generality, let $i_1 = 1, \ldots, i_l = l$. Assume, there is a weight $\lambda \in \Lambda(y^w) \setminus \left(\bigcup_{i=l+1}^p \Lambda(y^w, i)\right)$. Then,

$$\lambda \in \left(\bigcup_{i=1}^{l} \Lambda(y^w, i)\right) \setminus \left(\bigcup_{i=l+1}^{p} \Lambda(y^w, i)\right)$$

and, therefore, we get with $y \in \mathcal{Y}_N \cap (\{y^w\} - \mathbb{R}^p_{\leqq})$

$$\lambda_i y_i < \lambda_i y_i^w \qquad i = 1, \ldots, l, \tag{3.10}$$

$$\lambda_k y_k \leq \lambda_k y_k^w \qquad k = l+1, \ldots, p, \tag{3.11}$$

$$\text{there is some } i \in \{1, \ldots, l\} \text{ such that } \lambda_k y_k^w < \lambda_i y_i^w \qquad k = l+1, \ldots, p. \tag{3.12}$$

If $\|y\|_\infty^\lambda = \lambda_i y_i$ for an $i \in \{1,\dots,p\}$, we have with (3.10) $\lambda_i y_i < \lambda_i y_i^w \leq \|y^w\|_\infty^\lambda$. If $\|y\|_\infty^\lambda = \lambda_k y_k$ for a $k \in \{l+1,\dots,p\}$, we have with (3.11) and (3.12) $\lambda_k y_k \leq \lambda_k y_k^w < \lambda_i y_i^w \leq \|y^w\|_\infty^\lambda$ for an $i \in \{1,\dots,l\}$. Both are contradictions to $\lambda \in \Lambda(y^w)$. □ □

Thus, the dimension of the weight set components of weakly ND but dominated images highly depends on the number of images that dominate y^w and on which components those images are strictly better.

Proposition 11. *Let $y^w \in \mathcal{Y}_{wN} \backslash \mathcal{Y}_N$. If for all $I \subseteq \{1,\dots,p\}$, $|I| = l$, there exists an image $y \in \mathcal{Y}_N \cap \left(\{y^w\} - \mathbb{R}^p_{\leqq}\right)$ such that $y_i < y_i^w$ holds for all $i \in I$, then* $\dim(\Lambda(y^w)) \leq p-1-l$.

Proof. This follows from Lemma 6. □ □

That is, the dimension of the weight set components of images in $\mathcal{Y}_{wN} \setminus \mathcal{Y}_N$ is determined by the maximal cardinality of a set that satisfies the assumptions of Proposition 11.

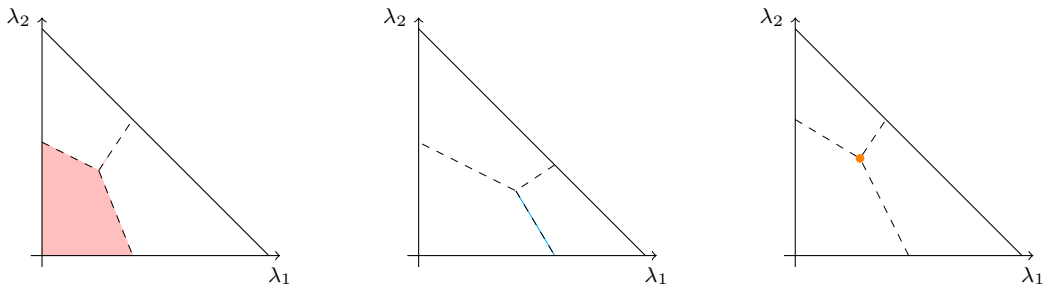
Example 9 (Example 6 cont.)**.** *We augment the set $\mathcal{Y}$ to*

$$\mathcal{Y}' = \mathcal{Y} \cup \{y^5 = (3,2,2)^\top, y^6 = (2,3,3)^\top, y^7 = (3,2,3)^\top\}.$$

Then, y^5, y^6 and y^7 are weakly ND, but dominated images. For y^5 we have $y_3^5 \leq y_3$ for all $y \in \mathcal{Y}$. Thus, Proposition 11 gives $\dim(\Lambda(y^5)) > 3-1-1 = 1$ *and, therefore, this weight set component is full-dimensional.*

The ND images dominating y^6 are $y^2 = (2,1,3)^\top, y^3 = (2,2,2)^\top$ and $y^4 = (1,2,3)^\top$. Thus, for all indices i there exists an image $y \in \mathcal{Y}_N \cap \left(\{y^6\} - \mathbb{R}^p_{\leqq}\right)$ such that $y_i < y_i^6$. Proposition 11 gives $\dim(\Lambda(y^6)) \leq 3-1-1 = 1$. *However, for the index pair $(1,3)$ there does not exist an image satisfying the assumptions of Proposition 11. Thus,* $\dim(\Lambda(y^6)) > 3-1-2 = 0$ *and we get* $\dim(\Lambda(y^6)) = 1$.

For the image y^7, there exists for all pairs of indices an image in $\mathcal{Y}_N \cap \left(\{y^7\} - \mathbb{R}^p_{\leqq}\right)$ satisfying the assumptions of Proposition 11. Thus, $\dim(\Lambda(y^7)) \leq 3-1-2 = 0$ *and,*



(a) $\Lambda(y^5)$ (b) $\Lambda(y^6)$ (c) $\Lambda(y^7)$

Figure 3.7: The weight set components $\Lambda(y^r)$, $r = 5, 6, 7$, of Example 9. Remark that $y^r \in Y_{wN} \setminus Y_N$ for $r = 5, 6, 7$, and thus the interior of at least one dimensional weight set component is empty.

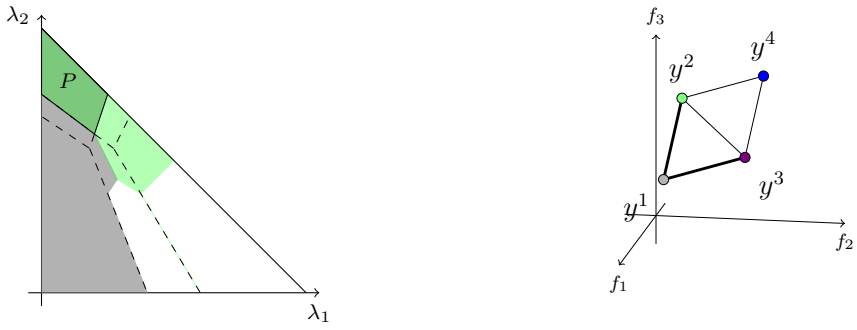
therefore, the weight set component consists of the kernel vertex only.

As a further consequence, we get a characterization of ND images.

Corollary 7. *Let $\int(\Lambda(y, i))$ denotes the set of all weights $\lambda \in \Lambda(y, i)$ such that there exists a scalar $\varepsilon > 0$ with $B_\varepsilon(\lambda) \subseteq \Lambda(y, i)\}$. An image $y \in \mathcal{Y}$ is ND if and only if $\int(\Lambda(y, i)) \neq \emptyset$ for all $i = 1, \ldots, p$.*

The Intersection of Weight Set Components We have seen in section 3.3 that the intersection of weight set components $\bigcap_{y \in \bar{Y}} \Lambda(y)$ for $\bar{Y} \subseteq \mathcal{Y}_N$ is exactly determined by the weight set component of the (dominated) local nadir point $y^N(\bar{Y})$. Thus, the dimension of the intersection sets is characterized by Proposition 11. Remark that a nonempty intersection implies that all images in $\bar{Y}$ contribute to $y^N(\bar{Y})$, and $\bar{Y} = \mathcal{Y}_N \cap \{y^N(\bar{Y})\} - \mathbb{R}^p_{\leqq}$. Thus, if all images in $\bar{Y}$ coincide in at least one component i, we have $\dim(\bigcap_{y \in \bar{Y}} \Lambda(y)) = p - 1$, that is, they share at least one full-dimensional polytope in their weight set complexes (in particular in their ith dimensional weight set component). Notice also that this cannot happen between different dimensional weight set components as $\Lambda(y^1, i) \cap \Lambda(y^2, j) \subseteq \{\lambda \in \Lambda : \lambda_i y^1_i = \lambda_j y^2_j\}$ and the dimension of the latter polytope is $p - 2$.

Considering only two ND images, we can therefore define a concept of (proper) adjacency regarding the weighted Tchebychev scalarization. (Note that proper adjacency aligns with the concept in [23] for the weighted sum scalarization.)



(a) (b)

Figure 3.8: Two weight set complexes can share a full-dimensional polytope P (a). Thus the images are adjacent, but not proper adjacent. The adjacency of the images in the image space is visualized in (b). The bold lines indicate an overlapping of the corresponding weight set components.

Definition 11. *Let two images $y, \bar{y} \in \mathcal{Y}$ be given.*

1. *The (dimensional) weight set components of y and $\bar{y} \in \mathcal{Y}$* overlap *if* $\dim(\Lambda(y,i) \cap \Lambda(\bar{y},i)) = p-1$ *for some* $i \in \{1,\dots,p\}$.

2. *The images are* adjacent *with respect to the weighted Tchebychev scalarization if* $\Lambda(y) \cap \Lambda(\bar{y}) \neq \emptyset$.

3. *The images are* properly adjacent *if* $\dim(\Lambda(y) \cap \Lambda(\bar{y})) = p-2$.

The Local Dimension of Weight Set Components We have seen that the weight set components are star-shaped sets and that a polytopal subdivision with at least one full-dimensional polytope exists. Nevertheless, this does not exclude that there may exist polytopes in $\mathcal{C}(y)$ of dimension less than $p-2$ that are necessary to describe $\Lambda(y)$. This can be seen for example in Figure 3.2. To formalize this, we introduce an adapted notion of dimension.

Definition 12. *Let a weight λ and a set S be given such that $\lambda \in S \subseteq \Lambda$. Further, we denote with $\mathcal{P}$ the set of all polytopes in $\mathbb{R}^p$. Then, the* local dimension of λ with respect to

S is given by

$$\dim(\lambda, S) := \max_{P \in \mathcal{P}, \lambda \in P \subseteq S} \dim(P).$$

From Lemma 5, we can directly deduce:

Corollary 8. *Let $y \in \mathcal{Y}_{wN}$.*

1. *Let $\lambda \in \int (\Lambda(y, i))$. Then,* $\dim(\lambda, \Lambda(y)) = p - 1$.
2. *Let $\emptyset \neq I \subseteq \{1, \ldots, p\}$ and $\lambda \in \bigcap_{i \in I} \Lambda(y, i)$, $\lambda \notin \bigcup_{i \in \{1,\ldots,p\} \setminus I} \Lambda(y, i)$.*
   Then, $\dim(\lambda, \Lambda(y)) \geq p - |I|$.

We analyse the local dimension in a more detailed fashion in the following propositions.

Proposition 12. *Let $\lambda \in \Lambda(y) \cap \Lambda^{>}$ for $y \in \mathcal{Y}_{wN}$. If $\dim(\lambda, \Lambda(y)) = p - \ell$ for some $\ell \in \{2, \ldots, p\}$, then there exists an index set $I \subseteq \{1, \ldots, p\}$ with $|I| \geq \ell$ and a set of ND images $\bar{Y} \subseteq \mathcal{Y}_N \setminus \{y\}$ such that*

1. *$\lambda \in \bigcap_{i \in I} \Lambda(y, i)$ and*
2. *For all index sets $I' \subsetneqq I$ with $|I'| < \ell$ there exists an image $\bar{y} \in \bar{Y}$ such that $\bar{y}_i < y_i$ for all $i \in I'$.*

Proof. Let $\lambda \in \Lambda(y) \cap \Lambda^{>}$ such that $\dim(\lambda, \Lambda(y)) = p - \ell$, $\ell \geq 2$. Choose I such that $\lambda \in \bigcap_{i \in I} \Lambda(y, i)$ and $\lambda \notin \bigcup_{j \notin I} \Lambda(y, j)$. If $|I| < \ell$, Corollary 82 gives

$$\dim(\lambda, \Lambda(y)) \geq p - |I| > p - \ell,$$

which is a contradiction. Thus, $|I| \geq \ell$ and item (i) holds.

Since the local dimension is strictly less than $p - 1$, we have $\lambda \in (\Lambda(y))$. Let $\bar{Y} = \{y^1, \ldots, y^{\bar{R}}\} \subseteq \mathcal{Y}_N \setminus \{y\}$ be all the ND images such that $\lambda \in \bigcap_{r=1}^{\bar{R}} \Lambda(y^r) \cap \Lambda(y)$.

Denote by $y^N := y^N(\bar{Y} \cup \{y\})$ the associated local nadir point. Then $\lambda \in \Lambda(y^N)$. Since $\|y^N\|_\infty^\lambda = \|y\|_\infty^\lambda = \lambda_i y_i \leq \lambda_i y_i^N \leq \|y^N\|_\infty^\lambda$ for all $i \in I$, we have equality and thus $\lambda \in \bigcap_{i \in I} \Lambda(y^N, i)$. In particular we get $y_i = y_i^N$ for all $i \in I$.

By the choice of $\bar{Y}$, we have $\|y^N\|_\infty^\lambda < \|y'\|_\infty^\lambda$ for all $y' \in \mathcal{Y} \setminus \big(\bar{Y} \cup \{y\}\big)$. Thus, there exists an $\varepsilon' > 0$ such that $B_{\varepsilon'}(\lambda) \subseteq \bigcup_{\bar{y} \in \bar{Y}} \Lambda(\bar{y}) \cup \Lambda(y)$ and $B_{\varepsilon'}(\lambda) \cap \Lambda(y') = \emptyset$ for any $y' \in \mathcal{Y} \setminus \big(\bar{Y} \cup \{y\}\big)$. For $\bar{I} \subseteq \{1, \ldots, p\}$, define $\lambda^{\bar{I}}$ by $\lambda_i^{\bar{I}}(\varepsilon) = \lambda_i$ if $i \in \bar{I}$ and $\lambda_i^{\bar{I}}(\varepsilon) = \lambda_i - \varepsilon$ else. Suppose there is a positive $\varepsilon < \min\{\frac{\varepsilon'}{2p}, \frac{1}{2p}\}$ and an index set $\emptyset \neq I' \subsetneqq I$, $|I'| < \ell$ such that $\lambda^{I'}(\varepsilon)$ fulfills $\|y^N\|_\infty^{\lambda^{I'}(\varepsilon)} \leq \|y'\|_\infty^{\lambda^{I'}(\varepsilon)}$ for all $y' \in \mathcal{Y}$.

Take $\bar{I} \subseteq \{1, \ldots, p\}$ such that $I' \subseteq \bar{I}$ holds. Clearly, $\lambda \geq \lambda^{\bar{I}}(\varepsilon) \geq \lambda^{I'}(\varepsilon)$. This gives

$$\|y'\|_\infty^{\lambda^{I'}(\varepsilon)} \leq \|y'\|_\infty^{\lambda^{\bar{I}}(\varepsilon)} \text{ for all } y' \in \mathcal{Y}. \tag{3.13}$$

On the other hand $I' \neq \emptyset$ and therefore we have for an arbitrary $i' \in I'$:

$$\lambda_{i'}^{\bar{I}}(\varepsilon) y_{i'}^N = \lambda_{i'} y_{i'}^N = \|y^N\|_\infty^\lambda \geq \lambda_k y_k^N \geq \lambda_k^{\bar{I}} y_k^N.$$

This gives $\lambda_{i'} y_{i'}^N = \|y^N\|_\infty^{\lambda^{\bar{I}}(\varepsilon)} = \|y^N\|_\infty^{\lambda^{I'}(\varepsilon)} = \|y^N\|_\infty^\lambda$. With (3.13) we get $\|y^N\|_\infty^{\lambda^{\bar{I}}(\varepsilon)} = \|y^N\|_\infty^{\lambda^{I'}(\varepsilon)} \leq \|y'\|_\infty^{\lambda^{I' \cup \bar{I}}(\varepsilon)}$ for all $y' \in \mathcal{Y}$. By construction, $\lambda^{\bar{I}}$ and $\lambda^{\bar{I} \cup \{k\}}$ for an index $k \notin \bar{I}$ satisfy the conditions of Lemma 3. Thus, applying the convexity property of Lemma 3 multiple times in a recursive manner and normalizing all weights yields

$$\operatorname{conv}\left\{ \frac{\lambda^{\bar{I}}(\varepsilon)}{\|\lambda^{\bar{I}}(\varepsilon)\|_1}, I' \subseteq \bar{I} \subseteq \{1, \ldots, p\} \right\} \subseteq \Lambda(y^N), \tag{3.14}$$

see also Figure 3.9 for a visualization of this argument. Clearly, $\lambda = \lambda^{\{1,\ldots,p\}}(\varepsilon)$, and the weights λ and $\big\{\lambda^{\{1,\ldots,p\} \setminus \{k\}} : k \in \{1, \ldots, p\} \setminus I'\big\}$ are affinely independent (compare the proof of Lemma 5 in the appendix). Affine independence is invariant with respect to normalization, and thus $\dim\Big(\operatorname{conv}\Big\{\frac{\lambda^{\bar{I}}(\varepsilon)}{\|\lambda^{\bar{I}}(\varepsilon)\|_1}, I' \subseteq \bar{I} \subseteq \{1, \ldots, p\}\Big\}\Big) = p - |I'| > p - \ell$. With $\Lambda(y^N) \subseteq \Lambda(y)$ we get $\dim(\lambda, \Lambda(y)) \geq \dim(\lambda, \Lambda(y^N)) \geq p - |I'|$, which is a

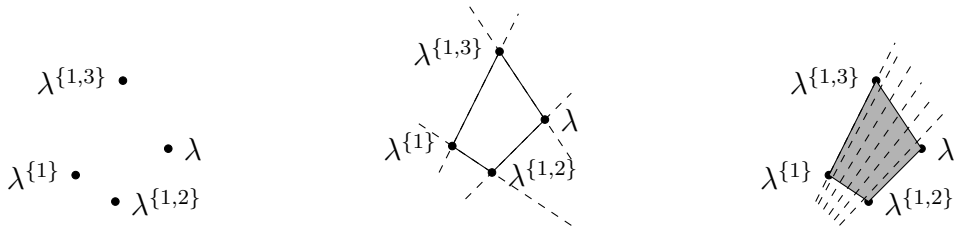


Figure 3.9: Visualization of the argument implying (3.14) for $p = 3$ and $I' = \{1\}$. The normalized weights $\lambda^{\bar{I}} \in \Lambda(y^N)$ for all $I' \subseteq \bar{I} \subseteq \{1, \dots, p\}$ are visualized in (a). The application of the convexity along lines $H_{k,a}$ (dashed lines) yields $[\lambda, \lambda^{\{1,2\}}], [\lambda, \lambda^{\{1,3\}}], [\lambda^{\{1\}}, \lambda^{\{1,2\}}], [\lambda^{\{1\}}, \lambda^{\{1,3\}}] \subseteq \Lambda(y^N)$ (b). The application of the convexity along (infinitely many) lines $H_{k,a}$ again yields $\operatorname{conv}(\{\lambda, \lambda^{\{1\}}, \lambda^{\{1,2\}}, \lambda^{\{1,3\}}\}) \subseteq \Lambda(y^N)$ (c).

contradiction.

Therefore, there exists an image $y' \in \mathcal{Y}$ for all $0 < \varepsilon < \min\{\frac{\varepsilon'}{2p}, \frac{1}{2p}\}$ and an index set $\emptyset \neq I' \subsetneqq I$ such that $\|y'\|_\infty^{\lambda^{I'}(\varepsilon)} < \|y^N\|_\infty^{\lambda^{I'}(\varepsilon)}$. Straight forward calculations show ${\lambda^{I'}(\varepsilon)}/{\|\lambda^{I'}(\varepsilon)\|_1} \in B_{\varepsilon'}(\lambda)$. Thus, $\|y^N\|_\infty^{\lambda^{I'}(\varepsilon)} < \|y'\|_\infty^{\lambda^{I'}(\varepsilon)}$ for any $y' \in \mathcal{Y}_N \setminus \big(\bar{Y} \cup \{y\}\big)$. Since also $\|y^N\|_\infty^{\lambda^{I'}(\varepsilon)} = \|y\|_\infty^{\lambda^{I'}(\varepsilon)}$, there exists a $\bar{y} \in \bar{Y}$ such that $\|\bar{y}\|_\infty^{\lambda^{I'}(\varepsilon)} < \|y^N\|_\infty^{\lambda^{I'}(\varepsilon)}$. In particular, we get for all $i \in I'$ that

$$\lambda_i^{I'}(\varepsilon)\bar{y}_i \leq \|\bar{y}\|_\infty^{\lambda^{I'}(\varepsilon)} < \|y^N\|_\infty^{\lambda^{I'}(\varepsilon)} = \lambda_i^{I'}(\varepsilon) y_i^N = \lambda_i^{I'}(\varepsilon) y_i,$$

which implies $\bar{y}_i < y_i$. □

Proposition 13. *Let $y \in \mathcal{Y}_{wN}$ and $\lambda \in \bigcap_{i \in I} \Lambda(y, i)$ for an index set $I \subseteq \{1, \dots, p\}$ with $2 \leq |I| \leq p - 2$. If there exists a set of ND images $\bar{Y} \subseteq \mathcal{Y}_N \setminus \{y\}$ such that*

1. *$\lambda \in \Lambda(y^N)$ and $\lambda \notin \bigcup_{j \notin I} \Lambda(y^N, j)$, where $y^N := y^N(\bar{Y} \cup \{y\})$ and*
2. *for all $I' \subsetneqq I$ there exists an image $\bar{y} \in \bar{Y}$ with $\bar{y}_i < y_i$ for all $i \in I'$,*

then $\dim(\lambda, \Lambda(y)) = p - |I|$.

Proof. Since $\lambda \in \Lambda(y^N) \cap \Lambda^>$, we get by Proposition 8 that $\|y^N\|_\infty^\lambda = \|y\|_\infty^\lambda = \|\bar{y}\|_\infty^\lambda$ for all $\bar{y} \in \bar{Y}$. Since $\lambda \notin \bigcup_{j \notin I} \Lambda(y^N, j)$, we have $\|y^N\|_\infty^\lambda > \lambda_j y_j^N$ for all $j \notin I$ and therefore

$\|y'\|_\infty^\lambda = \|y^N\|_\infty^\lambda > \lambda_j y_j^N \geq \lambda_j y'_j$ for all $y' \in \bar{Y} \cup \{y\}$. Thus,

$$\lambda \notin \bigcup_{j \notin I} \Lambda(y', j) \text{ for all } y' \in \bar{Y} \cup \{y\}. \tag{3.15}$$

Since $\lambda \in \bigcap_{i \in I} \Lambda(y, i)$, we have $\lambda_i y_i = \|y\|_\infty^\lambda = \|y^N\|_\infty^\lambda \geq \lambda_i y_i^N$ for all $i \in I$. By definition of y^N we get $y_i = y_i^N$ for all $i \in I$ and thus for all $\bar{y} \in \bar{Y}$:

$$y_i \geq \bar{y}_i \text{ for all } i \in I. \tag{3.16}$$

We show $B_\varepsilon(\lambda) \cap \Lambda(y) \subseteq \{\lambda \in \Lambda : \lambda_i y_i = \lambda_k y_k,\ i, k \in I\}P$. This gives $\dim(\lambda, \Lambda(y)) \leq \dim(P) = p - |I|$.

Suppose there exists a $\lambda' \in B_\varepsilon(\lambda) \cap \Lambda(y)$ with $\lambda' \notin P$. Then, there is a subset $\emptyset \neq I' \subsetneq I$ such that $\|y\|_\infty^{\lambda'} = \lambda'_{i'} y_{i'}$ for all $i' \in I'$ and $\|y\|_\infty^{\lambda'} > \lambda'_i y_i$ for all $i \in I \setminus I'$. By (ii) there exists a $\bar{y} \in \bar{Y}$ with $\bar{y}_{i'} < y_{i'}$ for all $i' \in I'$. We get with (3.16) both $\|y\|_\infty^{\lambda'} = \lambda'_{i'} y_{i'} > \lambda'_{i'} \bar{y}_{i'}$ for all $i' \in I'$ and $\|y\|_\infty^{\lambda'} > \lambda'_i y_i \geq \lambda'_i \bar{y}_i$ for all $i \in I \setminus I'$. By (3.15), we can assume without loss of generality that ε is small enough such that $\lambda' \notin \bigcup_{j \notin I} \Lambda(y', j)$ for all $y' \in \bar{Y} \cup \{y\}$, and thus $\|\bar{y}\|_\infty^{\lambda'} = \lambda'_i \bar{y}_i$ for an $i \in I$, which gives $\|\bar{y}\|_\infty^{\lambda'} < \|y\|_\infty^\lambda$, a contradiction to $\lambda' \in \Lambda(y)$.

Since $\lambda \in \bigcap_{i \in I} \Lambda(y, i)$ and $\lambda \notin \bigcup_{j \notin I} \Lambda(y, j)$ we have $\dim(\lambda, \Lambda(y^N)) \geq p - |I|$ by Corollary 82. □

Note the similarity between Proposition 11 and item (ii) of the above propositions.

3.6 Conclusion

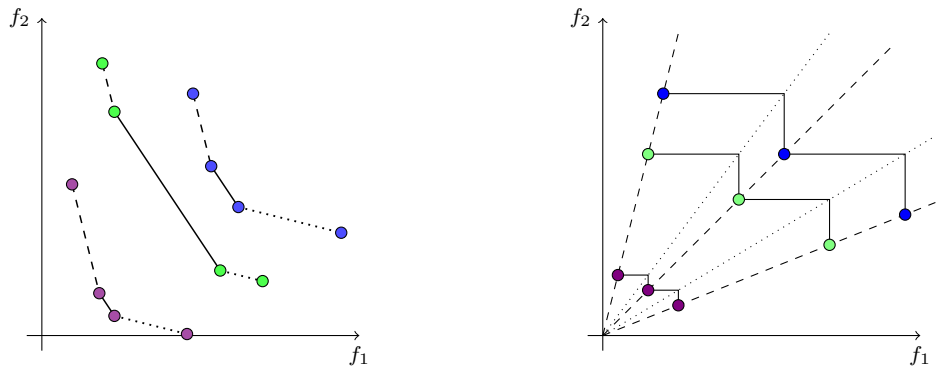
Together with the weighted sum and the ε-constraint method, the weighted Tchebychev method ranks highly with the most frequently applied scalarization techniques in multiobjective optimization. Moreover, the weighted Tchebychev scalarization problem is closely linked to many other single-objective optimization disciplines, including robust

optimization, goal programming, and location theory. It is a building block of many exact and heuristic algorithms, which systematically vary the choice of the weight to get a subset or even all ND images. In other words, these algorithms utilize elements of the weight set while the set itself has not yet been the focus of research.

In this article, we provide the first rigorous and comprehensive theory of the weight set decomposition by this weighted Tchebychev scalarization. We analyze the polyhedral and combinatorial structure of the weight set components as well as the composition of the weight set as a whole. To date, analogous research has only been published for the weighted sum method. However, there are substantial differences: The weighted Tchebychev scalarization is able to yield all efficient solutions (i.e., not only the supported ones as in the weighted sum method). Additionally, without "true" convexity, the structure of the weight set of the weighted Tchebychev method is more complex and the analysis is more technical. Through this analysis, convexity-related properties and bounds on dimension of the weight set components have been proven.

Contrasting the structures of the weight set decomposition of the weighted sum and the weighted Tchebychev scalarization provides some broader insights. For the weighted sum scalarization, the decomposition describes the gradients of the ND part of the convex hull of the set of images as well as information about adjacent ND facets. However, it neither provides information about the positioning nor the size of the convex hull in the image space. In fact, ND frontiers (of some multiobjective optimization problems) may vary substantially but still share the same weight set decomposition of the weighted sum scalarization (cf. Figure 3.10(a)).

In contrast, the weight set decomposition of the weighted Tchebychev norm yields more information about the positioning of the ND images. For an image y, the kernel vertex $\lambda(y)$ is defined by $y = t\lambda(y)$ for some $t > 0$, i.e. y is located on the ray $D(y) := \{\lambda(y)t : t > 0\}$. Recall that the weight set decomposition includes the kernel vertices for each ND image as well as the local Nadir weight vertices. Then, for each ND image



(a) (b)

Figure 3.10: Biobjective example of distinct sets of ND images (indicated by color), each of which have the same weight set decomposition with respect to weighted sum scalarization (a) or weighted Tchebychev norm (b). (a) The gradient vectors describing the convex hull are equivalent (parallel lines are indicated by line type, e.g., solid, dashed, and dotted) even though the frontiers vary widely in overall shape. (b) With the kernel vertices for weight set components known, then the ND images must exist within the associated rays (indicated by dashed lines). With the local nadir weights known as well, then the local nadir images must also exist within associated rays (indicated by dotted lines).

$y^r \in \mathcal{Y}_N = \{y^1, y^2, \ldots, y^R\}$, there is an associated ray, $D(y^r)$, with $y^r \in D(y^r)$ for $r = 1, \ldots, R$. A similar observation holds true for the local nadir images. Hence, the positioning of the ND images relative to each other can be narrowed down. In addition, knowing one ND image allows to restrict the locations of the remaining ND images such that nondominance is guaranteed and the local nadir images are not in conflict (see Figure 3.10(b)). An analogous deduction is not possible for the weight set decomposition of the weighted sum scalarization.

Future research must investigate the algorithmic applications of the derived properties. Star-shapedness and line convexity may be utilized to derive outer approximation [58] or inner approximation [49, 50] schemes that iteratively shrink or augment weight set components, respectively. The properties may be also utilized for interactive approaches with focus on a graphical exploration and presentation of solutions. Other directions include a thorough analysis of "duality" between the weight set decomposition and the image space illustrated in Figure 3.10 and the parameter sets of other scalarizations. For example,

weighted p-norm scalarizations or the augmented modified weighted Tchebychev scalarization yield ND images, only, and theoretically connect the already studied weight set decompositions. This may provide methods to deal algorithmically with overlapping components of weighted Tchebychev weight set components, and reveal additional insights on the images space and ND frontier of multiobjective optimization problems.

CHAPTER 4

TCHEBYCHEV WEIGHT SPACE DECOMPOSITION: APPLICATIONS

Abstract. *Multiobjective discrete optimization (MODO) techniques, such as weight space decomposition, have received great attention in the last decade. The primary weight space decomposition technique in the literature is defined for the weighted sum scalarization, through which sets of weights are assigned to a subset of the nondominated set. Recent work has added a new weight space decomposition defined for weighted Tchebychev scalarization, which provides the benefit of including all nondominated images but at the cost of "nice" convexity geometric properties. The current work applies the novel weight set decomposition as a dual perspective on existing MODO algorithms and includes strategies to improve algorithm measurement and design. This work includes a thorough evaluation of the added value of the new weight set decomposition by weighted Tchebychev in contrast to weighted sum. Existing algorithms are proven to return insufficient information to compute the weight set decomposition, then the necessary modifications to the algorithm are proven. Applications to existing algorithms include (inner and outer) approximations of the weight set components to measure and improve algorithm performance from perspective of the weight space. Additionally, a subset of the weight set can be used* a priori *to restrict an algorithm to return a targeted subset of the nondominated set. This work contributes an essential 2-dimensional visualization technique for MODOs with three objectives, which is inclusive of the entire nondominated set, by projecting the surface of the upper envelope of the nondominated set.* [1]

4.1 Introduction

Multiobjective discrete optimization problems (MODOs) have gained substantial and increasing attention in the optimization literature over recent years. This is motivated by the prevalence of multiple, conflicting objectives in applications as well as other fields of

[1]This work is intended to be submitted to *Mathematical Programming* (June 2021), coauthored with Stephan Helfrich, Pascal Halffmann, Natashia Boland, and Stefan Ruzika, with Tyler Perini as first author.

optimization, see e.g. [74, 75, 76, 77]. For such problems, there usually does not exist a solution that optimizes all objectives simultaneously. Hence, a decision maker is interested in the efficient set: the set of solutions for which one objective cannot be improved without depriving another. The vector of objective values associated with an efficient solution is referred to as a *nondominated* (ND) image. As a consequence, there exist multiple ND images. The solution to a multiobjective optimization problem is the set of all ND images, which is called the *ND frontier*.

Scalarization is a common approach to calculating single ND images. Scalarization transforms a multiobjective problem into a single-objective problem with the help of additional parameters such as reference points and weights on the objectives. Beyond discovering new ND images, the scalarizations can be used as a lens to analyze the structure of the ND images in relation to one another. Such analysis has already been done for the weighted sum scalarization [23, 78]. It is well known that this scalarization can solve multiobjective linear programs. For MODOs, the weighted sum scalarization is analogously used as a lens for analyzing the images that are extreme points of the convex hull of the ND frontier. While this analysis is limited in scope by the number of ND images belonging to this subset, the resulting geometric understanding of the ND frontier is valuable for decision makers and algorithmic development.

The outcomes of a weight-based scalarization resulting in an ND image, such as weighted sum, can generally be summarized by the set of nonnegative weights with unit sum, called the *weight set*. The technique of *weight space decomposition* utilizes a given weighted scalarization to assign subsets of the weight set to the associated ND images that are optimal for those weights. See Figure 4.1 as an example of two weight set decompositions. This technique satisfies the need for visualization in triobjective decision making, which has largely been satisfied for biobjective problems. Often, 3-dimensional visualizations of ND frontiers are difficult to intuitively make decisions from. Since the weights in the weight set have unit sum, then the weight set's dimension is one less than the dimension

of the problem, and so we have a 2-dimensional visualization for triobjective ND frontiers. Weight set decomposition views the weight set as a puzzle, and each ND image contributes a puzzle piece to the bigger picture. The area of each piece, which we call a *weight set component*, communicates a robustness to changes in the weight, and each component's neighbors are immediately clear. Visualization of the images, alone, is akin to presenting only a centroid for each of the puzzle pieces.

Here, we extend the work of Chapter 3, maintaining focus on the weighted Tchebychev scalarization. In contrast to weighted sum, this scalarization is capable of representing all images in the ND frontier. This lens provides a richer and more sophisticated view of the ND frontier structure, yielding better insights on adjacency and relative robustness of ND images. In addition, there is a tight relationship between the hyperrectangular level set of the weighted Tchebychev scalarization and the "boxes" used by algorithms to decompose the image space for MODOs (see [46] for a detailed list of such algorithms). As a result, this scalarization can directly improve this class of algorithms. We tailor insights from this weight space decomposition technique to modify classic MODO algorithmic approaches. By doing so, inner and outer approximations of the weight set components can be produced during run time. Additionally, computation can be reduced by solving for ND images associated with a subset of the weight set which represents the compromise between a set of weights.

4.1.1 Outline

Section 4.2 provides fundamental definitions, notation, and related literature. Section 4.3 restates the most important results on the geometry of the weighted Tchebychev weight set decomposition and includes new results. In Section 4.4, we characterize the link between primal and dual approaches, including how to modify a primal algorithm in order to compute information sufficient to represent the weighted Tchebychev weight set decomposition. The proof for covering the weight set as well as the procedure for plotting the

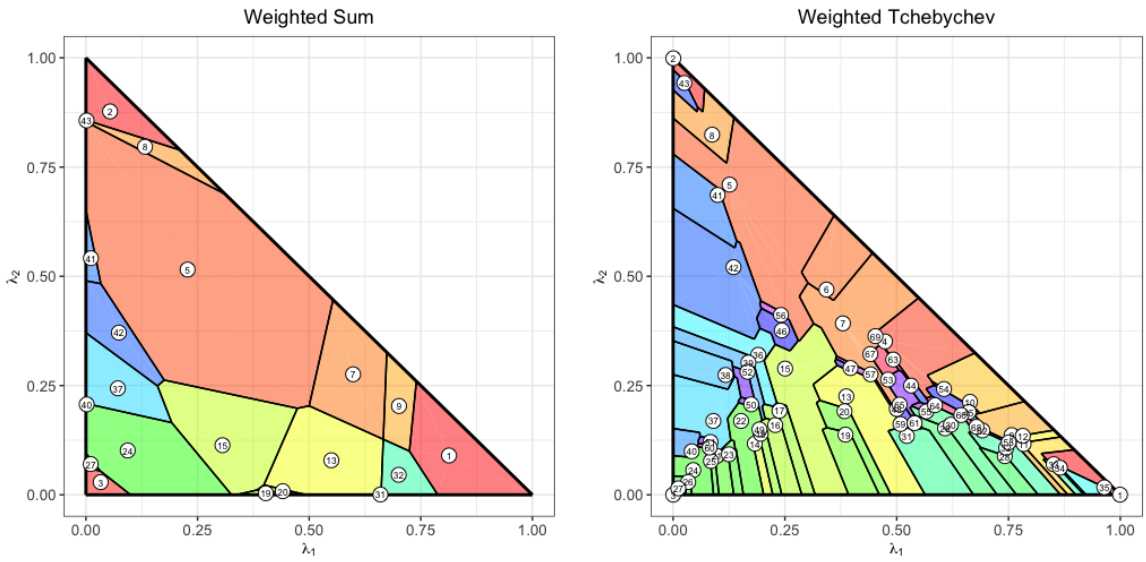


Figure 4.1: Weight space decomposition, according to two different scalarizations, of the same instance with 25 variables and $|\mathcal{Y}_N| = 69$. When using weighted sum scalarization (left), only 20 ESND images are included in the decomposition. When using weighted Tchebychev scalarization (right), all ND images are included in the decomposition. The numbered labels and color fill identify images and are consistent between figures.

weight set components, given this information, are presented in Section 4.5. Section 4.6 provides multiple direct applications of this weight set decomposition to primal algorithms. First, the weighted Tchebychev weight set decomposition is compared to the well-studied decomposition for weighted sum (Section 4.6.1). Second, we present procedures for computing the outer and inner approximations for the weight set components (Section 4.6.3). Lastly, we introduce a novel method for solving for a subset of the ND frontier associated with a subset of the weight set (Section 4.6.4).

4.2 Preliminaries

Recall the *weighted sum scalarization*, defined using nonnegative weights, λ_i for $i = 1, \ldots, p$, is given by $\min_{x \in \mathcal{X}} \{\lambda^T f(x)\}$ [18]. The image of every optimal solution is a (weakly) ND image of (MOP) if $\lambda \in \mathbb{R}^p_>$ ($\lambda \in \mathbb{R}^p_\geq$) [19]. By varying the weights, alternative ND images can be found. Weighted sum scalarizations only yield supported ND images. These are located on the convex hull of the set of images. Supported ND im-

ages that are also extreme points of the convex hull of Y are called *extreme supported ND (ESND)* images.

With a *reference point* $s \in \mathbb{R}^p$, a given weight vector $\lambda \in \mathbb{R}^p_{\geq}$ and $\|y\|^{\lambda}_{\infty} := \max_{i=1,\dots,p}\{|\lambda_i\, y_i|\}$, the *weighted Tchebychev scalarization* [60] can be stated as

$$\min \{\|f(x) - s\|^{\lambda}_{\infty} \ : \ x \in \mathcal{X}\}. \tag{$\Pi^{TS}(\lambda)$}$$

In practice, the scalarized single-objective in ($\Pi^{TS}(\lambda)$) is modeled by the introduction of an auxiliary variable. Usually, the reference point is chosen to be the *ideal point*, defined as $y^I_i := \min_{x\in\mathcal{X}}\{f_i(x)\}$ for $i = 1, \dots, p$, or to be a *utopia point* $y^U < y^I$.

For weights $\lambda \in \mathbb{R}^p_{>}$ and reference points $s \leqq y^I$, every optimal solution to $\Pi^{TS}(\lambda)$ is at least weakly efficient for (MOP). If the optimal solution is unique, its image is ND. Conversely, there exists a positive weight vector for each ND image y' such that an optimal solution x' for the weighted Tchebychev scalarization problem satisfies $y' = f(x')$ [20].

Observe for both weighted sum and weighted Tchebychev scalarizations that the set of optimal solutions does not change when the weight vector is multiplied by a positive scalar. Hence, the set of all weight vectors for which the scalarizations return (weakly) ND images ($\mathbb{R}^p_{\geq}$ and $\mathbb{R}^p_{>}$, respectively) can be projected to the $(p-1)$-dimensional *weight set*, defined as

$$\Lambda := \{\lambda \in \mathbb{R}^p_{\geq} : \sum_{i=1}^{p} \lambda_i = 1\},$$

i.e., the set of weights normalized to have unit sum. This set includes the boundaries at which one or more component of the weight vector is equal to zero. Some properties only apply to the interior of Λ, which we denote by $\text{int}(\Lambda) := \{\lambda \in \mathbb{R}^p_{>} : \sum_{i=1}^{p} \lambda_i = 1\}$. For each ND image, there exists a set of weights (possibly empty) such that the image is optimal for the weighted sum scalarization, and analogously there exists a set of weights such that the image is optimal for the weighted Tchebychev scalarization. Hence, for $y \in \mathcal{Y}$, we define *weight set components* as $\Lambda^{WS}(y) := \left\{\lambda \in \Lambda : \lambda^\top y \leq \ \lambda^\top \bar{y} \text{ for all } \bar{y} \in Y\right\}$ for the

weighted sum scalarization and $\Lambda(y) := \left\{\lambda \in \Lambda : \|y\|_\infty^\lambda \leq \|\bar{y}\|_\infty^\lambda \text{ for all } \bar{y} \in \mathcal{Y}\right\}$ for the weighted Tchebychev scalarization. The *weight set decomposition* refers to a collection of weight set components.

4.2.1 Related Literature

A growing library of algorithms for solving MODOs work directly in the image space [52, 41, 42, 43, 29, 44, 46]. We refer to [79] to a recent survey on solutions methods. Here, we focus on two different algorithm design approaches. Similar to [80], we call methods that work in the image space *primal approaches* and methods that operate in the weight set of scalarizations *dual approaches*.

Primal Approaches.

Primal algorithms are characterized by two predominant features: First, how the unexplored subset of the image space which may contain new ND images, called the *search region*, is represented. Second, how the mathematical optimization *subproblem* is defined to return a new ND image from (a subset of) the search region, if one exists. Solving the subproblem is generally the more expensive procedure; however, updating the representation of the search region determines the number of future subproblems to solve.

Given a set S of ND images, other ND images can only by contained in the search region $\mathcal{C} = \mathbb{R}^p \setminus \bigcup_{y \in S} y + \mathbb{R}^p_{\leqq}$. In Klamroth et al. [41], a representation of this search region for multiobjective problems is established by a so-called *upper bound set*, which is defined as a point-set $U(S) \subseteq \mathbb{R}^p$ such that $\bigcup_{u \in U(S)} (u - \mathbb{R}^p_{>}) = \mathcal{C}$ and is of minimum cardinality with this property. Each $u \in U(S)$ is referred to as an *upper bound point*. The proposed algorithmic procedure to generate all ND images of MODO iteratively picks a local upper bound u and explores the associated search zone $u - \mathbb{R}^p_{>}$ with an appropriate mathematical subproblem to discover a new ND image, if one exists. There is no new ND image if and only if the subproblem is infeasible, which can be computationally inefficient to determine.

If a ND image is found, it is guaranteed to not belong to S, and so the procedure updates the upper bound set $U(S)$. If no new image is found, u is removed from the upper bound set. Two methods are established to update the upper bound sets when a new ND image is found. One calculates candidate local upper bounds and filters redundant ones afterwards. It slightly improves the update procedure of [78]. The other stores for each upper bound an associated image set to establish an update procedure avoiding the filtering step. By the results of [81], the cardinality of the upper bound set is asymptotically bounded by $\mathcal{O}(|S|)^{\lfloor \frac{p}{2} \rfloor}$.

Dächert and Klamroth [42] presented an algorithm tailored to three objectives that also decomposes the search region into boxes. When a new ND image y is found, a so-called p-split of all boxes containing y is performed, which yields a feasible decomposition of the remaining search region based on the updated set of ND images. A straight-forward application of p-split induces redundant boxes. Thus, a neighbourhood concept of the boxes is defined, and the split method is adapted, based on this concept, to maintain a nonredundant decomposition of the search region. It is proved that for three objectives, this method requires at most $3|\mathcal{Y}_N| - 2$ subproblems to be solved to calculate the entire ND frontier.

Dächert et al. [44] extend the results of Dächert and Klamroth [42] and Klamroth et al. [41] to derive a neighbourhood relation of local upper bounds. Using the neighbourhood concept to update the local upper bound does not improve the asymptotic (worst case) running time. However, their computational results show a significant improvement in running time in practice, especially for instances with a high number of ND images.

Recent work by Tamby and Vanderpooten [46] has also added to the set of primal approaches. Their algorithm differs in the formulation of the subproblems. Rather than using p constraints (one for each objective), $p-1$ constraints are used and one objective remains unconstrained. In doing so, the algorithm avoids costly infeasible subproblems, and includes additional procedures, e.g., for selecting the unconstrained objective and handling

the rediscovery of images.

Dual Approaches.

Dual approaches primarily appear in the form of weight set decomposition methods (for the weighted sum scalarization) and dates back to the work by [67] who introduce a generalized simplex method for multiobjective linear programs. The work links basic efficient solutions with the set of weights in the polyhedral cone defined by the corresponding basis matrix. For biobjective problems, the well-known dichotomic search approach [22, 21] in fact calculates all ESND images based on the general idea of weight set decomposition: for two known ESND images, it calculates the unique weight such that the weighted sum scalarized value of both images coincide. Solving the weighted sum scalarization problem with this weight results in either a new ESND image or certification that no other ESND images exist 'between' the two. This idea was extended to establish a link between ESND images of a multiobjective linear optimization problem and a partitioning of the weight set [47, 82].

Przybylski et al. [23] adapt weight space decomposition by weighted sum scalarization to MODOs. They state fundamental properties concerning the weight set components, including convexity of weight space components, the weight set components for the set of ESND images covers the weight set, and a symmetric adjacency structure. Additionally, they present an algorithm for computing all ESND images for three objectives by investigating (one-dimensional) facets of weight set components with a dichotomic search approach. The weighted sum weight set decomposition is implicitly calculated by the procedures of [57] and [56]. The algorithms of [49] and [50] (improved in [51]) iteratively augment subsets of the weighted sum weight set components based on the convexity property.

For the weighted Tchebychev scalarization, [70] explicitly consider weight set components for biobjective problems. Based on this approach, [71] adapts the dichotomic search method to calculate all ND images of biobjective discrete optimization problems.

Chapter 3 gives a detailed study on the geometry of the weighted Tchebychev weight set decomposition for an arbitrary number of objectives.

4.3 Geometry of Weight Space Decomposition

We give an overview of the most relevant results from Chapter 3, which presents in-depth the geometry of the weight set decomposition for weighted Tchebychev scalarization. In the remainder of this work, we assume the following.

Assumption 2. *Without loss of generality, assume the image set is shifted such that* $Y \subset \mathbb{R}^p_>$. *Then the origin is a utopia point, and we fix this as the reference point, i.e.,* $s = \vec{0}$.

Assumption 2 allows s to be dropped from the objective of $(\Pi^{TS}(\lambda))$ and all equations moving forward. In Chapter 3, some useful properties of the weight set components are stated. We restate the results here.

Theorem 2. *Let* $B_\epsilon(\mu) := \{\mu' \in \Lambda : \sum_{i=1}^p (\mu_i - \mu'_i)^2 \leq \epsilon\}$.

1. *An image is weakly ND if and only if its weight set component is nonempty.*
2. $\Lambda = \bigcup_{y \in \mathcal{Y}_N} \Lambda(y)$.
3. *For* $y \in \mathcal{Y}_N$, *there exists* $\epsilon > 0$ *such that* $B_\epsilon(\lambda(y)) \subseteq \Lambda(y)$.
4. *Let* $y \in \mathcal{Y}_{wN}$. *Then*

$$\Lambda(y) = \{\lambda \in \Lambda : \|y\|^\lambda_\infty \leq \|\bar{y}\|^\lambda_\infty \text{ for all } \bar{y} \in \mathcal{Y}_N\}.$$

Note that many of these results are analogous to the results of Przybylski et al. [23] but for the weighted Tchebychev scalarization instead of the weighted sum scalarization. Result 1 indicates that only weakly ND images may contribute to the decomposition. However, results 2 and 4 indicate that ND images alone are sufficient to compute the decomposition. It follows that there may be multiple distinct decompositions (for instance, whether

weakly ND images are included in $\bar{Y}$); however, the decomposition for $\bar{Y} = \mathcal{Y}_N$ is unique for every (MODO) instance.

For a given (weakly) ND image, a particular weight, called the kernel weight, is central to the geometric analysis of its weight set component.

Definition 13. *For $y \in \mathcal{Y}_{wN}$, we denote the* kernel weight *or* kernel vertex[2] *of y by $\lambda(y)$ and define it by*

$$\lambda_i(y) := \frac{1}{y_i} \frac{1}{\sum_{j=1}^{p} \frac{1}{y_j}} \text{ for } i = 1, \ldots, p.$$

Observe that the level set for the objective function of ($\Pi^{TS}(\lambda)$) is the surface of a hyperrectangle that is centered at the origin with proportions determined by λ. Geometrically, the kernel weight $\lambda(y)$ is the unique weight in Λ such that image y exists on the (positive) vertex of such a hyperrectangle.

As seen in Figure 4.1, the weight set components with respect to weighted sum scalarization (left) are convex, while the weight set components with respect to weighted Tchebychev sclarization (right) are nonconvex. However, the nonconvex polygons do share a related structure, which is a weaker form of convexity called star-shapedness. We present a definition that is simplified from [72].[3]

Definition 14. *A set $S \subseteq \mathbb{R}^p$ is* star-shaped *if there exists $\bar{y} \in S$ such that $\theta\bar{y} + (1-\theta)y \in S$ for all $y \in S$ and all $\theta \in (0, 1)$. Such a point $\bar{y}$ is called a* kernel.

A fundamental result from Chapter 3 is that the kernel weight of $y \in \mathcal{Y}_{wN}$, $\lambda(y)$, is a kernel of the weight set component $\Lambda(y)$, as the name implies. In Figure 4.1, each of the numbered circles indicates the location of a kernel weight. Figure 4.2 also aids in demonstrating the star-shaped structure. The theorem is restated here.

Theorem 3. *Let $y \in \mathcal{Y}_{wN}$. Then, $\Lambda(y)$ is a star-shaped set with $\lambda(y)$ as a kernel.*

[2] Also known as *T-vertex* [64].

[3] The original definition allows for a *kernel set*, however we have simplified this set to always be a singleton.

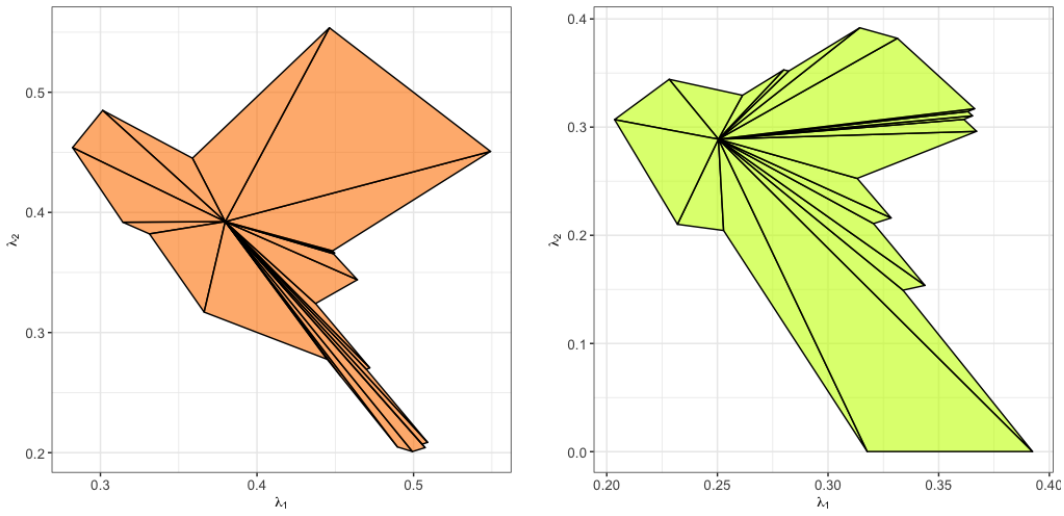


Figure 4.2: Star-shaped weight set components $\Lambda(y^7)$ (left) and $\Lambda(y^{15})$ (right) from Figure 4.1.

4.3.1 Running Example

Example 10. *Consider the following* (MOP)*:*

$$\min\{f(x) = x : x_1 + x_2 + x_3 = 6, 1 \leq x_i \leq 3 \; \forall i = 1, 2, 3, x \in \mathbb{Z}^3\}.$$

The following sets coincide: $\mathcal{X} = \mathcal{Y} = \mathcal{Y}_N$*. There are seven ND images, labeled:* $y^1 = (1, 2, 3)$*,* $y^2 = (1, 3, 2)$*,* $y^3 = (2, 2, 2)$*,* $y^4 = (2, 1, 3)$*,* $y^5 = (2, 3, 1)$*,* $y^6 = (3, 1, 2)$*, and* $y^7 = (3, 2, 1)$*. The complete weighted Tchebychev decomposition is illustrated in Figure 4.3(a) as well as a single weight set component,* $\Lambda(y^1)$*, in Figure 4.3(b). The remaining weight set components are illustrated in Figure B.1 of the Appendix.*

This example exhibits common complications observed when computing and representing the weight set decomposition. First, many of the weight set components overlap*, meaning the intersection is full-dimensional. This is made visible by choosing to plot each component with partially-transparent colors so that the overlap of components stands out. Observe in Figure 4.3 that each corner of* Λ *is an intersection between components, e.g.,* $\Lambda(y^1) \cap \Lambda(y^2)$ *in orange. Second, lower-dimensional regions occur. For three objectives,*

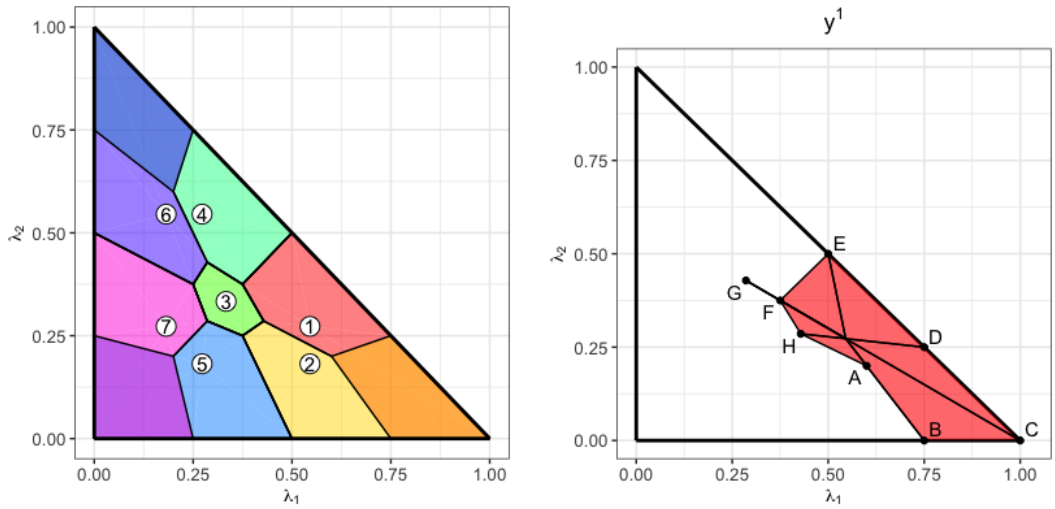


Figure 4.3: The weighted Tchebychev weight set decomposition for the running example (Example 10, left) as well as the individual weight set component $\Lambda(y^1)$ (right). On the left, the following intersections between weight set components can be observed: $\Lambda(y^1) \cap \Lambda(y^2)$ in orange, $\Lambda(y^4) \cap \Lambda(y^6)$ in navy, and $\Lambda(y^5) \cap \Lambda(y^7)$ in violet. On the right, each unique local nadir weight is labeled.

this appears as a 1-dimensional line segment protruding from a weight set, which we call whiskers. *The weight set component $\Lambda(y^1)$ contains one whisker, i.e., the line segment $[F, G]$, where $[a, b] := \{\mu a + (1 - \mu)b : 0 \leq \mu \leq 1\}$. Each weight set component in this example, $\Lambda(y^1), \ldots, \Lambda(y^7)$ (illustrated in Figure B.1 of the Appendix) include at least one whisker.*

4.3.2 Intersection Sets

The intersections of weight set components can be used to define an adjacency structure among ND images. The following definition of adjacency aligns with the definition by weighted sum.

Definition 15. *Two ND images, $y^1, y^2 \in \mathcal{Y}_N$, are* adjacent *if dim$(\Lambda(y^1) \cap \Lambda(y^2)) \geq p - 2$.*

Recall $\dim(\Lambda) = p - 1$. In the weighted sum weight set decomposition, intersections between weight set components have several nice properties, including that they are convex and $(p - 2)$-dimensional [23]. However, in the weighted Tchebychev weight set

decomposition, the intersection set can be very complex and full-dimensional, which here means that it contains a $(p-1)$-dimensional ball with small, positive radius ϵ. For a proper proof of dimension of these nonconvex sets, characterized by polytopal decomposition, see Chapter 3.

In the most common case of weighted Tchebychev weight set components, we observe a simple structure for the intersection sets between adjacent ND images. For $p = 3$, an intersection set between two adjacent ND images with *distinct* components is a union of two line segments of the form $[\lambda^1, \lambda^2] \cup [\lambda^2, \lambda^3]$ where $\{\lambda^1, \lambda^2, \lambda^3\} \subset \Lambda$.[4] This is illustrated in Example 11. Upon closer observation, all intersection sets in Figure 4.1(b) are of this simple form.

Example 11. *Let* $Y' = \{y^1, y^5\} = \{(1,2,3),(2,3,1)\}$ *(selected from Example 10 such that images are distinct in every component). The weighted Tchebychev weight set decomposition for* Y' *is illustrated in Figure 4.4(a). A minimal description of the intersection set is given by the interior weight* $\lambda^2 = (0.375, 0.375, 0.25)$ *and two weights on the boundary of* Λ, $\lambda^1 = (0.0, 0.6, 0.4)$ *and* $\lambda^3 = (0.6, 0.0, 0.4)$. *Hence,* $\Lambda(y^1) \cap \Lambda(y^2) = [\lambda^1, \lambda^2] \cup [\lambda^2, \lambda^3]$.

More complex intersections, including full-dimensional sets or "overlapping" weight set components, often occur when adjacent ND images share a component. This is illustrated in Example 12.

Example 12. *Let* $Y'' = \{y^1, y^2\} = \{(1,2,3),(1,3,2)\}$ *(selected from Example 10 such that images are equal in the first component). The weighted Tchebychev weight set decomposition for* Y'' *is illustrated in Figure 4.4(b). Now, the intersection contains a full-dimensional set (orange). We have* $\Lambda(y^1) \cap \Lambda(y^2) = [\lambda^1, \lambda^2] \cup \text{conv}(\{\lambda^2, \lambda^3, \lambda^4, \lambda^5\})$, *where*

- $\lambda^1 = (0.00, 0.50, 0.50)$,

[4] The weights do not necessarily have to be distinct, for example in the case where LNPs coincide, which is discussed later.

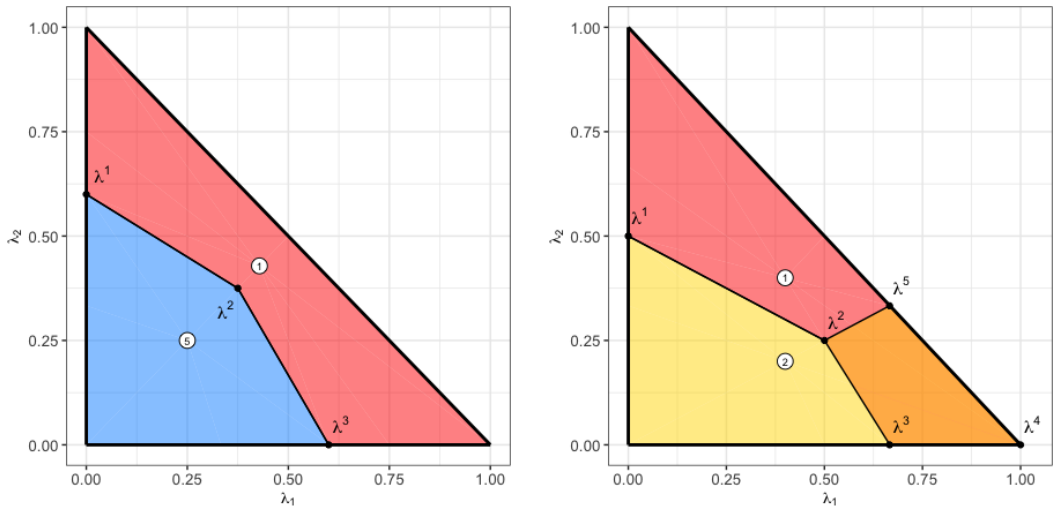


Figure 4.4: Weighted Tchebychev weight set decompositions for Y' (Example 11, left) and Y'' (Example 12, right). On left, the two images are distinct in all components, and both images are optimal for $\lambda \in [\lambda^1, \lambda^2] \cup [\lambda^2, \lambda^3]$. On right, images are equal in the first component, and both images are optimal for $\lambda \in [\lambda^1, \lambda^2] \cup \text{conv}(\lambda^2, \lambda^3, \lambda^4, \lambda^5)$. The orange polytope illustrates the overlap of the red and yellow weight set components, i.e., $\Lambda(y^1) \cap \Lambda(y^2) = \text{conv}(\lambda^2, \lambda^3, \lambda^4, \lambda^5)$.

- $\lambda^2 = (0.50, 0.25, 0.25)$,
- $\lambda^3 = (0.66, 0.00, 0.33)$,
- $\lambda^4 = (1.00, 0.00, 0.00)$, *and*
- $\lambda^5 = (0.66, 0.33, 0.00)$.

Extra complications for primal algorithms when ND images are equal in one (or more) components are discussed in [42]. Related complications are also apparent from the perspective of the dual.

To deconstruct and describe the intersection sets, we use local nadir points. Note that this oft-used concept goes by many different names, e.g., our definition generalizes the definition for "upper bound points" in [41], [42], and [44].

Definition 16. *Given a set of weakly ND images, $\bar{Y} = \{y^1, \ldots, y^{\bar{R}}\} \subseteq \mathcal{Y}_{wN}$, we define the*

local nadir point (LNP) *with respect to* $\bar{Y}$*,* $y^N(\bar{Y})$*, by*

$$y_i^N(\bar{Y}) = \max_{r=1,\dots,\bar{R}} y_i^r \text{ for } i = 1, \dots, p.$$

We often make explicit how many images are included in $\bar{Y}$*, e.g., if* $\bar{R} = 2$*, 3, or 4, then we call* $y^N(\bar{Y})$ *a 2-way, 3-way, or 4-way LNP, respectively.*

Definition 17. *We say an image* $y \in \bar{Y}$ contributes (in component i) *to the LNP* $n := y^N(\bar{Y})$ *if* $y_i = n_i$ *for some* $i \in \{1, \dots, p\}$*. For* $i \in \{1, \dots, p\}$*, the* ith dimensional contributing set of n*, denoted* $C^i(n)$*, is the subset of* $\bar{Y}$ *that contributes to LNP* n *in component* i*. An image* y uniquely contributes *to* $y^N(\bar{Y})$ *in component* i*, if* $C^i(n) = \{y\}$.

Due to ND images with shared components, it can be observed that a 2-way (k-way) LNP coincides with a 3-way ($k+1$-way) LNP. Therefore, there is not an immediate, one-to-one relationship between contribution sets and LNPs, which motivates the following definition for describing a particular contributing set of interest for a fixed LNP.

Definition 18. *For a given LNP, its contributing set* $\bar{Y}$ *is* complete *with respect to* $\mathcal{Y}_N$ *if for all* $y' \in \mathcal{Y}_N \setminus \bar{Y}$*,* $y^N(\bar{Y}) \neq y^N(\bar{Y} \cup \{y'\})$.

It can be trivially proven that for a fixed LNP and image set, there is a unique contributing set which is complete. As ND images are added to a contributing set, the coordinates of the LNP may or may not change, and the resulting LNP may or may not be weakly ND. Definition 18 is concerned with the former condition, and the following definition is concerned with the latter.

Definition 19. *A given LNP with contributing set* $\bar{Y}$ *is* maximal *with respect to* $\mathcal{Y}_N$ *if* $y^N(\bar{Y})$ *is not strictly dominated by any* $y' \in \mathcal{Y}_N$*, but* $y^N(\bar{Y} \cup \{y'\})$ *is strictly dominated by some* $y' \in \mathcal{Y}_N$.

When a LNP is not strictly dominated by any $y' \in Y$, we say it is *weakly ND with respect to Y*. A maximal LNP is weakly ND with respect to $\mathcal{Y}_N$ unless any new image is

added to its contributing set. Note that Definition 19 is not concerned with the coordinates of LNP changing by adding a new image to its contributing set but only its status as weakly ND. In general, we will aim to compute all maximal LNPs, each with its complete contributing set.

Example 13. *Consider the following list of LNPs defined with images from Example 10:*

1. $y^N(\{y^1, y^2\}) = (1, 3, 3)$,
2. $y^N(\{y^1, y^2, y^3\}) = (2, 3, 3)$,
3. $y^N(\{y^1, y^2, y^3, y^4\}) = (2, 3, 3)$,
4. $y^N(\{y^1, y^2, y^3, y^4, y^5\}) = (2, 3, 3)$,
5. $y^N(\{y^1, y^2, y^3, y^4, y^5, y^6\}) = (3, 3, 3)$, *and*
6. $y^N(\{y^1, y^2, y^3, y^4, y^5, y^6, y^7\}) = (3, 3, 3)$.

Since LNPs 2-4 coincide, we say that the contributing set $\{y^1, y^2, y^3, y^4, y^5\}$ *is* complete *for LNP* $(2, 3, 3)$. *Furthermore, note that LNPs 1-4 are weakly ND with respect to* $\mathcal{Y}$*, while LNPs 5 and 6 are strictly dominated by* $y^2 = (2, 2, 2)$. *Since adding* y^6 *or* y^7 *to the contributing set results in a strictly dominated LNP, we say* $(2, 3, 3)$ *is a* maximal *LNP. This LNP corresponds to* H *in Figure 4.3.*

We note some subtle distinctions between our definitions and definitions in the literature. First, our definition of *maximal* LNPs fundamentally align with the definition for upper bounds in Klamroth et al. [41]. However, the definition for upper bound point only includes k-way LNPs where $k \geq p$. Hence, for three objectives, 2-way LNPs are excluded simply by definition. Our later results make clear why this exclusion makes practical sense for the primal approach without loss of accuracy; regardless, 2-way LNPs are essential in the weight space decomposition. In general, primal approaches do not afford much attention to the specific number of ND images R, e.g., whether $R = p$, $R = p + 1$, etc. Since

that distinction is more consequential in the weight space, we emphasize it more in this work. Second, note that in Klamroth et al. [41], Dächert and Klamroth [42], and Dächert et al. [44], upper bound points are described as either *redundant* or *nonredundant*. By our terminology, we refer to LNPs as either strictly dominated (say, by an image or another LNP) or weakly ND, respectively. For sake of consistency, hereon we use our terminology, and only where appropriate we relate to alternative definitions.

In general, $(p+1)$-way (and higher-order) LNPs are prevalent where adjacent ND images coincide in components. For instance, Example 13 illustrates a 5-way, maximal LNP. Therefore, an algorithm that is designed to only consider p-way LNPs but not higher-order points will miss adjacency information.

Analogous to kernel weights, we now define the weights that correspond to the LNPs.

Definition 20. *We call the kernel weight of a LNP the* local nadir weight *and denote it as* $\lambda^N(\bar{Y}) := \lambda(y^N(\bar{Y}))$.[5] *Two LNPs, or their corresponding local nadir weights, are said to* coincide *if they are equal in all* p *components.*

It can easily be shown that the weighted Tchebychev norm has equal value for all images contributing to the LNP, i.e., $\|y\|_\infty^{\lambda^N(\bar{Y})} = \|y'\|_\infty^{\lambda^N(\bar{Y})}$ for all $y, y' \in \bar{Y}$.

The properties of star-shapedness apply to all (nonempty) intersections of weight set components. Furthermore, the kernel of an intersection set, say $\cap_{y\in\bar{Y}}\Lambda(y)$ for some $\bar{Y} \subseteq \mathcal{Y}_N$ is given by the local nadir weight, $\lambda^N(\bar{Y})$. This geometric property is proven in detail in Chapter 3 and given succinctly as the following theorem which fundamentally proves that an intersection set has the same properties as a weight set component.

Theorem 4. *Let* $\bar{Y} \subseteq \mathcal{Y}_{wN}$. *Then,*

$$\bigcap_{y\in\bar{Y}} \Lambda(y) = \Lambda(y^N(\bar{Y})).$$

[5]The definition for $y^N : \mathcal{Y}_{wN} \to \mathbb{R}^p$ requires an input set of images from $\mathcal{Y}_{wN}$. For ease of exposition, we assume in the definition that the LNP exists in $\mathcal{Y}_{wN}$.

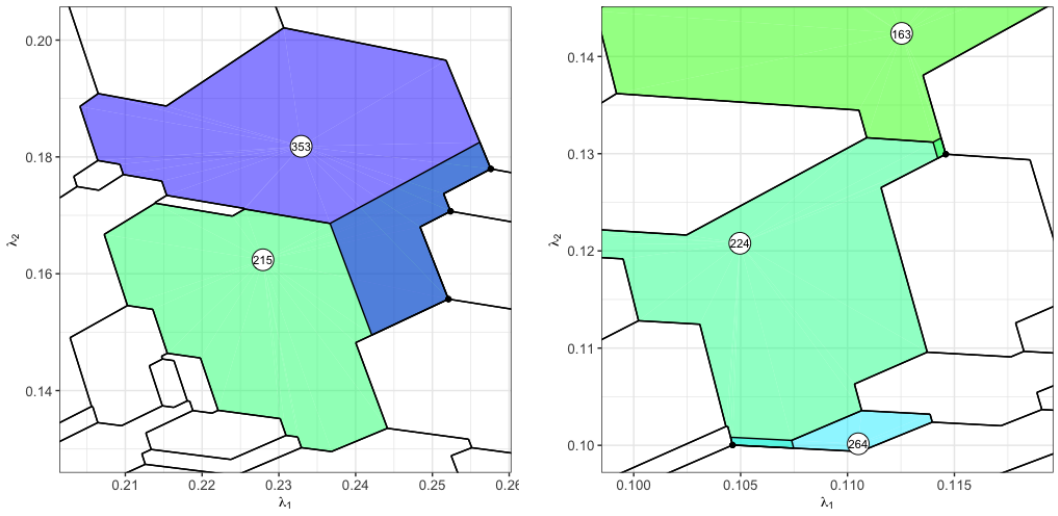


Figure 4.5: Examples of overlapping weight set components from an instance with 50 variables and $|\mathcal{Y}_N| = 507$. For clarity, the non-overlapping weight set components are unfilled. The numbers indicate the location of the kernel weight for the labeled weight set component. Black points indicate 4-way local nadir weights. (a) The intersection $\Lambda(y^{215}) \cap \Lambda(y^{353})$ is in dark blue. (b) Weight set component $\Lambda(y^{224})$ has two disconnected, full-dimensional intersections: $\Lambda(y^{224}) \cap \Lambda(y^{163})$ (small, neon green rectangle) and $\Lambda(y^{224}) \cap \Lambda(y^{264})$ (thin, teal rectangle).

Example 14. *In Example 11, the kernel point of the intersection set is the local nadir weight for the 2-way LNP:* $\lambda^2 = \lambda^N(\{y^1, y^5\})$. *In Example 12, the kernel point of the intersection set is similarly* $\lambda^2 = \lambda^N(\{y^1, y^2\})$. *See Figures 4.4(a) and (b), respectively.*

For more complex intersection sets, see Figure 4.5. These examples illustrate that star-shapedness of the intersection set is satisfied as well as star-shapedness of the individual weight set components to which they belong. The result of intersecting star-shaped sets being star-shaped, when the kernel weights are not included in the intersection, is nontrivial.

Theorem 4 asserts that an intersection of weight set components is fundamentally the same as a weight set component. This implies the following corollaries.

Corollary 9. *Let* $\bar{Y} \subseteq \mathcal{Y}_{wN}$. *Then,*

1. *The intersection* $\bigcap_{y \in \bar{Y}} \Lambda(y)$ *is a star-shaped set with* $\lambda^N(\bar{Y})$ *in its kernel.*

2. *If* $\lambda^N(\bar{Y}) \notin \bigcap_{y \in \bar{Y}} \Lambda(y)$, *then* $\bigcap_{y \in \bar{Y}} \Lambda(y) = \emptyset$.

3. A LNP $y^N(\bar{Y})$ is weakly ND if and only if $\bigcap_{y\in\bar{Y}} \Lambda(y)$ is nonempty.

Result 3 from Corollary 9 provides a test for determining whether an intersection is empty or nonempty.

4.3.3 Boundary weights

Given subset $\bar{Y} \subseteq \mathcal{Y}_N$, the local nadir weight $\lambda^N(\bar{Y})$ exists in int(Λ). However, as we have seen in Examples 11 and 12, weight set components (and their intersections) also include weights on the boundaries of Λ. It is equivalent to introduce dummy images, each with a large component, such that each boundary weight is also the kernel weight of a LNP. Define the p dummy images, $y^{b(1)}, \ldots, y^{b(p)}$ by $y_i^{b(i)} = M$ for some large $M > y^N(\mathcal{Y}_N)$ and $y_j^{b(i)} = y_j^U$ for all $j \neq i$. Let $Y_B := \{y^{b(1)}, \ldots, y^{b(p)}\}$ be the set of dummy weights for representing boundary weights.

First, observe that a local nadir weight $\lambda^N(\bar{Y})$ is not defined for any $\bar{Y} \subset Y_B$ due to zero in the denominator (but $\bar{Y} = Y_B$ is defined). That said, it is implied from now on that Y_B is included in $\mathcal{Y}_N$ and that local nadir weights, $y^N(\bar{Y})$, are defined such that $\bar{Y} \not\subset Y_B$.

Second, observe that if $y^{b(i)} \in \bar{Y} \subseteq \mathcal{Y}_N$, then for finite M the local nadir weight $\lambda^N(\bar{Y})$ will have small, positive ith component and hence lie in int(Λ); however, by limit we have $\lim_{M\to\infty} \lambda_i^N(\bar{Y}) = 0$ and so $\lim_{M\to\infty} \lambda^N(\bar{Y})$ yields a weight on bound(Λ).[6] Also, note that more than one dummy image can be used to define a LNP or local nadir weight, which is how the vertices of Λ are represented. For example, in Figure 4.4(a), the vertex $(0, 1, 0)$ is equivalent to $\lim_{M\to\infty} \lambda^N(\{y^2, y^{b(1)}, y^{b(3)}\})$, and in Figure 4.4(b), $\lim_{M\to\infty} \lambda^N(\{y^1, y^2, y^{b(1)}, y^{b(3)}\}) = (0, 1, 0)$. The definitions for weakly ND local nadir weights are still well-defined for such weights.

Assumption 3. *We assume that the set of dummy images is included in the set of ND images, i.e., $Y_B \subset \mathcal{Y}_N$.*

[6] An alternative way to compute boundary weights is to set the ith component to zero and compute the remaining components by computing the $(p-1)$-dimensional local nadir weight while dropping the ith dimension.

Example 15. *Image y^1 contributes to four maximal LNPs. These LNPs and each of the complete (dimensional) contributing sets are given in Table 4.1 with labels corresponding to Figure 4.3. Note that C and E are 4-way LNPs but G and H are 7-way LNPs.*

Table 4.1: The maximal LNPs to which y^1 contributes with complete contributing sets in Example 10. For each LNP, the label used in Figure 4.3 is given as well as the images in each dimensional contributing set, C^1, C^2, C^3. Images $y^{b(2)}$, and $y^{b(3)}$ refer to the dummy images.

LNP	$(1, M, M)$	$(2, 2, M)$	$(3, 2, 3)$	$(2, 3, 3)$
Label	C	E	G	H
$C^1(n)$	y^1, y^2	y^3, y^4	y^6, y^7	y^3, y^4, y^5
$C^2(n)$	$y^{b(2)}$	y^1, y^3	y^1, y^3, y^7	y^2, y^5
$C^3(n)$	$y^{b(3)}$	$y^{b(3)}$	y^1, y^4	y^1, y^4

4.4 Computing Maximal LNPs with Complete Contributing Sets

We are interested in an algorithm which computes for each ND image its complete weight set component. As for primal algorithms, maximal LNPs are key to this procedure. Maximal LNPs have been studied for primal algorithms, as they are useful for decomposing the search region. Unlike the primal approach, the dual approach requires the complete contributing sets for each maximal LNP. In Section 4.5.1, we prove that the set of maximal LNPs with each complete contributing set is sufficient to cover the weight set. For now, we present how to modify a primal approach to compute the maximal LNPs along with complete contributing sets, simultaneously.

An existing primal algorithm, presented in Klamroth et al. [41], updates a set of maximal LNPs when given a new ND image. Algorithm 1 is adapted from [41] such that the output contributing sets are complete. Input is denoted by horizontal bars, e.g., the input set of maximal LNPs as $\bar{N}$ and the input image as $\bar{y}$. The algorithm takes as input the contributing set of images for each component $j \in \{1, \ldots, p\}$ of each LNP $n \in N$, denoted by $\bar{C}^k(n)$. The output includes a set of LNPs, N, and contributing sets $\{D^j(n)\}_{j=1..p, n \in N}$ and $\{C^j(n)\}_{j=1..p, n \in N}$. The former set, $D^j(n)$, represents the contributing sets as defined

Algorithm 1 Extended update procedure: maximal LNPs and complete contributing sets

Input: Set of maximal LNPs $\bar{N} \subset \mathbb{R}^p$, Contributing Sets $\{\bar{C}^j(n)\}_{j=1..p,n\in\bar{N}}$, and ND image $\bar{y}$.
Output: Updated set of maximal LNPs N, contributing sets $\{D^j(n)\}_{j=1..p,n\in N}$, and complete contributing sets $\{C^j(n)\}_{j=1..p,n\in N}$.

1: $D^j(n), C^j(n) \leftarrow \bar{C}^j(n) \quad \forall j = 1..p, n \in \bar{N}$
2: $A \leftarrow \{n \in \bar{N} : \bar{y} < n\}$
3: $P \leftarrow \emptyset$
4: **for** $n \in \bar{N}$ and $j \in \{1, \ldots, p\}$ such that $\bar{y}_j = n_j$ **do**
5: **if** $(\bar{y}_{-j} < n_{-j})$ **then** $D^j(n) \leftarrow D^j(n) \cup \{\bar{y}\}$
6: **if** $(\bar{y}_{-j} \leq n_{-j})$ **then** $C^j(n) \leftarrow C^j(n) \cup \{\bar{y}\}$
7: **end for**
8: **for** $n \in A$ **do**
9: **for** $j = 1, \ldots, p$ **do**
10: $m_j(n) \leftarrow \max_{k\neq j} \min\{y_j : y \in \bar{C}^k(n)\}$
11: **if** $\bar{y}_j > m_j(n)$ **then**
12: $n' \leftarrow y^N(\{\bar{C}^k(n)\}_{k\neq j} \cup \{\bar{y}\}) = (\bar{y}_j, n_{-j})$
13: $P \leftarrow P \cup \{n'\}$
14: $D^j(n') \leftarrow \{\bar{y}\}$
15: $C^j(n') \leftarrow \{\bar{y}\} \cup \{y \in C^k(n) : y \leq n', y_j = \bar{y}_j,\ k = 1, \ldots, p\}$
16: **for** $k \in \{1, \ldots, p\} \setminus \{j\}$ **do**
17: $D^k(n') \leftarrow \{y \in D^k(n) : y_j < \bar{y}_j\}$
18: $C^k(n') \leftarrow \{y \in C^k(n) : y_j \leq \bar{y}_j\}$
19: **end for**
20: **end if**
21: **end for**
22: **end for**
23: $N \leftarrow (\bar{N} \setminus A) \cup P$
24: **return** N, $\{D^j(n)\}_{j=1..p,n\in N}$, $\{C^j(n)\}_{j=1..p,n\in N}$

in [41]'s algorithm. The latter set, $C^j(n)$, represents where we modify the procedure to capture the *complete* contributing sets. Remark that, in practice, only one of these sets is of interest, so either the lines with $D^j(n)$ or the lines with $C^j(n)$ will be omitted. Throughout the algorithm, we use as shorthand (y^1_j, y^2_{-j}) to represent the vector where the jth component is given by y^1 and all other components are given by y^2.

The extended update method begins with new contribution sets are initialized as equal to input contribution steps (step 1). The set A represents the LNPs that are strictly dominated by the new ND image and must therefore be replaced (defined in step 2). The set P is initialized as empty (step 3) and will contain the new maximal LNPs as they are constructed. The extended update method will conclude by removing A from the input set of LNPs and adding the new LNPs in P (step 3), followed by returning the maximal LNPs along with contributing sets (step 24).

Details and proofs for the primal algorithm are given in Klamroth et al. [41]. Here, we highlight the rationale for the modifications to define complete contribution sets (steps 6, 15, and 18).

The for loop over $n \in \bar{N}$ (steps 4-7) of the primal algorithm is intended to add to contributing sets $D^j(n)$ only for the particular case that for some $n \in \bar{N}$, $\bar{y}_j = n_j$ for exactly one index j. However, if this is true for two indices (or more for $p > 3$), then step 6 in the modified approach accounts for this image's contribution to the LNP n by adding to $C^j(n)$.

The for loop over $n \in \bar{A}$ (steps 8-22) of the primal algorithm generates the new LNPs that replace the removed LNPs. A straight-forward p-split approach would replace n with p new LNPs each iteration, some of which may not be maximal. However, the condition in step 11 reduces the overall effort required by determining which of the p LNPs are necessary to add; see [41] for details. The new LNPs are defined in step 12 and added to P in step 13. The primal approach for defining the new contributing set is to give a *minimal* set of contributing images per component. Therefore, the new ND image alone is sufficient for defining $D^j(n')$ (step 14). However, for complete contributing sets $C^j(n')$, it is necessary to consider all other images that contribute to the LNP that are also equal in the jth component (step 15). Lastly, the modification in step 18 follows the same rationale as for step 6.

The following example illustrates how the modifications return complete contributing sets that are otherwise incomplete in the original primal approach.

Example 16. *Since Algorithm 1 is sensitive to the order of images used to update, let $\bar{Y}$ represent the set of images used to update before $y^1 = (1, 2, 3)$ is input to update. Let $\bar{Y} = \{y^3, y^4, y^{b(1)}, y^{b(2)}, y^{b(3)}\}$, $\bar{N}$ contain all maximal LNPs with respect to $\bar{Y}$, and contributing sets $\bar{C}^j(n)$ be complete with respect to $\bar{Y}$ for all $j = 1, \ldots, p$ and $n \in \bar{N}$. Consider $n := y^N(y^{b(1)}, y^3, y^4) = (M, 2, 3) \in \bar{N}$ with $\bar{C}^1(n) = \{y^{b(1)}\}$, $\bar{C}^2(n) = \{y^3\}$, and $C^3(n) = \{y^4\}$. Now with y^1 as the input ND image, it satisfies $y^1_j = n_j$ for both*

$j = 2, 3$*, but it does not satisfy* $y^1_{-j} < n_{-j}$ *for either* $j = 2, 3$*. Hence, the condition of line 5 is not satisfied, and so* y^1 *is not added to* $D^j(n)$ *for* $j = 2, 3$*. On the other hand,* $y^1_{-j} \leq n_{-j}$ *is satisfied for* $j = 2, 3$*, so* y^1 *is added to* $C^j(n)$ *for* $j = 2, 3$*. The LNP* n *belongs to the output set* N*, regardless, however the contributing sets* $D^2(n), D^3(n)$ *will be missing* y^1 *whereas the* complete *contributing sets* $C^2(n'), C^3(n')$ *will include* y^1*.*

Next, let $\bar{Y} = \{y^1, y^3, y^4, y^{b(1)}, y^{b(2)}, y^{b(3)}\}$ *and now consider* incomplete *contributing sets are input from the previous step, i.e.,* $\bar{C}^j(n) = D^j(n)$*, and so* $D^2(n) = \{y^3\}$ *and* $D^3(n) = \{y^4\}$*. Let* $y^6 = (3, 1, 2)$ *be input next. Note that* $y^6 < n$*, and so* $n \in A$ *by step 2. For* $j = 1$*, the condition in step 11 is satisfied, so* $n' := y^N(\bar{C}^2(n) \cup \bar{C}^3(n) \cup \{y^6\}) = (3, 2, 3)$ *is defined. By step 14,* $D^1(n') = \{y^6\}$*, and by steps 16-17,* $D^2(n') = D^2(n)$ *and* $D^3(n') = D^3(n)$*. Note that the local nadir weight for* n' *is equivalent to* G *(recall Table 4.1), which is the endpoint of whisker* $[F, G]$ *in Figure 4.3. Therefore, the basic Klamroth2015 algorithm's output would not indicate that* G *belongs to* $\Lambda(y^1)$ *and* misses the whisker $[F, G]$.

However, if input contributing sets, $\bar{C}^j(n)$*, are complete, then output contributing sets* $C^j(n)$ *will be complete and thus will include this whisker* $[F, G] \subseteq \Lambda(y^1)$*.*

As Algorithm 1 is an update procedure, we must clarify the initialized state. The following assumption applies to the application of the algorithm and all subsequent proofs.

Assumption 4. *We assume the initial set of ND images is* $\bar{Y}^0 := \{y^0, y^{b(1)}, \ldots, y^{b(p)}\}$*, i.e., all dummy images with one additional image; the initial set of maximal LNPs is* $\bar{N} = \{n^1, \ldots, n^p\}$ *where* $n^j := \{y^N(Y^0 \setminus \{y^{b(j)}\})\}$*; and for all* $j = 1..p$*, the complete, dimensional contributing sets for* n^j *are* $\bar{C}^j(n^j) := \{y^0\}$ *and* $\bar{C}^k(n^j) := \{y^{b(k)}\}$ *for all* $k \neq j$*.*

Note that by this initialization, each local nadir weight associated with n^i corresponds to a vertex of Λ, so essentially the weight set decomposition is initialized as $\Lambda(y^0) = \Lambda$.

For the following claims, let $\bar{Y} := \{y^1, \ldots, y^n\} \subset \mathcal{Y}_N \setminus Y_B$ be the set of first $n \geq 1$ ND

images input into Algorithm 1, which provides maximal LNPs $\bar{N}$ and complete contributing sets $\{\bar{C}^j(n)\}_{j=1..p,n\in\bar{N}}$ such that $\bar{C}^j(n) \subseteq \bar{Y}$ for all j, n. (For sake of readability, we do not explicitly denote the dependence on the image set $\bar{Y}$.) For update number $n+1$, image $\bar{y} \in \mathcal{Y}_N \setminus \bar{Y}$ is input along with $\bar{N}$ and $\{\bar{C}^j(n)\}_{j=1..p,n\in\bar{N}}$.

Theorem 5. *If input set $\bar{N}$ is maximal, $\bar{C}^j$ are nonempty, and $\bar{y}$ is ND, then the output set N is maximal.*

Proof. Proof given in Klamroth et al. [41]. □

It remains to be proven that the output contributing sets C^j are complete. Given image $\bar{y}$ and LNP n, it is impossible to simultaneously satisfy $\bar{y} < n$ (in step 2) and $\bar{y}_j = n_j$ (in step 4). Hence, during one call of the update algorithm, at most one of the following two operations will occur per $n \in \bar{N}$. First, when $\bar{y}$ is added to a contributing set of n (steps 5-6), the components of n remain unchanged. Second, when $n \in A$, there exists some $n' \in P$ (i.e., n is removed from $\bar{N}$ and new n' is added) such that $n' \leq n$ and for some j, $\bar{y}_j = n'_j < n_j$ and $D^j(n') = \{\bar{y}\}$. The following theorem shows that the relaxed conditions used for including images in $C^j(n)$ results in complete contributing sets.

Theorem 6. *Let $\bar{y}$, $\bar{N}$, and $\{\bar{C}^j(n)\}_{j=1..p,n\in\bar{N}}$ denote the input to Algorithm 1, and let N and $\{C^j(n)\}_{j=1..p,n\in N}$ denote the output. If $\bar{N}$ contains all maximal LNPs and the contributing sets $\{\bar{C}^j(n)\}_{j=1..p,n\in\bar{N}}$ are complete, then the output contributing sets, $C^j(n)$, are complete for all $j = 1, \ldots, p$ and $n \in N$.*

Proof. By definition of output set N in step 23, we have three cases for each $n \in N$: (1) $n \in \bar{N}$ and $\bar{y}$ does not dominate n, (2) $n \in \bar{N}$ and $\bar{y} \leq n$, or (3) $n \notin \bar{N}$. Case (1) is trivial: Steps 4-5 are irrelevant since the condition in step 4 does not hold for any $j \in \{1, \ldots, p\}$, and steps 7-19 are irrelevant since $n \notin A$. Since n is not dominated by $\bar{y}$, the image cannot contribute to any of the LNP's components. Hence if the input contributing sets $\{\bar{C}^j(n)\}_{j=1..p}$ are complete, then the output contributing sets $\{C^j(n)\}_{j=1..p}$ are also complete.

Observe that conditions in steps 4 and 8 are mutually exclusive. Therefore, the remaining cases essentially refer to either steps 4-6 or steps 8-22. Case (2) occurs when $\bar{y}$ dominates but not strictly dominates $n \in \bar{N}$, and therefore the contributing sets are updated without removing n. However, case (3) occurs when $\bar{y}$ strictly dominates some $n \in \bar{N}$, and thus new $n' \in P$ is defined with its contributing sets.

Case (2): Suppose $n \in \bar{N}$ and $\bar{y} \leq n$ with at least one equal component. Let $J := \{j = 1, \ldots, p : \bar{y}_j = n_j\}$, which is nonempty. By definition in step 2, $J \neq \emptyset$ implies $n \notin A$, so n is included in the output set N. Given the inequality in step 6, all $j \in J$ satisfy $\bar{y}_{-j} \leq n_{-j}$, so $\bar{y}$ is added to $C^j(n)$ for all j. Steps 8-22 are irrelevant. We must only guarantee that the image $\bar{y}$ is added to all necessary contributing sets. This follows from the for loop in step 4 iterating over all $j = 1, \ldots, p$. The result therefore holds.

Case (3): Suppose $\bar{y} < n$. By definition in step 2, $n \in A$. If $P = \emptyset$, then there is nothing to show (although this case never happens). Otherwise, for at least one $j \in \{1, \ldots, p\}$, $\bar{y}_j > m_j(n)$. Consider one such index j, and let n' be the corresponding LNP defined in step 12. Note that $n' \leq n$ and $\bar{y}_j = n'_j < n_j$. It remains to prove that the contributing sets, $\{C^k(n')\}_{k=1..p}$, are maximal.

First, observe that $C^j(n') = \{\bar{y}\} \cup \{y \in C^k(n) : y \leq n', y_j = \bar{y}_j, k = 1, \ldots, p\}$ (by step 15) is incomplete only if the initialization of $C^k(n)$ for some k is incomplete. Since $C^k(n)$ are initialized to be $\bar{C}^k(n)$ for all $k = 1, \ldots, p$, this leads to a contradiction.

Second, suppose for $k \neq j$ that $C^k(n')$ is not complete. By definition, $\bar{y} \in C^k(n')$. Then, there exists some $y' \in \bar{Y}$ such that $y'_k \leq n'$ and $y'_k = \bar{n}'_k$. Since $n'_k = n_k$, then $y' \leq n$ and $y'_k = n_k$, so y' contributes to n in the kth component. By assumption of completeness, $y' \in \bar{C}^k(n)$, and so we have a contradiction since $C^k(n)$ is initialized by $\bar{C}^k(n)$.

By exhausting all possibilities, we have that all contributing sets $\{C^k(n')\}_{k=1..p}$ are complete for all $n' \in P$. Thus, the claim is proven in all cases. □

Assumption 5. *Since we have now proven how to compute complete contributing sets, hereon we let $C^j(n)$ represent the complete, jth-dimensional contributing set for LNP n.*

4.5 Covering and Triangularization of Weight Set Components

We introduce a novel property which explicitly provides convex subsets of the weight set component. The following inductive argument relies on the star-shapedness of all intersection sets between weight set components.

Lemma 7. *For $\bar{Y} \subset \mathcal{Y}_N$ where $q := |\bar{Y}| \geq p$, let $y^N(\bar{Y})$ be weakly ND where all images contribute. Then,*

$$\operatorname{conv}(\{\lambda^N(y^1), \lambda^N(y^1, y^2) \ldots, \lambda^N(y^1, y^2, \ldots, y^q)\}) \subseteq \Lambda(y^1),$$

where $\lambda^N(y^1) = \lambda(y^1)$.

Proof. Let $\lambda = \sum_{r=1}^{q} \alpha_r \lambda^N(y^1, \ldots, y^r)$ with $\alpha \in [0,1]^q$ and $\sum_{r=1}^{q} \alpha_r = 1$. We show $\lambda \in \Lambda(y^1)$. Without loss of generality, we can assume that $\alpha_i > 0$ for all i. If λ is equal to a local nadir weight, $\lambda \in \Lambda(y^1)$ is trivially satisfied; otherwise, with zero coefficients, the following procedure can be adapted to the subset of nadir weights with strictly positive coefficients.

First, consider

$$\begin{aligned}
&\alpha_q \lambda^N(y^1, \ldots, y^q) + \alpha_{q-1} \lambda^N(y^1, \ldots, y^{q-1}) \\
&= (\alpha_q + \alpha_{q-1}) \left[\frac{\alpha_q}{\alpha_q + \alpha_{q-1}} \lambda^N(y^1, \ldots, y^q) + \frac{\alpha_{q-1}}{\alpha_q + \alpha_{q-1}} \lambda^N(y^1, \ldots, y^{q-1}) \right] \\
&=: (\alpha_q + \alpha_{q-1}) \tilde{\lambda}^{q-1}
\end{aligned}$$

for $\tilde{\lambda}^{q-1}$ defined as the convex combination of the q-way and $(q-1)$-way local nadir weights. By star-shapedness of $\bigcap_{r=1}^{q-1} \Lambda(y^r)$, which includes both the local nadir weights,

we have $\tilde{\lambda}^{q-1} \in \bigcap_{r=1}^{q-1} \Lambda(y^r)$. This implies

$$\begin{aligned}
&\alpha_q \lambda^N(y^1, \ldots, y^q) + \alpha_{q-1}\lambda^N(y^1, \ldots, y^{q-1}) + \alpha_{q-2}\lambda^N(y^1, \ldots, y^{q-2}) \\
&= (\alpha_q + \alpha_{q-1})\tilde{\lambda}^{q-1} + \alpha_{q-2}\lambda^N(y^1, \ldots, y^{q-2}) \\
&= (\alpha_q + \alpha_{q-1} + \alpha_{q-2}) \left[\frac{\alpha_q + \alpha_{q-1}}{\alpha_q + \alpha_{q-1} + \alpha_{q-2}} \tilde{\lambda}^{q-1} + \frac{\alpha_{q-2}}{\alpha_q + \alpha_{q-1} + \alpha_{q-2}} \lambda^N(y^1, \ldots, y^{q-2}) \right] \\
&=: (\alpha_q + \alpha_{q-1} + \alpha_{q-2})\tilde{\lambda}^{q-2}
\end{aligned}$$

for $\tilde{\lambda}^{q-2}$ defined as the convex combination of the $(q-1)$-way and $(q-2)$-way local nadir weights. Again, by star-shapedness we have $\tilde{\lambda}^{q-2} \in \bigcap_{r=1}^{q-2} \Lambda(y^r)$. Inductively applying this procedure yields

$$\begin{aligned}
\lambda =& \sum_{r=1}^{q} \alpha_r \lambda^N(y^1, \ldots, y^r) = (\sum_{r=2}^{q} \alpha_r)\tilde{\lambda}^2 + \alpha_1 \lambda^N(y^1) \\
=& (\sum_{r=1}^{q} \alpha_r) \left[\frac{(\sum_{r=2}^{q} \alpha_r)}{(\sum_{r=1}^{q} \alpha_r)} \tilde{\lambda}^2 + \frac{\alpha_1}{(\sum_{r=1}^{q} \alpha_r)} \lambda^N(y^1) \right] \\
=& \frac{(\sum_{r=2}^{q} \alpha_r)}{(\sum_{r=1}^{q} \alpha_r)} \tilde{\lambda}^2 + \frac{\alpha_1}{(\sum_{r=1}^{q} \alpha_r)} \lambda^N(y^1) \in \Lambda(y^1).
\end{aligned}$$

□

We define the family of polytopes presented in Lemma 7.

Definition 21. *For LNP $n = y^N(\bar{Y})$ with $\bar{Y} \subset \mathcal{Y}_N$, with dimensional contributing sets $C^j(n), j = 1, \ldots, p$, let $\mathcal{P}(n)$ denote the family of simplices*

$$\begin{aligned}
\mathcal{P}(n) = \{\text{conv}(\{&\lambda^N(y^{\sigma(1)}, \ldots, y^{\sigma(p-1)}, y^{\sigma(p)}), \lambda^n(y^{\sigma(1)}, \ldots, y^{\sigma(p-1)}), \ldots, \lambda^N(y^{\sigma(1)})\}) : \\
&y^j \in C^j(n), j = 1, \ldots, p, \textit{ for some permutation } \sigma : \{1, \ldots, p\} \to \{1, \ldots, p\}\}.
\end{aligned}$$

Example 17. *For Example 10, consider maximal LNP $n := y^N(\{y^1, y^2, y^{b(2)}, y^{b(3)}\}) = (1, M, M)$. From Table 4.1 (label C), we have $C^1(n) = \{y^1, y^2\}$, $C^2(n) = \{y^{b(2)}\}$, and $C^3(n) = \{y^{b(3)}\}$. Only one contributing set is not a singleton, i.e., $|C^1(n)| = 2$. Consider*

permutations $\sigma^1 = (1, 2, 3)$ *and* $\sigma^2 = (1, 3, 2)$. *The corresponding two polytopes belong to* $\mathcal{P}(n)$*:*

$$P^1 := \text{conv}\left(\lambda^N(\{y^1, y^{b(2)}, y^{b(3)}\}), \lambda^N(\{y^1, y^{b(2)}\}), \lambda^N(\{y^1\})\right)$$
$$P^2 := \text{conv}\left(\lambda^N(\{y^1, y^{b(2)}, y^{b(3)}\}), \lambda^N(\{y^1, y^{b(3)}\}), \lambda^N(\{y^1\})\right)$$

Using the labeled local nadir weights given in Figure 4.3(b), these polytopes are equivalent to triangles: $P^1 = \text{conv}(\{C, B, \lambda(y^1)\})$ *and* $P^2 = \text{conv}(\{C, D, \lambda(y^1)\})$

Polytopes P^1 *and* P^2 *are contained in* $\Lambda(y^1)$ *since* $\sigma^1(1) = \sigma^2(1) = 1$. *The remaining polytopes in* $\mathcal{P}(n)$ *are defined with the singleton local nadir weight of a boundary weight, i.e.,* $\sigma(1) = b(2)$ *or* $\sigma(1) = b(3)$. *Such polytopes are generally ignored.*

Subsets of weight set components can be constructed by the union of simplices given by $\mathcal{P}$. Theorem 7 states that the maximal LNPs are indeed sufficient to cover weight set components. The following Lemma is essential for the proof of Theorem 7.

Lemma 8. *Let* $\lambda \in \Lambda(y)$ *for some* $y \in \mathcal{Y}_{wN}$ *and* $I := \{1, \ldots, p : \|y\|_\infty^\lambda = \lambda_i y_i\}$ *with* $|I| < p$. *Then there exists* $\bar{y} \in \mathcal{Y}_N$, *weight* $\bar{\lambda} \in \Lambda$, *and index set* $J \subseteq \{1, \ldots, p\}$ *with* $I \cap J = \emptyset$ *such that the 1-6 below hold:*

1. $\bar{\lambda} \in \Lambda(y) \cap \Lambda(\bar{y})$,
2. $\lambda \in \text{conv}(\{\lambda^N(y), \bar{\lambda}\})$,
3. $\|y\|_\infty^{\bar{\lambda}} = \bar{\lambda}_i y_i$ *for* $i \in I$ *and* $\|y\|_\infty^{\bar{\lambda}} > \bar{\lambda}_k y_k$ *for* $k \notin I$,
4. $\|\bar{y}\|_\infty^{\bar{\lambda}} = \bar{\lambda}_j \bar{y}_j$ *for* $j \in J$ *and* $\|\bar{y}\|_\infty^{\bar{\lambda}} > \bar{\lambda}_k \bar{y}_k$ *for* $k \notin J$,
5. $y_i > \bar{y}_i$ *for all* $i \in I$, *and*
6. $y_j < \bar{y}_j$ *for all* $j \in J$.

Proof. Since $|I| < p$, $\lambda \neq \lambda^N(y)$. Set I must be nonempty, so let

$$H(I) := \{\lambda' \in \Lambda : \lambda'_i y_i = \|y\|_\infty^{\lambda'} \; \forall i \in I, \; \lambda'_k y_k < \|y\|_\infty^{\lambda'} \; \forall k \notin I\}.$$

By assumption, $\lambda \in H(I)$ and $\lambda^N(y) + \theta(\lambda - \lambda^N(y)) \in H(I)$ for all $\theta > 0$. Since $\Lambda(y)$ is compact and star-shaped, there exists $\theta' \geq 1$ such that

$$\bar{\lambda} := \lambda^N(y) + \theta'(\lambda - \lambda^N(y)) \in \Lambda(y),$$
$$\bar{\lambda}^\varepsilon := \lambda^N(y) + (\theta' + \varepsilon)(\lambda - \lambda^N(y)) \notin \Lambda(y)$$

for all $\varepsilon > 0$. For small $\varepsilon > 0$, let $\bar{y} \in \mathcal{Y}_N$ such that $\bar{\lambda}^\varepsilon \in \Lambda(\bar{y})$ ($\bar{y}$ may be in Y_B). Thus, by continuity of the weighted Tchebychev norm in λ, we have $\|y\|_\infty^{\bar{\lambda}} = \|\bar{y}\|_\infty^{\bar{\lambda}}$ and $\|y\|_\infty^{\bar{\lambda}^\varepsilon} > \|\bar{y}\|_\infty^{\bar{\lambda}^\varepsilon}$. This satisfies results 1 and 2.

For $i \in I$, $\|y\|_\infty^{\bar{\lambda}} = \bar{\lambda}_i y_i$ by definition. Suppose for contradiction that i also maximizes $\|\bar{y}\|_\infty^{\bar{\lambda}}$, i.e., $\|\bar{y}\|_\infty^{\bar{\lambda}} = \lambda_i \bar{y}_i$. Since $\bar{\lambda} \in H(I)$, this implies

$$\bar{\lambda}_i \bar{y}_i = \|\bar{y}\|_\infty^{\bar{\lambda}} = \|y\|_\infty^{\bar{\lambda}} = \bar{\lambda}_i y_i$$

and thus $\bar{y}_i = y_i$. However, $\bar{\lambda}^\varepsilon \in H(I)$ also holds and so $\|\bar{y}\|_\infty^{\bar{\lambda}^\varepsilon} \geq \bar{\lambda}_i^\varepsilon \bar{y}_i = \bar{\lambda}_i^\varepsilon y_i = \|y\|_\infty^{\bar{\lambda}^\varepsilon}$, which is a contradiction to $\lambda^\varepsilon \in \Lambda(\bar{y}) \setminus \Lambda(y)$. Therefore, $\|\bar{y}\|_\infty^{\bar{\lambda}} > \lambda_i \bar{y}_i$ and $y_i > \bar{y}_i$ for all $i \in I$. Let $J \subseteq \{1, \ldots, p\} \setminus I$ such that $\|\bar{y}\|_\infty^{\bar{\lambda}} = \bar{\lambda}_j \bar{y}_j$ for $j \in J$ and $\|\bar{y}\|_\infty^{\bar{\lambda}} > \bar{\lambda}_k \bar{y}_k$ for $k \notin J$. Then, $y_j < \bar{y}_j$ for all $j \in J$ follows by $\bar{\lambda}_j \bar{y}_j = \|\bar{y}\|_\infty^{\bar{\lambda}} = \|y\|_\infty^{\bar{\lambda}} > \bar{\lambda}_j y_j$ for all $j \in J$. Index sets I and J satisfies results 3-6. □

In summary, results 1 and 2 of Lemma 8 prove that the line segment $[\lambda(y), \lambda]$ may be extended linearly to an intersection of weight set components, i.e., $[\lambda(y), \lambda] \subseteq [\lambda(y), \bar{\lambda}]$ where $\bar{\lambda} \in \Lambda(y) \cap \Lambda(\bar{y})$ for some $\bar{y} \in \mathcal{Y}_N$. The index sets I and J indicate which components maximize $\|y\|_\infty^{\bar{\lambda}}$ and $\|\bar{y}\|_\infty^{\bar{\lambda}}$, respectively, and satisfy results 4-6. Note that the statement of the Lemma intentionally uses the set of *weakly* ND images, $\mathcal{Y}_{wN}$, so that the

lemma can be applied to LNPs in the following theorem.

Theorem 7. *For $\lambda \in \Lambda$, there exists a maximal LNP, $n = y^N(\bar{Y})$ for some $\bar{Y} \subseteq \mathcal{Y}_N$, such that there are p contributing images $y^1 \in C^1(n), \ldots, y^p \in C^p(n)$ with*

$$\lambda \in \text{conv}(\{\lambda^N(y^{\sigma(1)}), \lambda^N(y^{\sigma(1)}, y^{\sigma(2)}), \ldots, \lambda^N(y^1, y^2, \ldots, y^p)\}) \in \mathcal{P}(n)$$

for some permutation $\sigma : \{1, \ldots, p\} \to \{1, \ldots, p\}$.

Proof. Let $\lambda \in \Lambda$ and $y^1 \in \mathcal{Y}_N$ such that $\lambda \in \Lambda(y^1)$. By iteratively applying Lemma 8, one can construct a (not necessarily maximal) local nadir weight which is defined by $R \leq p$ ND images $y^1, \ldots, y^R$, each with respective index set $I^1, \ldots, I^R$ satisfying the following four properties:

1. $\lambda^N(\{y^1, \ldots, y^R\}) \in \bigcap_{r=1}^R \Lambda(y^r)$.
2. $\lambda \in \text{conv}(\{\lambda^N(y^1), \lambda^N(\{y^1, y^2\}), \ldots, \lambda^N(\{y^1, y^2, \ldots, y^R\})\})$.
3. $I^r \cap I^s = \emptyset$ for all $r, s \in \{1, \ldots, R\}$ where $r \neq s$, and $\cup_{r=1}^R I^r = \{1, \ldots, p\}$.
4. For all r it holds $y_i^r > y_i^s$ for all $i \in I^r$ and $r \neq s$.

The properties can follow as a corollary to Lemma 8, but for completeness it is proven in the construction section below. Here, assume we have constructed such a local nadir weight.

Note if there existed some $\bar{y} \in \mathcal{Y}_N$ such that $\bar{y} < y^N(y^1, \ldots, y^r)$, this would contradict property (1). Also, there exists a maximal LNP n such that $y^N(y^1, \ldots, y^r) \leqq n$. For an image y^r, suppose for contradiction $y_i^r < n_i$ for all $i \in I^r$. Since $y_k^r < y_k^s \leq n_k$ for $k \notin I^r$ and s chosen such that $s \in I^s$, it follows that $y^r < n$, which is a contradiction. Therefore, for each image y^r there exists an index i such that $y_i^r = n_i$ and thus $y^r \in C_i(n)$.

Without loss of generality, we have $y^r \in C_r(n)$ for $r = 1, \ldots, R$. If $R < p$, we can

choose images $\tilde{y}^{R+1} \in C^{R+1}(n), \ldots, \tilde{y}^p \in C^p(n)$ arbitrarily. Then, property (2) yields

$$\begin{aligned}\lambda &\in \operatorname{conv}(\{\lambda^N(y^1), \ldots, \lambda^N(y^1, y^2, \ldots, y^R)\}) \\ &\subseteq \operatorname{conv}(\{\lambda^N(y^1), \ldots, \lambda^N(y^1, y^2, \ldots, y^R), \\ &\qquad \lambda^N(y^1, y^2, \ldots, y^R, \tilde{y}^{R+1}), \ldots, \lambda^N(y^1, y^2, \ldots, y^R, , \tilde{y}^{R+1}, \ldots, \tilde{y}^p)\}) \in \mathcal{P}(n).\end{aligned}$$

Construction: It remains to construct the local nadir weight and its defining images satisfying properties (1)-(4). Let $I^1 \subseteq \{1, \ldots, p\}$ with $\|y^1\|_\infty^\lambda = \lambda_i y_i^1$ for all $i \in I^1$ and $\|y^1\|_\infty^\lambda > \lambda_k y_k^1$ for $k \notin I^1$. If $I^1 = \{1, \ldots, p\}$, the construction terminates. Otherwise, we obtain by Lemma 8 an image $y^2 \in \mathcal{Y}_N$, a weight $\lambda^2 \in \Lambda(y^1) \cap \Lambda(y^2)$, and an index set $I^2 \subseteq \{1, \ldots, p\}$ with

- $I^1 \cap I^2 = \emptyset$,
- $\lambda \in \operatorname{conv}(\{\lambda^N(y^1), \lambda^2\})$,
- $\|y^2\|_\infty^{\lambda^2} = \lambda_j^2 y_i^2$ for $i \in I^2$ and $\|y^2\|_\infty^{\lambda^2} > \lambda_k^2 y_k^2$ for $k \notin I^2$,
- $\|y^1\|_\infty^{\lambda^2} = \lambda_i^2 y_i^1$ for $i \in I^1$ and $\|y^1\|_\infty^{\lambda^2} > \lambda_k^2 y_k$ for $k \notin I^1$,
- $y_i^1 > y_i^2$ for all $i \in I^1$,
- $y_j^1 < y_j^2$ for all $j \in I^2$.

If $I^1 \cup I^2 = \{1, \ldots, p\}$, the construction terminates. Otherwise, we can apply Lemma 8 with $y^N(y^1, y^2)$, $I = I^1 \cup I^2$, and λ^2 to obtain an image $y^3 \in \mathcal{Y}_N$, a weight $\lambda^3 \in \Lambda(y^N(y^1, y^2)) \cap \Lambda(y^3)$ and an index set $I^3 \subseteq \{1, \ldots, p\}$ such that

- $I^2 \cap (I^1 \cup I^2) = \emptyset$,
- $\lambda^2 \in \operatorname{conv}(\{\lambda^N(y^1, y^2), \lambda^3\})$,
- $\|y^3\|_\infty^{\lambda^3} = \lambda_j^3 y_i^3$ for $i \in I^3$ and $\|y^3\|_\infty^{\lambda^3} > \lambda_k^3 y_k^3$ for $k \notin I^3$,

- $\|y^N(y^1, y^2)\|_\infty^{\lambda^3} = \lambda_i^3 y_i^N(y^1, y^2)$ for $i \in I^1 \cup I^2$ and $\|y^N(y^1, y^2)\|_\infty^{\lambda^3} > \lambda_k^3 y_k^N(y^1, y^2)$ for $k \notin I^1 \cup I^2$,
- $y_i^N(y^1, y^2) > y_i^3$ for all $i \in I^1 \cup I^2$,
- $y_j^N(y^1, y^2) < y_j^3$ for all $j \in I^3$.

This implies

- $I^r \cap I^s = \emptyset$ for $r, s \in \{1, 2, 3\}$,
- For all $r \in \{1, 2, 3\}$ it holds $y_i^r > y_i^s$ for all $i \in I^r$ and $s \in \{1, 2, 3\} \setminus \{r\}$,
- $\lambda^N(y^1, y^2, y^3) \in \Lambda(y^2) \cap \Lambda(y^2) \cap \Lambda(y^3)$,
- $\lambda \in \text{conv}(\{\lambda^N(y^1), \lambda^N(y^1, y^2), \lambda^N(y^1, y^2, y^3)\})$.

If $I^1 \cup I^2 \cup I^3 = \{1, \ldots, p\}$, the construction terminates. Otherwise, we can, again, apply Lemma 8 with $y^N(y^1, y^2, y^3), I = I^1 \cup I^2 \cup I^3$ and λ^3.

This construction step can be repeated at most $p - 1$ times until we obtain the desired set of images $y^1, \ldots, y^R$ and index sets $I^1, \ldots, I^R$. □

4.5.1 Triangularization and Plotting

For plotting weight set components, we now restrict ourselves to $p = 3$, and so we aim to plot a 2-dimensional representation of weight set components and compute their area.

Let N represent the set of maximal LNPs. We have proven so far that the union of families of simplices, $P^* := \{\mathcal{P}(n) : n \in N\}$ covers Λ. However, for a given weight set component, say $\Lambda(y)$ for $y \in \mathcal{Y}_N$, P^* will contain many duplicate simplices, and therefore it does not give a *minimal* representation of $\Lambda(y)$. In terms of computing the area of the weight set component, this is a computational hindrance. More critically, if the complete contributing sets were not known for the maximal LNPs, then $\Lambda(y)$ may not be fully

covered by P^*, even if Λ is. This section develops tailored procedure for computing a minimal representation of the weight set components by triangularization when given the set of maximal LNPs and complete contributing sets.

While there exist common subroutines to determine the convex hull of an input set of points, to our knowledge, there are no analogous routines for a "star-shaped hull." However, the highly structured geometry of the weight space components enables them to be triangulated in a very reasonable way. This partially employs Lemma 7, which shows that a q-way LNP implies lower-order LNPs. Therefore, the set of maximal LNPs and their complete contributing sets imply all other LNPs. In this section, we suppose this set of LNPs and maximal contributing sets are given. For ND set $Y' \subseteq \mathcal{Y}_N$, the set of maximal LNPs is denoted by N, and the dimensional contributing sets are denoted by $\{C^1(n), C^2(n), C^3(n)\}_{n \in N}$. These sets may be computed by Algorithm 1.

The pseudocode for the following algorithm is given in Algorithm 2. For each $y \in Y' \subseteq \mathcal{Y}_N$, the perimeter of $\Lambda(y)$ is composed of maximal LNPs and lower-order LNPs. Therefore, the set of maximal LNPs to which y contributes (M in step 2) is used to generate all the "implied" lower-order LNPs to which y also contributes; this set of LNPs is generated in a recursive-style function and is denoted by L (step 3).

The perimeter of $\Lambda(y)$ is then computed counter-clockwise as three subsets defined by the dimensional contributing sets C^1, C^2, and C^3. The fundamental approach for each of these contributing sets follows the same three steps. Let $i \in \{1, 2, 3\}$. First, the subset of L for which y belongs to C^i is determined (e.g., step 5). Since all LNPs in this set are equal in the ith component, then we can reduce dimension to a biobjective perspective for the two subsequent steps. Second, for LNPs $n, n' \in L$, if $n < n'$ in the biobjective view, then the dominating LNP, i.e., n, is removed. This function, denoted by Remove-Dominating(*set*, *dimensions*), e.g., in step 6. Finally, the LNPs are ordered in the natural ordering for biobjective problems: decreasing in one objective and increasing in the other (e.g., step 7). Taken together, these steps return the *upper envelope* of the LNPs and, in

Algorithm 2 Plot weight set decomposition

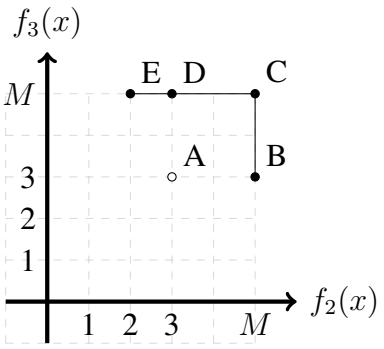
Input: Subset of ND frontier $Y' \subseteq \mathcal{Y}_N$, maximal LNPs $N \subset \mathbb{R}^3$ with associated contributing sets $\{C^1(n), C^2(n), C^3(n)\}_{n \in N}$.

Output: For all weight set components, a set of triangles used to plot and cumulative computed area.

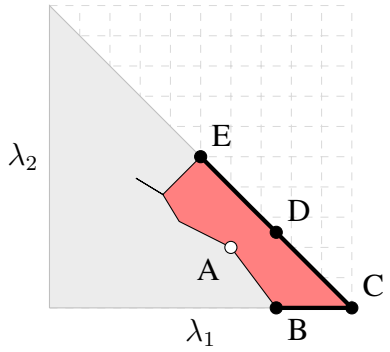
1: **for** $y \in Y'$ **do**
2: $M \leftarrow \{n \in N : \ y \in C^1(n) \cup C^2(n) \cup C^3(n)\}$ (set of maximal LNPs to which y contributes)
3: $L \leftarrow$ImpliedLNPs(M, y)
4: # Perimeter defined by C^1:
5: $L^1 \leftarrow \{n \in L : y \in C^1(n)\}$
6: $L^1 \leftarrow$RemoveDominating$(L^1, (y_2, y_3))$
7: $L^1 \leftarrow$Order$(L^1, (-y_2, y_3))$
8: # Perimeter defined by C^2:
9: $L^2 \leftarrow \{n \in L : y \in C^2(n)\}$
10: $L^2 \leftarrow$RemoveDominating$(L^2, (y_3, y_1))$
11: $L^2 \leftarrow$Order$(L^2, (-y_3, y_1))$
12: # Perimeter defined by C^3:
13: $L^3 \leftarrow \{n \in L : y \in C^3(n)\}$
14: $L^3 \leftarrow$RemoveDominating$(L^3, (y_1, y_2))$
15: $L^3 \leftarrow$Order$(L^3, (-y_1, y_2))$
16: # Join perimeters and triangulate:
17: $P \leftarrow$ join(L^1, L^2, L^3)
18: **for** $(n^i, n^{i+1}) \in P$ **do**
19: $T \leftarrow \text{conv}\{\lambda(n^i), \lambda(n^{i+1}), \lambda(y)\}$
20: **if(**T **duplicated) continue**
21: $\Lambda(y) \leftarrow \Lambda(y) \cup T$
22: area$(y) \leftarrow$area$(y)+$area(T)
23: **end for**
24: **end for**
25: **Return** $\{\Lambda(y)\}_{y \in Y'}$ and $\{\text{area}(y)\}_{y \in Y'}$

effect, provides the outer-most description for triangulation of the weight set component. The resulting ordered set of LNPs is denoted as L^i. Figure 4.6 illustrates the procedure and results from the biobjective perspective.

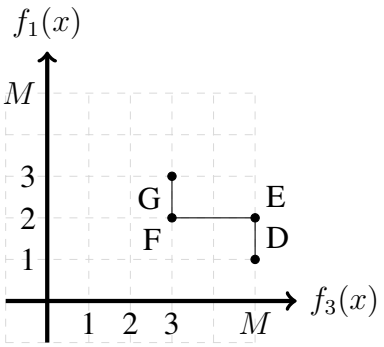
Once L^1, L^2, and L^3 are computed, they are joined and denoted as P (step 17) which is sufficient to describe the full perimeter. The remaining loop (steps 18-22) uses every neighboring pair of LNPs in P to define a triangle (with the kernel weight as the third vertex). Step 19 checks if this triangle is a duplicate (i.e., already been added to this weight set component): if so, then this triangle is discarded. Otherwise, the triangle is added to the description of $\Lambda(y)$, and its area is added to the estimated area for the component (steps 21 and 22, respectively). Note that when $n^i = n^{i+1}$, the triangle is trivially small with area equal to zero. Furthermore, the loop in step 18 should include the triangle between $P.last$



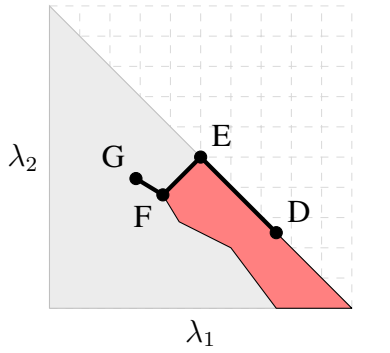
(a) Initial set L^1 projected onto (f_2, f_3). LNP A is strictly dominates C and is removed.



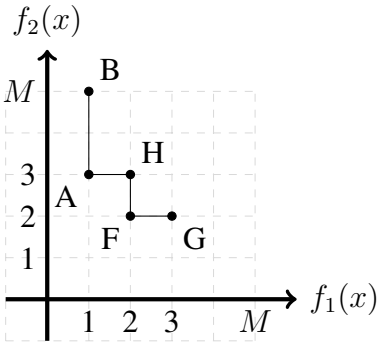
(b) Perimeter of $\Lambda(y^1)$ defined by local nadir weights from the final L^1.



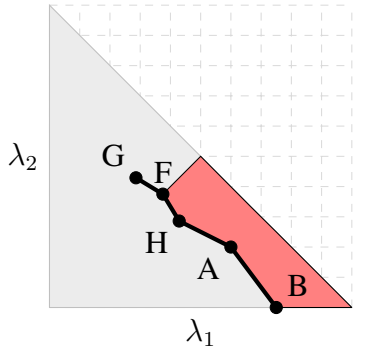
(c) Set L^2 projected onto (f_3, f_1) and ordered.



(d) Perimeter of $\Lambda(y^1)$ defined by local nadir weights from L^2.



(e) Set L^3 projected onto (f_1, f_2) and ordered.



(f) Perimeter of $\Lambda(y^1)$ defined by local nadir weights from L^3.

Figure 4.6: Computing the perimeter of weight set components using biobjective projections of L^1, L^2, L^3 sets.

and $P.first$. Once all loops are complete, the algorithm returns the weight set components and estimated area for each (step 24).

Example 18. *Consider computing the weight set component $\Lambda(y^1)$ in Example 10. We use the labels for the LNPs and local nadir weights interchangeably. Table 4.1 gives the four maximal LNPs, labeled C, E, G, and H. In Algorithm 2 step 2, $M = \{C, E, G, H\}$. Step 3 generates the set of implied LNPs $L = \{A, B, C, \ldots, G, H\}$, as labeled in Figure 4.3.*

The projection and ordering of L^1, L^2, L^3 sets are illustrated in Figure 4.6. First, consider $L^1 = \{A, B, C, D, E\}$ since these LNPs share the first component with y^1. These LNPs projected onto the (f_2, f_3) biobjective space in Figure 4.6(a). LNP A strictly dominates LNP C, so it will not be included in this section of the perimeter. After removing C from L^1, the LNPs (and local nadir weights) are ordered B, C, D, E. Figure 4.6(b) highlights the resulting portion of the perimeter of $\Lambda(y^1)$.

Figure 4.6(c)-(e) illustrates the procedure for L^2 and L^3. We note some additional observations:

- *The next sequence of LNPs, L^2, includes D and E again, which means that when joined, the sequence (D,E,D,E) occurs. Extra triangles are detected by step 19 to prevent from "double counting" these areas.*
- *The whisker $[F, G]$ is captured by the terminal sequence $(F, G) \in L^2$ followed by the initial sequence $(G, F) \in L^3$.*
- *While LNP A is removed from L^1, it remains in L^3 and therefore contributes to the final perimeter.*

We have implemented a plotting tool in R using basic functions and the ggplot2 package. The triangles are plotted as transparent ("alpha" parameter set to 0.5) so that full-dimensional intersections between weight set components are shown visibly as overlapping regions, e.g., Figure 4.4(b). In addition to computing each triangle, the outline of the

weight set component is represented as line segments of the form $[\lambda(n^i), \lambda(n^{i+1})]$ for each $n^i, n^{i+1} \in P$.

4.6 Application to Primal Algorithms

Section 4.4 presented how to modify a primal algorithm used to update maximal LNPs. Now, motivated by the geometry of the weighted Tchebychev weight set decomposition, we illustrate applications of the dual perspective for primal algorithms used to find ND images. We begin with a proper comparison between weight set decompositions for the two scalarizations, weighted sum and weighted Tchebychev , in order to motivate the strengths of the latter (Section 4.6.1). Then, Section 4.6.3 presents how outer and inner approximations of weighted Tchebychev weight set components can be computed during run time of a primal algorithm. Finally, Section 4.6.4 presents how to restrict computation of the primal algorithm to return a subset of the ND frontier associated with a subset of the weight set.

Two instances of ND sets are used, each defined as a (MOP) instances with 25 variables. Weight set decompositions for the first instance have already been presented in Figure 4.1. Weight set decompositions for the second instance are illustrated in Figure 4.7; this instance is the focus of the remaining sections due to its small number of ND images.

4.6.1 Comparison to Weighted Sum Decomposition

In addition to the instances in Figures 4.1 and 4.7, three additional instances are used to compare weight set decompositions; the additional weight set decompositions are illustrated in Figure B.2 of the Appendix. For each instance and each ND image, the area assigned to the corresponding weight set component are compared between the two decompositions in Figure 4.8 with paired bar graphs. Note the proportion of ND images that are ESND: on average, only 28% of the ND images are extreme supported (range 23%-36%). From these graphs, we make two observations.

First, while the weight set component with maximum area in each decomposition cor-

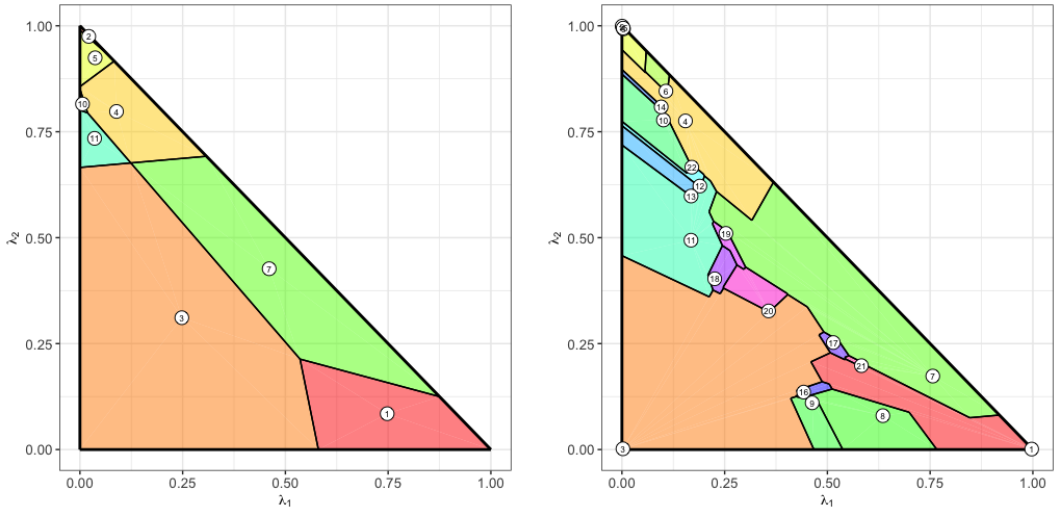
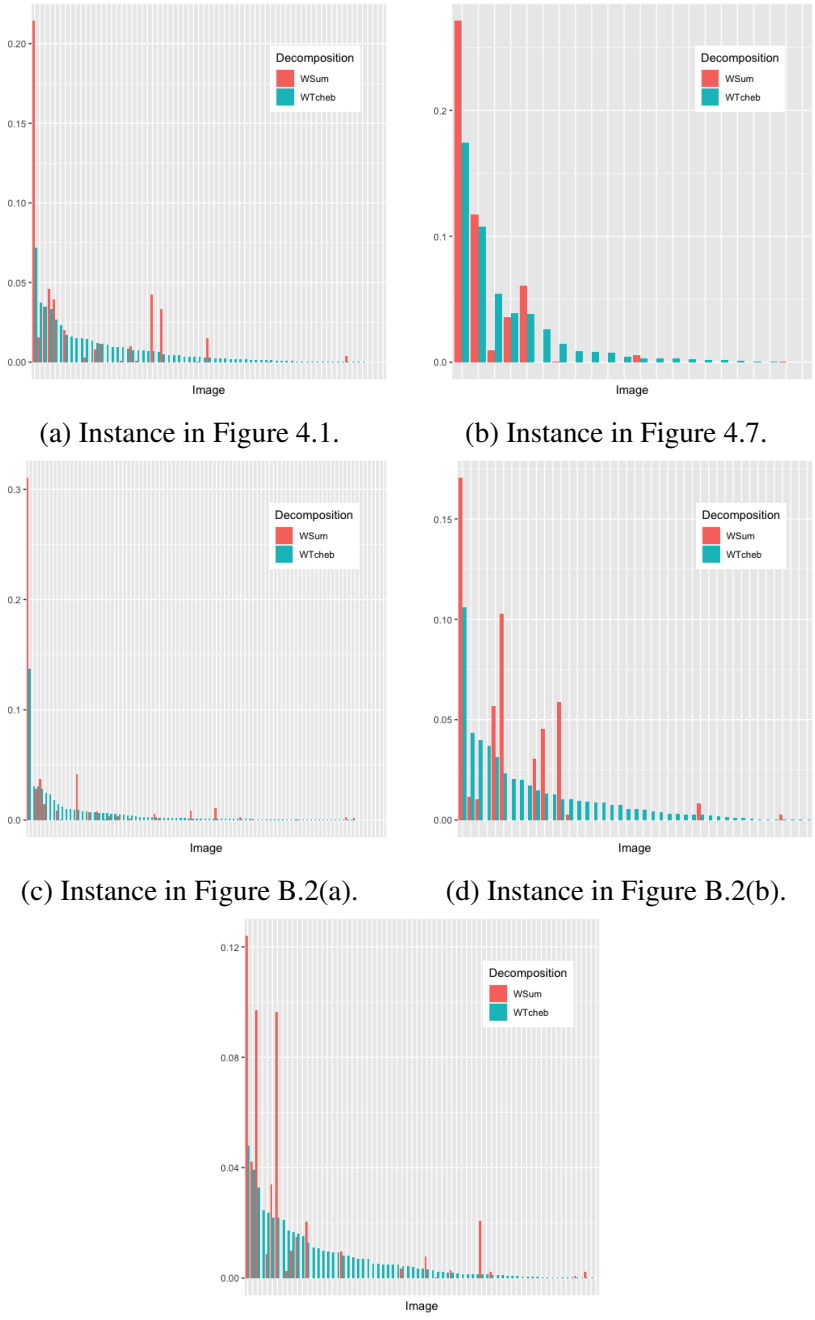


Figure 4.7: Comparing weight set decompositions with respect to weighted sum scalarization (left) and weighted Tchebychev scalarization (right) for a sample ND set with 22 ND images, including 8 ESND images. Color key is consistent between pairs. Continued in Figure B.2 in the Appendix.

responds to the same image, the magnitude of the difference between the largest weight set component and other components is significantly larger for the weighted sum scalarization. That is, the weighted sum decomposition inflates the area or the relative value of its top image compared to the weighted Tchebychev decomposition. This observation can be partially explained by simply having fewer ND images included in the decomposition; however, the following observation indicates that this alone does not fully explain this difference.

Second, beyond the top image, the ranked orders of the area per weight set component do not match between decompositions, and there are notable, extreme differences. Figure 4.8(a) and (e) include at least one image in the right tail that (1) comprises a small portion of the weight space in the weighted Tchebychev decomposition while (2) occupying a much larger space in the weighted sum decomposition. We argue this is a second major weakness of decomposing the weight set with respect to a subset of the ND set: as ND images are omitted, the crucial adjacency structure is significantly altered. This leads to disproportionate gains in some of the remaining images. With the weight set de-



(a) Instance in Figure 4.1.

(b) Instance in Figure 4.7.

(c) Instance in Figure B.2(a).

(d) Instance in Figure B.2(b).

(e) Instance in Figure B.2(c).

Figure 4.8: Paired bar graphs compare weight set component area for decompositions with respect to weighted sum scalarization (red) and weighted Tchebychev scalarization (blue) for sample ND sets. The y-axis represents area, and the (pairs of) bars are ordered in descending order of area for the weighted Tchebychev decomposition.

composition for weighted Tchebychev now available, which includes all ND images, this disproportionate reallocation becomes clear.

Algorithm 3 Primal Algorithm, modified from [42].

Input: Image set $\mathcal{Y} \in \mathbb{R}^3_>$.
Output: Nondominated set $\mathcal{Y}_N \subseteq \mathcal{Y}$.
1: $\mathcal{Y}_N \leftarrow \emptyset$
2: $\mathcal{B}_1 \leftarrow$\{InitStartingBox($\mathcal{Y}$)\}
3: Step $s \leftarrow 1$
4: **while** $\mathcal{B}_s \neq \emptyset$ **do**
5: Choose $B \in \mathcal{B}_s$
6: Solve for $y^s := opt(\mathcal{Y}, u(B))$
7: **if** $y^s = \emptyset$ **then**
8: $\mathcal{B}_{s+1} \leftarrow \mathcal{B}_s \setminus B$
9: **else**
10: $\mathcal{Y}_N \leftarrow \mathcal{Y}_N \cup \{y^s\}$
11: $\mathcal{B}_{s+1} \leftarrow$ UpdateBoxes($\mathcal{B}_s, B, y^s$)
12: **end if**
13: $s \leftarrow s + 1$
14: **end while**
15: **Return** $\mathcal{Y}_N$

4.6.2 Primal Algorithm for Finding ND Set

In Section 4.2.1, we introduced the algorithms following the primal approach to solving for all ND images in image space [42, 41, 44, 46]. The general structure of these primal algorithms is the same, which is summarized by the pseudocode given in Algorithm 3.

The primal algorithm takes an image set $\mathcal{Y}$ and returns the ND set $\mathcal{Y}_N$ (it is understood that $\mathcal{Y}$ is not known explicitly but can be represented implicitly by a MODO formulation). Initialization includes defining an empty ND set (step 1) and the initial search region as a box (step 2), which requires minimizing each of the individual objectives so that the ideal point is available. Boxes are contained within a queue, called $\mathcal{B}_s$ for step s, from which a new box is pulled from to process (step 5) and new boxes are added for future processing (steps 11-13). The algorithm continues until an empty queue is reached.

For every step s, the algorithm proceeds as follows: First, given a box B from queue $\mathcal{B}_s$, an integer program (IP)[7] is solved with a black box solver (represented by opt in step 6) for a new ND image, if one exists. If no ND image is found (equivalently, the interior of

[7]The precise form of the integer program differs depending on the primal algorithm used. An example is a weighted sum scalarization with additional constraints given by the objectives bounded above by the maximum LNP.

the box is empty, the IP is infeasible, and the LNP is weakly ND), then the box is removed from the queue (steps 7-8). Otherwise, if a new ND image is found, denoted by y^s, then y^s is added to $\mathcal{Y}_N$ (step 10), and the queue is updated to compute the next queue, $\mathcal{B}_{s+1}$ (step 11). Computing the next queue often involves splitting B, and any other $B' \in \mathcal{B}_s$ such that $y_s \in B'$, into new boxes that eliminate the region dominated by y^s but contain the necessary regions to be searched in the future; previous primal algorithms use different approaches to this [41, 42, 44]. In particular, the boxes can be defined by the maximal LNPs defined by the update method presented in Algorithm 1 or its extension discussed in Section 4.4. Before terminating, the primal algorithm deletes all boxes that are removed from the queue and only returns the set of all ND images (step 15).

The elimination of an empty box in step 8, along with its maximal LNP, is the notable difference between the primal and dual paradigms. While the primal algorithm is designed only to return the ND images, the dual approach also seeks the maximal LNPs defining these boxes, especially "empty" ones. When no new ND image is found in a box, then its associated maximal LNP is weakly ND with respect to $\mathcal{Y}_N$ and should therefore be returned for use in the weight set decomposition. The queue of boxes should then be understood as the current list of maximal LNPs. Each LNP has a flag to represent whether it has been certified as weakly ND or not; for new LNPs, this flag is set to *false*. In step 5, a LNP is only chosen from the queue if it is flagged as *false*. In step 8, instead of removing a maximal LNP, it is flagged as *true*. Therefore, most primal algorithms can be modified to return a set of weakly ND, maximal LNPs with the only additional effort being memory storage.

In summary, at each step of the primal algorithm, either a new ND image is discovered or a maximal LNP is certified as weakly ND. In general, the most costly procedure at every step is solving the IP optimization problem (step 6).

Example 19. *The progression of Algorithm 3 on the ND set from Example 10 is illustrated in Table 4.2. We let the boxes in the primal algorithm be chosen in step 5 by naive, first-*

in-first-out order. Algorithm 1 is used in step 11 to update maximal LNPs and complete contributing sets. The algorithm requires 16 steps before the queue is empty, which terminates the procedure.

Table 4.2: Progression of primal algorithm (Algorithm 3) on Example 10.

Step	ND Image found	LNP certified weakly ND
1	y^1	
2		$(1, M, M)$
3	y^4	
4	y^2	
5		$(2, 2, M)$
6		$(M, 1, M)$
7	y^3	
8	y^5	
9		$(2, 3, 3)$
10	y^6	
11		$(2, M, 2)$
12	y^7	
13		$(M, M, 1)$
14		$(3, 2, 3)$
15		$(3, 3, 2)$
16		$(M, 2, 2)$

4.6.3 Approximation of the Weight Space Decomposition

Before the Algorithm 3 terminates, which provides sufficient and necessary information for the weighted Tchebychev weight set decomposition, the partial information can be used for an approximation of the ND set and associated weight set components. Approximations of the weight set components during run time offer two valuable applications. First, the order of boxes chosen (step 5 in Algorithm 3) can be informed by the approximate area of the weight set components, e.g., choosing the box defined by the LNP whose contributing images have largest summed area of weight set components. Second, primal algorithms can be evaluated against each other by metrics that evaluate the approximation of the weight space components. For evaluating the quality of an approximate weight space decomposition, [51] suggests to use the Hausdorff distance.

To approximate weight set components, we provide both an outer and an inner approximation scheme, analogous to the approaches of [23] and [49], respectively, for weighted sum weight set decomposition. An *outer approximation*, $\{\Lambda^+(y)\}_{y\in Y'}$ for $Y' \subset \mathcal{Y}_N$, for a weight set decomposition $\{\Lambda(y)\}_{y\in\mathcal{Y}_N}$ is an approximation for a proper subset of the images where the approximated weight set components are supersets of the true components, i.e., $\Lambda(y) \subseteq \Lambda^+(y)$ for all $y \in Y'$. On the other hand, an *inner approximation*, $\{\Lambda^-(y)\}_{y\in Y'}$ is such that the approximated weight set components are subsets of the true components, i.e., $\Lambda^-(y) \subseteq \Lambda(y)$ for all $y \in Y'$.

Outer approximation.

In general, an outer approximation can be achieved trivially by following the standard steps for computing the weighted Tchebychev weight set decomposition but for a proper subset of the ND set. For instance, given $Y' \subset \mathcal{Y}_N$ from the primal algorithm (Algorithm 3) and the updated set of maximal LNPs with complete contributing sets, computed by Algorithm 1, then the plotting algorithm (Algorithm 2) will triangularize and plot the outer approximation. When integrated into the primal algorithm, the outer approximation is computed with all the maximal LNPs available at the current step, regardless if they are maximal or weakly ND with respect to the complete ND set. At each step of the primal algorithm, when a new ND image is found, then the set of maximal LNPs and complete contributing sets are updated, and the outer approximation of each component either reduces or stays the same.

Example 20. *Figure 4.9 illustrates the outer approximation of weight set components for the instance illustrated in Figure 4.7.*

Inner approximation

The inner approximation is less straightforward to compute. Given $Y' \subset \mathcal{Y}_N$ from Algorithm 3 and the updated set of maximal LNPs computed by Algorithm 1, then at least one

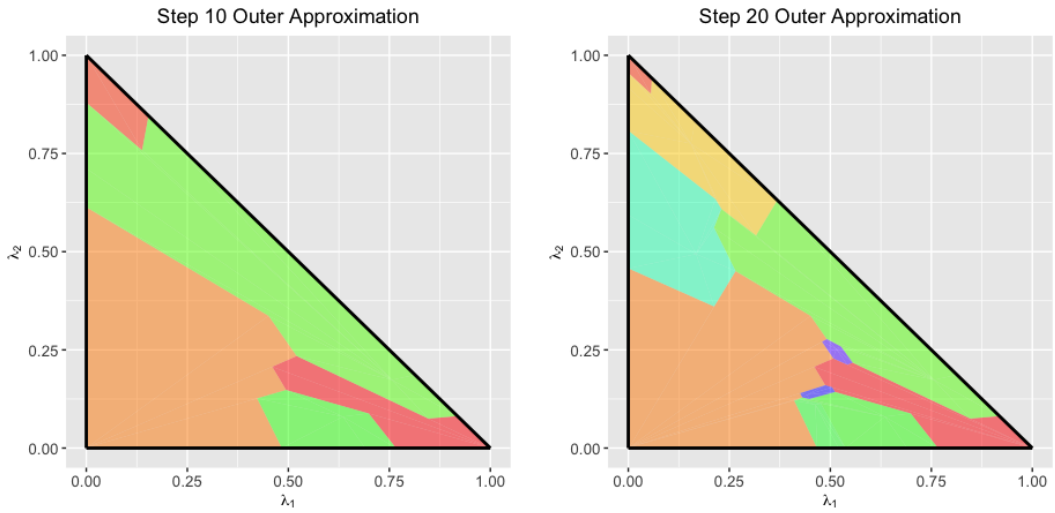


Figure 4.9: Outer approximations are illustrated for the example in Figure 4.7(b). On the left, the weight set decomposition is computed at step 10 of Algorithm 3, which includes 5 ND images. On the right, the weight set decomposition is computed at step 20, which includes 10 ND images. The colors assigned to each image's weight set component are consistent with Figure 4.7.

maximal LNP must be certified as weakly ND. When integrated into the primal algorithm, the inner approximation is computed with only these weakly ND, maximal LNPs certified at the current step. Note that these LNPs will be weakly ND with respect to the full ND set even if their contributing sets are not complete with respect to the full ND set.

With the subset of maximal LNPs that are certified as weakly ND, the plotting algorithm (Algorithm 2) must be modified. Often, an inner approximation will not contain the full perimeter of the component but rather disconnected subsets of the perimeter. Therefore, triangularization must occur independently for each dimensional contribution set, C^1, C^2, C^3; hence, in Algorithm 2, the sets L^1, L^2, L^3 are not joined (step 17), and the for loop (steps 18-23) is performed over each L^i set, independently.

At each step of the primal algorithm, when an empty box is found, then the set of certified LNPs is increased by one, and the inner approximation of the contributing images' weight set components may increase (or stay the same). Furthermore, since the primal algorithm using Algorithm 1 computes maximal LNPs with *complete* contributing sets, then it is possible to check when a single image's weight set component can be fully computed:

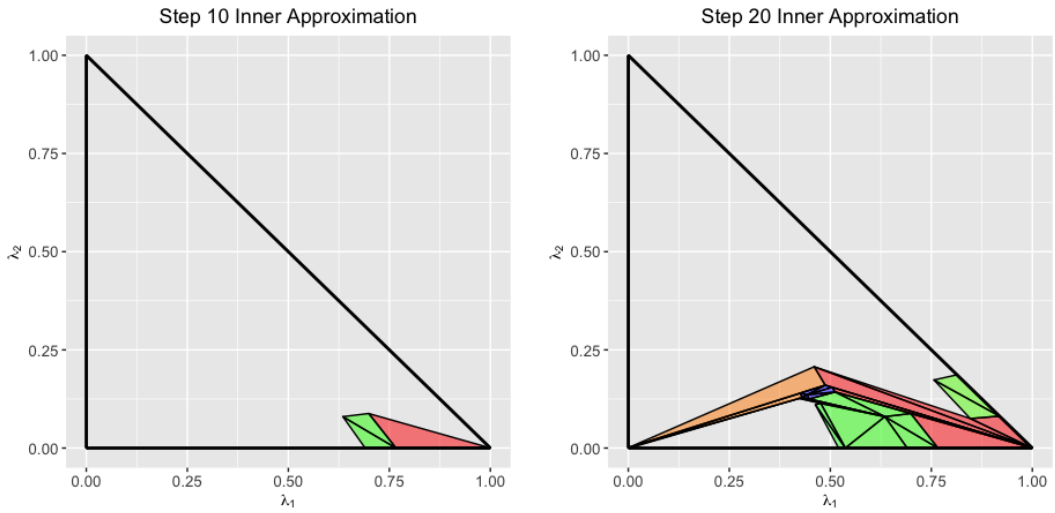


Figure 4.10: Inner approximations are illustrated for the example in Figure 4.7(a). On the left, the weight set decomposition is computed at step 10 of the primal algorithm, which includes 5 ND images and only 4 certified weakly ND LNPs. One LNP implies lower-order LNPs that define all three visible triangles. The other three LNPs are near the vertices of Λ, so the resulting triangles are miniscule and therefore not visible. On the right, the weight set decomposition is computed at step 20 of the primal algorithm, which includes 10 ND images and 9 certified weakly ND LNPs. The colors assigned to each image's weight set component are consistent.

if all maximal LNPs to which image y' contributes are certified as weakly ND, then $\Lambda(y')$ can be computed in full. Otherwise, if y' contributes to some LNPs that are yet to be certified, then $\Lambda(y')$ cannot be computed in full.

Example 21. *Figure 4.10 illustrates the inner approximation of weight set components for the instance illustrated in Figure 4.7. Note that at step 20, the inner approximation $\Lambda^-(y^8)$ (in green) is nearly equal to the outer approximation $\Lambda^+(y^8)$ in Figure 4.9. However, since y^8 contributes to one of the remaining uncertified LNPs, then $\Lambda^-(y^8)$ is not yet complete.*

Approximation time series.

At each step of Algorithm 3, the inner and outer approximation for each image's weight set component can be computed along with its area. This area can be plotted for each step of the algorithm as a time series. An example is given in Figure 4.11 for the algorithm

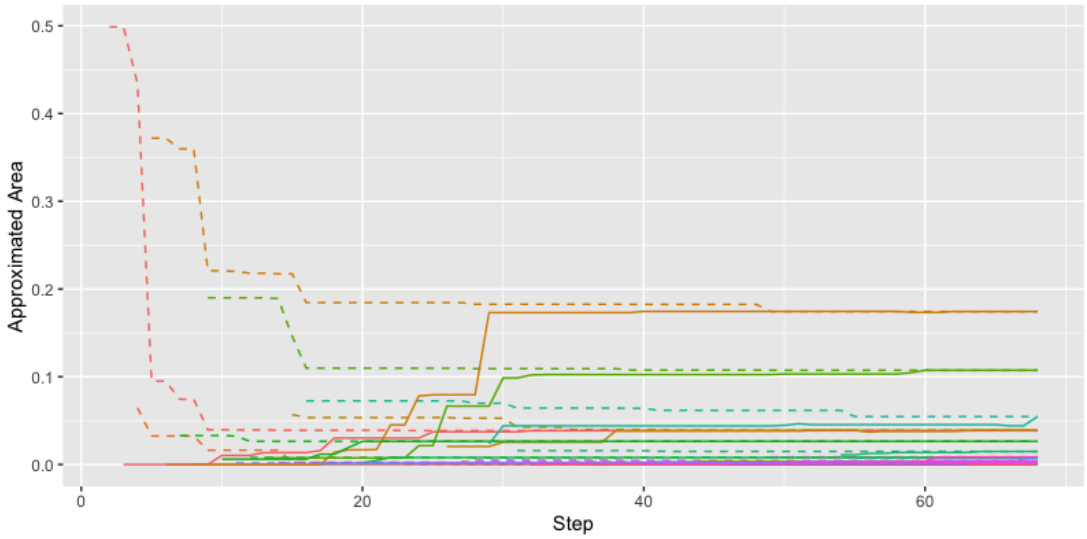


Figure 4.11: Area of the approximated weight set components for each image are plotted as a time series. The x-axis represents the step of the primal algorithm. Outer approximated areas are indicated by dashed lines and are monotonically decreasing. Inner approximated areas are indicated by solid lines and are monotonically increasing.

run on the instance in Figure 4.7. The outer approximated and inner approximated areas are indicated by lines (dashed and solid, respectively), and the x-axis represents the step s in Algorithm 3. The outer approximated area is monotonically decreasing and acts as an upper bound for the true area of the weight set component. The inner approximated area is the opposite: monotonically increasing and acts as a lower bound. When Algorithm 3 terminates, the gap is zero for all weight set components. Figure 4.11 motivates additional insights.

First, this time series analysis can be used as a *simulated* computational study, which is conducted with a known ND set and without solving IPs. It has been observed that infeasible subproblems require longer solve times than feasible subproblems [46]. However, the variety of black box IP solvers implies varied performance for solving the same subproblems. This justifies the proposed analysis of primal algorithms which is independent of differing performances between black box IP solvers.

Second, valuable data are easily accessible by this time series analysis: the ranked order of weight set components based on outer- or inner-approximated area as well as the gap

between these two estimates. Performance metrics may be designed around these data in order to evaluate approximations of primal algorithms. For instance, the gap between outer and inner approximations would ideally be minimized for high-performing algorithms.

Lastly, this analysis highlights the critical ordering of boxes from the queue chosen by step 5 of Algorithm 3. Note that for analyzing the terminal performance of a primal algorithm, the choice of the box does not effect the accuracy of the algorithm; however, the choice is critical to the quality of the approximation prior to termination. For example, it is common to use the hypervolume measure to choose the box based on largest hypervolume. Notably, the algorithm by [46] includes a heuristic of this type which is designed to improve the selection of the subproblem. Now with the approximated areas of the weight space components available, either or both approximations can be used to objectively improve this choice.

4.6.4 Compromise Region

Consider there are $N \geq 3$ decision makers, each with their own individual weight vectors for the three objectives, $\lambda^1, \ldots, \lambda^N \in \Lambda$. Solving $\Pi^{TS}(\lambda^i)$ for each weight λ^i identifies images $y^1, \ldots, y^N \in \mathcal{Y}_N$ such that $\lambda^i \in \Lambda(y^i)$. However, the union of these weight set components does not necessarily cover the set conv$(\lambda^1, \ldots, \lambda^N)$, which we call the *compromise region*. Let Λ^{CR} :=conv$(\lambda^1, \ldots, \lambda^N)$, and assume dim$(\Lambda^{CR}) = 2$. Determining the set of ND images whose weight set components cover Λ^{CR} requires finding all the weight set components that intersect with Λ^{CR}. Once $\mathcal{Y}_N$ is known, this is trivial via postprocessing; however, Algorithm 3 may be modified to use Λ^{CR} *a prior* to reduce the computation while still achieving coverage.

Note that the set-up of this problem aligns with the problem definition for the McRow method [30]. However, the McRow method is defined for continuous feasible sets, and even when applied to MODO, it returns a single solution which minimizes the worst-case outcome for a single decision maker. Our approach returns all of the ND images in the

region of interest from the image set frontier.

The initialization (step 2) of Algorithm 3 must be extended to solve for each $i = 1, \ldots, N$, $\Pi^{TS}(\lambda^i)$ augmented with small ϵ times sum of the objectives in order to return a ND image. The update algorithm, Algorithm 1, is called with each of the resulting (unique) images $y^1, \ldots, y^N$ to compute an updated set of maximal LNPs with complete contributing sets. This gives the initial queue with boxes defined by the maximal LNPs.

Subsequently, boxes are chosen from the queue to be processed; however, a box can be discarded instead if it is known that, for its maximal LNP n, $\mathcal{P}(n) \cap \Lambda^{CR} = \emptyset$. Testing this requires both checking if the "vertices" (i.e., local nadir weights) of the polytopes belong to Λ^{CR} as well as testing if an "edge" between vertices intersects with Λ^{CR}. This can be checked with a recursive-style boolean function. If the maximal LNP leads to intersection with Λ^{CR}, then the box is processed as normal. Otherwise, the LNP may be flagged as certified weakly ND so that the algorithm does not process it any further.

Example 22. *A simple compromise region is given by*

$$\Lambda^{0.5} = \{\lambda \in \Lambda : \lambda_i \leq 0.5 \; \forall i = 1, 2, 3\} = \text{conv}\left(\{(0, 0.5, 0.5), (0.5, 0, 0.5), (0.5, 0.5, 0)\}\right).$$

Figure 4.12(a) illustrates $\Lambda^{0.5}$ in weight space: observe that this inverted triangle has a quarter of the area of Λ, and it represents the ND images that are most "balanced" with respect to the three objectives.

Algorithm 3 restricted to compromise region $\Lambda^{0.5}$ for the example instance in Figure 4.7 is shown in Figure 4.12(b). Clearly from this example, each of the weight set components that intersect with $\Lambda^{0.5}$ are discovered; compared to the full decomposition, the perimeters of these components are correct within $\Lambda^{0.5}$. During processing, some ND images whose weight set components do not intersect with $\Lambda^{0.5}$ are also found, e.g., the yellow component labeled 4 in Figure 4.12(b). This occurs due to long edges extending outside of the compromise region, so these images should be removed via postprocessing.

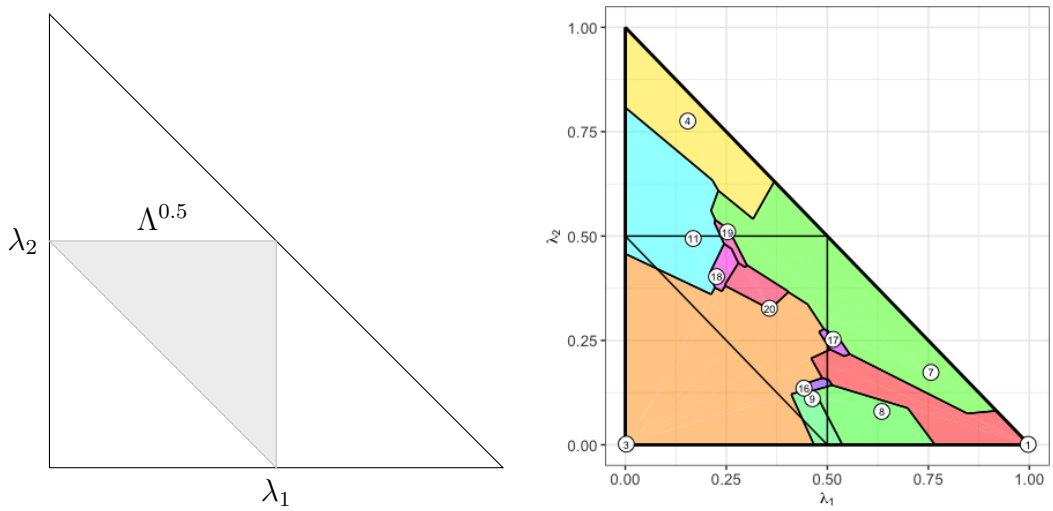


Figure 4.12: A sample compromise region is illustrated (left), and the output of the primal algorithm when restricted to this compromise region (right).

The reduction in computational effort is proportionally larger than the reduction in output size. Out of the 22 total ND images in this instance, only 12 are returned (54.5%). Whereas the full primal algorithm requires 68 optimization subproblems, this restriction only requires 28 (41.2%). This reduction is achieved because 7 maximal LNPs are essentially "skipped" without solving an optimization subproblem.

Note that compromise regions $\Lambda^{0.4}, \Lambda^{0.45}$*, and* $\Lambda^{1.0}$ *can be defined analagously to* $\Lambda^{0.5}$*, where* $\Lambda^{0.4}$ *is the smallest in area and* $\Lambda^{1.0} = \Lambda$*. The computational results of these different compromise regions are given in Table 4.3.*

Observe that this approach is distinct from using constraints in the image space. Consider simple constraints of the form $f_i(x) \leq b_i$ for $i = 1, \ldots, p$, which is equivalent to intersecting $\mathcal{Y}_N$ with a hyperrectangle. However, this is often times a superset of the set of

Table 4.3: Results from simulated computational study for restricting primal algorithm to various compromise regions. Base case is represented by Λ^1.

Compromise Region	Λ^1	$\Lambda^{0.5}$	$\Lambda^{0.45}$	$\Lambda^{0.4}$
ND Images Returned	22	12	10	6
Optimization Subproblems	68	28	21	12
Skipped LNPs	0	7	8	8

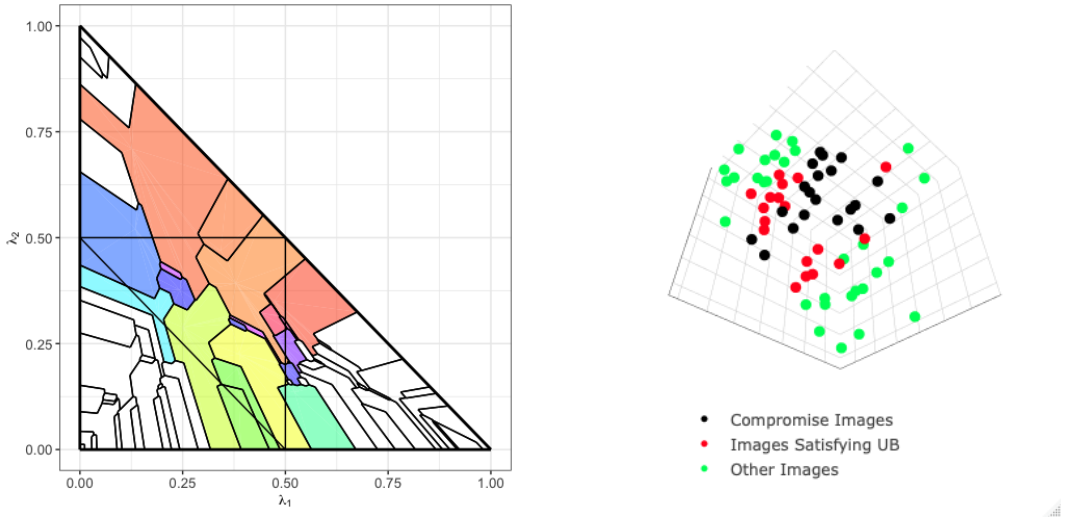


Figure 4.13: A set of compromise images is represented for a set of three preference weights. (Left) The compromise region is shown as the convex hull of the preference weights (black triangle). The colored weight set components intersect with this region. (Right) The image set is shown in image space (looking "upward" from the perspective of the utopia point): the compromise images (black); the images that satisfy the upper bounds only (red) but are not compromise images; and the remaining images (green).

compromise solutions, including many additional images. Computing tighter constraints is more computationally costly and ultimately changes the structure of the optimization subproblem. These are illustrated by the following example.

Example 23. *Consider the example in Figure 4.1, which includes 25 variables and 69 ND images, restricted to* $Y' = \{y \in \mathcal{Y}_N : \Lambda(y) \cap \Lambda^{0.5} \neq \emptyset\}$. *Figure 4.13(a) illustrates the weight set decomposition for* $\mathcal{Y}_N$ *and* $\Lambda^{0.5}$*; out of 69 ND images,* $|Y'| = 20$.

Simple constraints may be defined from the first images found, i.e., images found by minimizing $(\Pi^{TS}(\lambda))$ *with* $\lambda^1, \lambda^2, \ldots, \lambda^n$. *Consider constraints of the form*

$$f_i(x) \leq y_i^N(\{y^1, y^2, \ldots, y^n\})$$

for all $i = 1, 2, 3$. *For this instance, the constraints are* $f_1(x) \leq 7398$, $f_2(x) \leq 6364$, *and* $f_3(x) \leq 8314$. *However, 38 images in* $\mathcal{Y}_N$ *satisfy these constraints, which is nearly double the size of* Y'. *The images in* $\mathcal{Y}_N$ *are illustrated in Figure 4.13(b), colored to indicate*

whether they are the target compromise solutions (black), the non-compromise images that satisfy the upper bound constraints only (red), or neither (green).

Alternatively, the convex upper envelope of this set of images includes 5 linear constraints. Indeed, only the desired 20 ND images satisfy these constraints. However, it is unclear how to identify the images defining these constraints during run time. Furthermore, the added constraints could interfere with desirable structures in class-specific algorithms or negatively affect solve time for IP solvers.

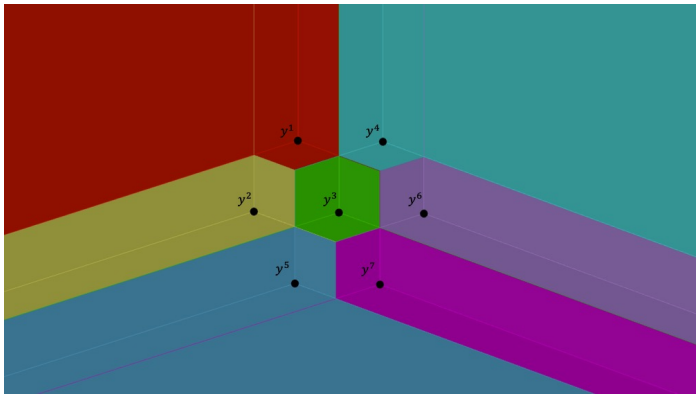
4.7 Conclusions

Until recently, the technique of weight set decompositions has only been thoroughly studied for weighted sum scalarization. This work applied geometric properties to compute the weighted Tchebychev weight set decomposition for three objectives.

The weighted sum weight set decomposition projects a well-studied 3-dimensional *surface* to the 2-dimensional weight set: the surface defined as the ND portion of the convex hull of the image set, represented by $\text{ND}(\text{conv}(\mathcal{Y}))$. Points from this surface are projected to weights in the weight space, which are then assigned to ESND images.

Maybe surprisingly, the weighted Tchebychev weight set decomposition also projects a 3-dimensional surface, one that has also received a fair amount of attention in the multiobjective literature: the surface defined as the *weakly ND portion of the upper envelope* of the image set, represented by $\text{weaklyND}(\mathcal{Y} + \mathbb{R}^p_{\geq})$. See this surface illustrated in Figure 4.14. Points from this surface are projected to weights in the weight space and then assigned to *all* ND images. In fact, the perspective of this surface from the utopia point, shown in Figure 4.14(b), illustrates the striking similarity to the weighted Tchebychev weight set decomposition, shown in Figure 4.3, with some minor transformations.

(a) View from the side.



(b) View from the perspective of the utopia point looking up at the weakly ND surface.

Figure 4.14: The boxed regions of $\mathcal{Y} + \mathbb{R}^3_{\geq}$ where $\mathcal{Y}$ is the set of images from Example 10. Images in $\mathcal{Y}$ are indicated by black points, and each nonnegative cone is uniquely colored. The coplanar surfaces correspond to: $y^1 + \mathbb{R}^3_{\geq}$ (red) and $y^2 + \mathbb{R}^3_{\geq}$ (yellow), $y^4 + \mathbb{R}^3_{\geq}$ (teal) and $y^6 + \mathbb{R}^3_{\geq}$ (violet), and $y^5 + \mathbb{R}^3_{\geq}$ (aqua) and $y^7 + \mathbb{R}^3_{\geq}$ (fuschia). An interactive web version of this image is available at www.geogebra.org/m/cngecnvq.

CHAPTER 5
CONCLUSION

We conclude with special computational considerations and motivations for future work.

Triobjective Mixed Integer Programs

It is well-known that increasing the number of objectives from $p = 2$ to $p = 3$ exceptionally increases the challenge of MOP algorithms [23, 43]. In particular, triobjective mixed integer programs (TOMIPs) illustrate the limitations of both the Boxed Line Method and the weighted Tchebychev weight set decomposition. Consider the following example.

Example 24 (TOMIP)**.** *A recent doctoral thesis [83] studies European electricity markets. A triobjective formulation of the "day-ahead electricity market clearing problem" aims to maximize market surplus and minimize market loss and missed surplus. This work applies the weighted sum weight space decomposition [23] to this TOMIP.*

The Boxed Line Method cannot be extended to include a third objective in a straightforward way. This is mainly due to the structural properties of the TOMIP ND frontier. In the ND frontier of a BOMIP, the basic building block is the ND line segment. However, in triobjective image space, one polytope can dominate some portion of another polytope, so the basic building block of the ND frontiers of a TOMIP can be described as polytopes with *polytopal holes*. To the best of our knowledge, there is only one published algorithm for TOMIPs to date [84].

A strength of the weight set decomposition for weighted sum is that the structural properties of the convex hull are consistent whether the decision variables are continuous, integer, or mixed [49, 50]. This, however, is *not* true for the weighted Tchebychev weight set decomposition. In fact, just as the structure of the ND frontier for TOMIPs become

increasingly more complicated in the image space, this complexity is also observed in the weight space. Indeed, this complexity deserves further rigorous study from the perspective of the weight space, as it may offer some simplification.

Approximation of the Nondominated Set

A theme throughout Chapters 2-4 is the value of approximating a ND set in partial run-time of an algorithm. By using *exact* multiobjective methods, which are guaranteed to return ND images, a decision maker gains valuable insights into the ND set long before the algorithm terminates. Unlike some other BOMIP algorithms, which construct the ND set from left-to-right or vice versa [26, 33], the Boxed Line Method (Chapter 2) constructs ND line segments sparsely among the ND frontier.

Approximation and "representation" of the ND frontier of MOPs have been well-studied. See [85] for a survey of more than 50 articles about approximation published since 1975, and see [86] for an early paper on representation. In most applications, decision makers must choose a single efficient solution to implement, regardless of how many are presented to them. Hence, in abbreviated run-time, multiobjective algorithms should be designed to produce a *minimal* set of valuable efficient solutions, where minimal means the set contains the final solution and as few other solutions as necessary to make this conclusion. If the decision maker's function for selection can be described explicitly, then this motivates optimization over the efficient set [87]. Otherwise, the weight space allows to systematically prioritize regions of image space to be explored. Algorithms may also be restricted to a desirable subset of the weight set (Chapter 4).

Parallelization

An additional benefit to the criterion space search algorithms discussed in this thesis is the ability for them to be *parallelized* efficiently. When the image space is decomposed into multiple boxes to be explored, these boxes can often be explored independently so that the

ND frontier and queue of boxes are updated in parallel. A parallel implementation of the Boxed Line Method was studied [88].

Increasing Efficiency of Black Box Solvers

The success of commercial and open-source IP solvers, including CPLEX, GUROBI, XPRESS, and SCIP, are the precursors to the success of criterion space search and weight space algorithms. In particular, the speed of these solvers enables the feasibility of implementing multiobjective techniques in real-world settings with large instances. As advertised by GUROBI:

> "An optimization business problem that can be solved today in one second would have taken 55 years in 1991."[1]

Here, we cite the steady progress of some of these solvers, which is seen as promising for the further integration of multiobjective techniques into the "mainstream" of operations research methods.

Commercial solvers have historically competed against one another and tend to outcompete open-source solvers, by far. Estimates from 2012 [89] evaluated the speed-up of solving mixed integer programs (MIPs). Depending on the source and the instances, the speed-up factor of CPLEX from version 6.5 to 11 ranged from 7.47 to 30. On one set of instances, the percentage of instances solved to optimality increased from 46.5% to 67.1%. From 2003 to 2010, the versions of XPRESS were observed to show a speed-up factor of about 7.5 [89]. In 2018, Gurobi reported 53 times improvement over the previous 8 years.[2]

While not the fastest option, open-source software are crucial for non-industrial applications, especially non-profit organizations. The three most recent updates to the open-source solver SCIP presented the speed-ups for solving MIPs. In 2017, SCIP version 5.0 was about 41% faster than version 4.0 [90]. In 2018, version 6.0 was about 8% faster than

[1] https://www.gurobi.com/resource/mathematical-optimization-what-you-need-to-know/
[2] https://www.gurobi.com/wp-content/uploads/2018/12/benchmarks.pdf

version 5.0.[3] In 2020, version 7.0 was about 14% faster than version 6.0 [91]. To illustrate this progress, a MIP that would have required 1 hour in 2017 would be solved in: 35 minutes in 2018, 33 minutes in 2019, and 28 minutes in 2020.

Equity as an Imperative Multiobjective Problem

The absence of equity measures has led to highly consequential, oppressive algorithms in the public sector. Books like Cathy O'Neil's *Weapons of Math Destruction* and Safiya Noble's *Algorithms of Oppression* document inequities such as courtroom sentences, bank loans, and Google's search results. In the last decade, operations research has been used to study equity in many domains, including influenza vaccination [92], kidney exchange [93], and other health applications [94]; emergency response [95] and other humanitarian logistics [96]; and public transportation [97] and other transportation networks [98]. A multiobjective paradigm allows decision makers to accurately quantify the trade-offs between equity and other objectives in a more sophisticated manner than an "equity constraint" permits. This understanding of trade-offs contributes to the decision maker's holistic understanding of the problem, which is sometimes as valuable as the final decision, itself.

Successful optimization for fairness relies on a library of exact and efficient MOP algorithms that *covers* key classes of problems. For instance, before 2015, the library of exact algorithms had zero support for adding an equity objective to a single-objective mixed integer program. This was solved by the publication of the Triangle Splitting Algorithm for BOMIP [31], which was then further improved upon by the work in this thesis. To date, the current library hardly supports the addition of equity as the third objective to a BOMIP (which results in a TOMIP and can be solved by a single existing algorithm [84]); furthermore, the library does not at all support the addition of equity as a fourth objective to a TOMIP. This thesis presented a valuable contribution to the class of BOMIP algorithms and a more inclusive weight space decomposition for MODOs with three objectives. However,

[3] http://www.optimization-online.org/DB_HTML/2018/07/6692.html

many more classes of problems require improved algorithmic support.

Many fairness metrics have been proposed [99], but the two most commonly used notions of group fairness are proportional fairness and max-min fairness [100]. These metrics may be modeled as multiple objectives, one for each member of the group. The weighted variant of the max-min fairness metric can be understood as the weighted Tchebychev scalarization that we studied. The proportional fairness metric can be understood as the Nash social welfare scalarization, which has also received great attention [101, 102, 103]. The weighted variant of this scalarization is prone to similar analysis via weight space decomposition, and therefore deserves rigorous study.

Behind us are the days in which industries can simply minimize cost or maximize efficiency; the pressure to contribute locally to their communities and globally to social good causes is more intense than ever. Airlines must satisfy crew preferences and reduce carbon emissions; grocery chains must consider locally sourced food suppliers and the increasing food deserts across America; and natural resource companies must strategize around sustaining their local ecosystems. Furthermore, the widespread need for equity and fairness in the algorithms used for decision making is only becoming more dire. The future of industry requires multiobjective optimization tools and experts to face these growing challenges. The progress of operations research has been made possible by many fields, including faster computer processors, highly effective pre-solving techniques, and theory-based improvements to algorithm design. Each of these improvements makes solving multiobjective optimization problems ever more practical. It is time that the field of operations research be prepared for the imminent paradigm shift towards multiobjective optimization.

Appendices

APPENDIX A
BOXED LINE METHOD: RELEVANT ALGORITHMS

Here we provide a formal proof of the claim (Section 2.2.3) that $L(z^1, z^2)$ *is nondominated if and only if (2.5) yields an optimal solution,* y^**, with* $\vec{w}^T z(y^*) = \vec{w}^T z^*$. We assume that $L(z^1, z^2)$ is nontrivial, i.e., $z^1 \neq z^2$. Note that the gradient vector $\vec{w}$ is given by (2.4), and its components are positive since z^1 and z^2 are from the same slice (i.e., they are NDPs with respect to the same slice problem).

Proof. ($\Rightarrow$) Assume line segment $L(z^1, z^2)$ is nondominated and $\vec{w}$ is its gradient vector. Let $\tilde{x}$ be feasible for (2.5) such that for $\tilde{z} := z(x)$, $\vec{w}^T\tilde{z} < \vec{w}^T z^*$. Then $\tilde{z}_1 \leq z_1^2$ and $\tilde{z}_2 \leq z_2^1$. Note that if $\tilde{z}$ dominated z^1 (or z^2) when the endpoint is closed, we would have an obvious contradiction with $L(z^1, z^2)$ being nondominated. Even when z^1 (or z^2) is open, then the conditionally strict inequality in (2.5) requires that $\tilde{z}_2 < z_2^1$. Therefore, for small enough $\epsilon > 0$, the point $z^\epsilon = (1-\epsilon)z^1 + \epsilon z^2$ belonging to $L(z^1, z^2)$ and close to z^1 satisfies $\tilde{z}_2 < z_2^\epsilon < z_2^1$ and $z_1^1 < z_1^\epsilon$. Since z^ϵ is nondominated by assumption, this implies $z_1^\epsilon < \tilde{z}_1$. In both cases, we have $z_1^1 < \tilde{z}_1$ and $z_2^2 < \tilde{z}_2$. Therefore, the orthogonal projection of $\tilde{z}$ onto $L(z^1, z^2)$ must be given by $\tilde{z} + k\vec{w}$ for some $k > 0$. Thus, $\tilde{z}$ dominates $\tilde{z} + k\vec{w} \in L(z^1, z^2)$, which again is a contradiction.

($\Leftarrow$) Let y^* be an optimal solution as given, and suppose that $L(z^1, z^2)$ contains a *dominated* point, $\breve{z}$. Since $\breve{z} \in L(z^1, z^2)$, then $\vec{w}^T\breve{z} = \vec{w}^T z^* = \vec{w}^T z(y^*)$. Then $\breve{z}$ is dominated by some NDP, say $z^N = z(x^N)$ where x^N is feasible for (2.5). Since $\vec{w}$ is positive and $\breve{z} - z^N$ is nonzero and nonnegative, then $\vec{w}^T(\breve{z} - z^N) > 0 \Leftrightarrow \vec{w}^T\breve{z} > \vec{w}^T z^N \Leftrightarrow \vec{w}^T z(y^*) > \vec{w}^T z(y^N)$. Then y^* is not optimal, which is a contradiction. □

Next, we provide pseudo-code for the basic variant of BLM, for the recursive variant of BLM, and for the generation of the new instances.

Algorithm 4 BasicOuterLoop

Input: Objective functions $z_1(x), z_2(x)$ and feasible set X are *fixed and global*

Output: N the entire nondominated frontier as a set of line segments, Q any unexplored regions of the frontier space (for partial run-time)

$z^L \leftarrow$ lexmin$\{(z_1, z_2), IP\}$ is the UL nondominated point
$z^R \leftarrow$ lexmin$\{(z_2, z_1), IP, ifs = z^L.solution\}$ is the BR nondominated point
$N \leftarrow point(z^L) \cup point(z^R)$
if $\min_{i=1,2}\{|z_i^L - z_i^R|\} < \epsilon$ **then**
  $Q \leftarrow \emptyset$ no region to explore
  return (N, Q)
else
  $Q \leftarrow B(z^L, z^R)$ the unexplored region defined by box with corner points z^L, z^R
end if
while $Q \neq \emptyset$ **do**
  $B(z^L, z^R) \leftarrow element(Q)$
  $Q \leftarrow setminus(Q, B(z^L, z^R))$
  $\mu \in (z_2^R, z_2^L)$ arbitrary horizontal dividing line between z^L, z^R
  $z^* \leftarrow$ lexmin$\{(z_1, z_2) : z_2(x) \leq \mu, IP, ifs = z^R.solution\}$ find NDP in lower half
  if $\mu - z_2^* > \epsilon$ **then**
    if $\min_{i=1,2}\{|z_i^* - z_i^R|\} < \epsilon$ **then**
      $N \leftarrow N \cup point(z^*)$ if epsilon-close, add to ND frontier, do not add region to queue
    else
      $Q \leftarrow Q \cup B(z^*, z^R)$ otherwise, add a new unexplored region to the queue
    end if
    $\hat{z} \leftarrow$ lexmin$\{(z_2, z_1) : z_1(x) \leq z_1^* - \epsilon, IP, ifs = z^L.solution\}$
    if $\min_{i=1,2}\{|\hat{z}_i - z_i^L|\} < \epsilon$ **then**
      $N \leftarrow N \cup point(\hat{z})$
    else
      $Q \leftarrow Q \cup B(z^L, \hat{z})$
    end if
  else $\{|z_2^* - \mu| < \epsilon\}$
    $(z^1, z^2, z1\_open, z2\_open, M) \leftarrow$ BasicInnerLoop(z^*, z^L, z^R)
    if $\min_{i=1,2}\{|z_i^1 - z_i^2|\} < \epsilon$ **then**
      $N \leftarrow N \cup M \cup point(z^1) \cup point(z^2)$ M *only* includes NDPs that dominate z^1 or z^2
    else
      $N \leftarrow N \cup M \cup line(z^1, z^2)$
    end if

(continued on next page)

(BasicOuterLoop continued)
if $z1\_open$ **then**
$\hat{z}^1 \leftarrow$ argmin$\{m_2 : m \in M \cup z^L, m_2 \geq z_2^1\}$ NDP found from Inner Loop that dominates z^1
else $\{z^1$ is closed$\}$
$\hat{z}^1 \leftarrow z^1$ closed endpoint as LR boundary of unexplored region
end if
if $\min_{i=1,2}\{|z_i^L - \hat{z}_i^1|\} < \epsilon$ **then**
$N \leftarrow N \cup point(z^L) \cup point(\hat{z}^1)$ add to frontier
else
$Q \leftarrow Q \cup B\left(z^L, \hat{z}^1\right)$ add to queue
end if
if $z2\_open$ **then**
$\hat{z}^2 \leftarrow$ argmin$\{m_1 : m \in M \cup z^R, m_1 \geq z_1^2\}$ NDP found from Inner Loop that dominates z^2
else $\{z^2$ is closed$\}$
$\hat{z}^2 \leftarrow z^2$ closed endpoint as UL boundary of unexplored region
end if
if $\min_{i=1,2}\{|\hat{z}_i^2 - z_i^R|\} < \epsilon$ **then**
$N \leftarrow N \cup point(\hat{z}^2) \cup point(z^R)$ add to frontier
else
$Q \leftarrow Q \cup B\left(\hat{z}^2, z^R\right)$ add to queue
end if
end if
end while
return (N, Q)

Algorithm 5 BasicInnerLoop

Input: nondominated point $z^* = \text{lexmin}\{(z_1, z_2) : z_2(x) \leq \mu\}$ such that z^* on the split, i.e. $z_2^* = \mu$

Input: z^L, z^R upper-left and lower-right boundaries of the rectangle to explore

Output: $(z^1, z^2, z1\_open, z2\_open, M)$ the UL and BR nondominated points endpoints defining the line segment of the nondominated frontier containing z^*; markers to tell whether each endpoint is open (TRUE) or closed (FALSE); and the nondominated points that dominate open endpoints in M

$(z^1, z^2, \vec{w}, w\_known) \leftarrow LineGen(z^*, z^L, z^R)$ generate line segment of frontier that contains z^* for integer vector x_I^* via Line Generation subroutine (line segment defined by two endpoints) and return slope of the line segment to be used for scalarization
if $|z_2^1 - z_2^L| < \epsilon$ and $|z_1^1 - z_1^L| > \epsilon$ **then**
    $z1\_open \leftarrow TRUE$ because z^L dominates z^1
else
    $z1\_open \leftarrow FALSE$ temporary status with best of known information
end if
if $|z_1^2 - z_1^R| < \epsilon$ and $|z_2^2 - z_2^R| > \epsilon$ **then**
    $z2\_open \leftarrow TRUE$
else
    $z2\_open \leftarrow FALSE$
end if
$M \leftarrow \emptyset$
if $\min_{i=1,2} \left\{|z_i^1 - z_i^2|\right\} < \epsilon$ **or** $\neg w\_known$ **then**
    return $(z^*, z^*, z1\_open, z2\_open, M)$ ND segment is just the isolated nondominated point z^*
end if
$y^* \leftarrow solution \left(\min\{\vec{w}^T z(x) : z_1(x) \leq z_1^2 - \epsilon(z2\_open),\ \ z_2(x) \leq z_2^1 - \epsilon(z1\_open),\ \ IP, ifs = z^*.solution$ (epsilon adjustments only for open endpoints)
while $\vec{w}^T z(y^*) < \vec{w}^T z^* - \epsilon$ (z^* is suboptimal to $z(y^*)$ w.r.t. $\vec{w}$) **do**
    if $\min_{i=1,2} \{|z_i^* - z_i(y^*)|\} < \epsilon$ and $max_{i=1,2} \{|z_i^* - z_i(y^*)|\} > \epsilon$ **then**
        ERROR message
    end if
(continued on next page)

(BasicInnerLoop continued)
if $z_1(y^*) \leq z_1^* - \epsilon$ **then**
  $\hat{v}.solution \leftarrow \min\{z_2(x) : \vec{w}^T z(x) \leq \vec{w}^T z^*,\ \ x_I = y_I^*,\ \ LP,\ ifs = y_I^*\}$ explore the slice of y_I^*
  $\hat{v} \leftarrow z(\hat{v}.solution)$
  if $\hat{v} \in line(z^1, z^2)$ **then**
    $\hat{v}.solution \leftarrow \min\{z_1(x) : z_2(x) \leq \hat{v}_2 + \epsilon,\ \ x_I = y_I^*,\ \ LP,\ ifs = y_I^*\}$ correct $\hat{v}$
    $\hat{v} \leftarrow z(\hat{v}.solution)$
  end if
  if $\hat{v} \in line(z^1, z^2)$ **then**
    $z^1 \leftarrow \hat{v}$
    $z1\_open \leftarrow FALSE$
  else $\{\hat{v} \notin line(z^1, z^2)\}$
    $v^1 \leftarrow \hat{v}$ keep track of most recent $\hat{v}$ that dominates z^1
    $z^1 \leftarrow point(line(z^1, z^2), z_2 = \hat{v}_2)$
    $z1\_open \leftarrow TRUE$
  end if
end if
if $z_1(y^*) \geq z_1^* + \epsilon$ **then**
  $\hat{v}.solution \leftarrow \min\{z_1(x) : \vec{w}^T z(x) \leq \vec{w}^T z^*,\ \ x_I = y_I^*,\ \ LP,\ ifs = y_I^*\}$ explore the slice of y_I^*
  $\hat{v} \leftarrow z(\hat{v}.solution)$
  if $\hat{v} \in line(z^1, z^2)$ **then**
    $z^2 \leftarrow \hat{v}$
    $z2\_open \leftarrow FALSE$
  else $\{\hat{v} \notin line(z^1, z^2)\}$
    $v^2 \leftarrow \hat{v}$ keep track of most recent $\hat{v}$ that dominates z^2
    $z^2 \leftarrow point(line(z^1, z^2), z_1 = \hat{v}_1)$
    $z2\_open \leftarrow TRUE$
  end if
end if
$y^* \leftarrow solution\left(\min\{\vec{w}^T z(x) : z_1(x) \leq z_1^2 - \epsilon(z2\_open),\ \ z_2(x) \leq z_2^1 - \epsilon(z1\_open),\ \ IP,\ ifs = z^*\}\right)$
end while
if $z1\_open$ and $|z_2^1 - z_2^L| > \epsilon$ **then**
  $\hat{z}^1 \leftarrow point(\min\{z_1(x) : z_2(x) \leq z_2^1,\ IP,\ ifs = v^1.solution\}, z_2^1)$ NDP that dominates z^1
  $M \leftarrow M \cup \hat{z}^1$
end if
if $z2\_open$ and $|z_1^2 - z_1^R| > \epsilon$ **then**
  $\hat{z}^2 \leftarrow point(z_1^2, \min\{z_2(x) : z_1(x) \leq z_1^2,\ IP,\ ifs = v^2.solution\})$ NDP that dominates z^2
  $M \leftarrow M \cup \hat{z}^2$
end if
return $(z^1, z^2, z1\_closed, z2\_closed, M)$

Algorithm 6 RecursiveOuterLoop

Input: Objective functions $z_1(x), z_2(x)$ and feasible set X are *fixed and global*

Output: N the entire nondominated frontier as a set of line segments, Q any unexplored regions of the frontier space (for partial run-time)

$z^L \leftarrow$ lexmin$\{(z_1, z_2), IP\}$ is the UL nondominated point
$z^R \leftarrow$ lexmin$\{(z_2, z_1), IP, ifs = z^L.solution\}$ is the BR nondominated point
$N \leftarrow point(z^L) \cup point(z^R)$
if $\min_{i=1,2}\{|z_i^L - z_i^R|\} < \epsilon$ **then**
  $Q \leftarrow \emptyset$ no region to explore
  return (N, Q)
else
  $Q \leftarrow B(z^L, z^R)$ the unexplored region defined by box with corner points z^L, z^R
end if
while $Q \neq \emptyset$ **do**
  $B(z^L, z^R) \leftarrow element(Q)$
  $Q \leftarrow setminus(Q, B(z^L, z^R))$
  $\mu \in (z_2^R, z_2^L)$ arbitrary horizontal dividing line between z^L, z^R
  $z^* \leftarrow$ lexmin$\{(z_1, z_2) : z_2(x) \leq \mu, IP, ifs = z^R.solution\}$ find NDP in lower half
  if $\mu - z_2^* > \epsilon$ **then**
    if $\min_{i=1,2}\{|z_i^* - z_i^R|\} < \epsilon$ **then**
      $N \leftarrow N \cup point(z^*)$ if epsilon-close, add to ND frontier, do not add region to queue
    else
      $Q \leftarrow Q \cup B(z^*, z^R)$ otherwise, add a new unexplored region to the queue
    end if
    $\hat{z} \leftarrow$ lexmin$\{(z_2, z_1) : z_1(x) \leq z_1^* - \epsilon, IP, ifs = z^L.solution\}$
    if $\min_{i=1,2}\{|\hat{z}_i - z_i^L|\} < \epsilon$ **then**
      $N \leftarrow N \cup point(\hat{z})$
    else
      $Q \leftarrow Q \cup B(z^L, \hat{z})$
    end if
  else $\{|z_2^* - \mu| < \epsilon\}$
    $L \leftarrow \emptyset$ no known line segments so far
    $M \leftarrow$ RecursiveInnerLoop(z^*, z^L, z^R, L) M includes all found ND points and line segments
    (continued on next page)

(RecursiveOuterLoop continued)
$M_< \leftarrow Orderbyz1(\{z^L, z^R\} \cup M)$ giving ordered set $(z^L, m^1, ..., m^k, z^R)$
for $\forall$ consecutive pairs $(a, b) \subset M_<$ **do**
if $line(a, b) \in M$ **then**
$N \leftarrow N \cup line(a, b)$ add line segment to frontier
else
$N \leftarrow N \cup point(a) \cup point(b)$ add individual points to frontier
if $\min_{i=1,2}\{|a_i - b_i|\} > \epsilon$ **then**
$Q \leftarrow Q \cup B\,(a, b)$ add unexplored region to queue
end if
end if
end for
end if
end while
return (N, Q)

Algorithm 7 Inner Loop: Recursive

Input: nondominated point z^*

Input: z^L, z^R known NDPs providing the upper-left and lower-right corner points of the box

Input: L the set of all inherited line segments so far (for line trimming); note that $L = \emptyset$ on first call

Output: M all found nondominated points/line segments

$(z^1, z^2, \vec{w}, w\_known) \leftarrow LineGen(z^*, z^L, z^R)$
if w_known **then**
  $(z^1, z^2) \leftarrow LineTrim(z^*, z^1, z^2, \vec{w}, L)$
end if
if $|z_2^1 - z_2^L| < \epsilon$ and $|z_1^1 - z_1^L| > \epsilon$ **then**
  $z1\_open \leftarrow TRUE$ because z^L dominates z^1
else
  $z1\_open \leftarrow FALSE$ temporary status with best of known information
end if
if $|z_1^2 - z_1^R| < \epsilon$ and $|z_2^2 - z_2^R| > \epsilon$ **then**
  $z2\_open \leftarrow TRUE$
else
  $z2\_open \leftarrow FALSE$
end if
$M \leftarrow \emptyset$
if $\min_{i=1,2} \left\{|z_i^1 - z_i^2|\right\} < \epsilon$ or $\neg w\_known$ **then**
  $M \leftarrow M \cup point(z^*)$ frontier is just the isolated nondominated point z^*
  return M
else
  $L \leftarrow L \cup line(z^1, z^2)$
end if
$y^* \leftarrow solution\left(\min\{\vec{w}^T z(x) : z_1(x) \leq z_1^2 - \epsilon(z2\_open),\ \ z_2(x) \leq z_2^1 - \epsilon(z1\_open),\ \ IP,\ ifs = z^*.solutio\right.$
(epsilon adjustment only for open endpoints)
while $\vec{w}^T z(y^*) < \vec{w}^T z^* - \epsilon$ (z^* is suboptimal to $z(y^*)$ w.r.t. $\vec{w}$) **do**
  if $\min_{i=1,2} \{|z_i^* - z_i(y^*)|\} < \epsilon$ and $max_{i=1,2} \{|z_i^* - z_i(y^*)|\} > \epsilon$ **then**
    ERROR message
  end if
  $M \leftarrow M \cup z(y^*)$
    (continued on next page)

(RecursiveInnerLoop continued)
if $z_1(y^*) \leq z_1^* - \epsilon$ **then**
  $\alpha \leftarrow \text{argmin}\{p_2 : p \in z^L \cup M, p_1 < z_1^*\}$ bound by the nearest (on left) found NDP to z^*
  $M' \leftarrow InnerLoop(z(y^*), \alpha, z^*, L)$ recurse within $B(\alpha, z^*)$ with inherited line segments in L
  $M \leftarrow M \cup M'$
  $L \leftarrow L \setminus line(z^1, z^2)$ remove "old" line segment
  $\hat{p} \leftarrow \text{argmin}\{p_2 : p \in M, p_1 < z_1^*\}$ choose the nearest (on left) found NDP to z^*
  if $\hat{p} \in line(z^1, z^2)$ **then**
    $z^1 \leftarrow \hat{p}$
    $z1\_open \leftarrow FALSE$
  else $\{\hat{p} \notin line(z^1, z^2)\}$
    $z^1 \leftarrow point(line(z^1, z^2), z_2 = \hat{p}_2)$
    $z1\_open \leftarrow TRUE$
  end if
end if
if $z_1(y^*) \geq z_1^* + \epsilon$ **then**
  $\beta \leftarrow \text{argmin}\{p_1 : p \in z^R \cup M, p_2 < z_2^*\}$
  $M' \leftarrow InnerLoop(z(y^*), z^*, \beta, L)$ recurse within $B(z^*, \beta)$
  $M \leftarrow M \cup M'$
  $\hat{p} \leftarrow \text{argmin}\{p_1 : p \in M, p_2 < z_2^*\}$ choose the nearest (on right) found NDP to z^*
  if $\hat{p} \in line(z^1, z^2)$ **then**
    $z^2 \leftarrow \hat{p}$
    $z2\_open \leftarrow FALSE$
  else $\{\hat{p} \notin line(z^1, z^2)\}$
    $z^2 \leftarrow point(line(z^1, z^2), z_1 = \hat{p}_1)$
    $z2\_open \leftarrow TRUE$
  end if
end if
$L \leftarrow L \cup line(z^1, z^2)$ add updated line segment
$y^* \leftarrow solution\left(\min\{\vec{w}^T z : z_1(x) \leq z_1^2 - \epsilon(z2\_open),\ \ z_2(x) \leq z_2^1 - \epsilon(z1\_open),\ \ IP,\ ifs = z^*.solution\right.$
(epsilon adjustment only for open endpoints)
end while
$M \leftarrow M \cup line(z^1, z^2)$
return M

Algorithm 8 Line Segment Generation

Input: z^* nondominated point whose integer vector is fixed
Input: z^L, z^R upper-left and lower-right boundaries of the rectangle to explore
Output: $(z^1, z^2, \vec{w}, w\_known)$ the UL and BR endpoints for the line segment of frontier including z^* and the slope of the line segment (if found)

$z1\_known, z2\_known \leftarrow FALSE$
$w\_known \leftarrow FALSE$ weight vector for which z^* is optimal (if found, can be used to directly compute z^1, z^2)
$\delta_1, \delta_2 \leftarrow 100\epsilon$ starting horizontal/vertical distance
(Explore for z^2 in lower-right:)
$i \leftarrow 0$
$t^i \leftarrow$ lexmin$\{(z_2, z_1) : z_1(x) \leq z_1^* + \delta_1,\ LP,\ ifs = z^*.solution\}$ is the BR nondominated point
while $w\_known = FALSE$ **and** $z2\_known = FALSE$ **do**
  if $\min\left(|z_1^* - t_1^i|, |z_2^* - t_2^i|\right) < \epsilon$ **then**
    $z^2 \leftarrow z^*$
    $z2\_known \leftarrow TRUE$
    BREAK
  end if
  $\vec{w}^i \leftarrow slope(z^*, t^i)$
  $t^{i+1} \leftarrow \min\{(\vec{w}^i)^T z(x),\ LP,\ ifs = z^*.solution\}$ (scalarized by $\vec{w}^i$)
  if $(\vec{w}^i)^T t^{i+1} = (\vec{w}^i)^T z^*$ and $|z_1^* - t_1^i| > \epsilon$ and $|z_2^* - t_2^i| > \epsilon$ **then**
    $\vec{w} \leftarrow \vec{w}^i$
    $w\_known \leftarrow TRUE$
    if $i \geq 1$ and $t^i \in B(z^L, z^R)$ **then**
      $z^2 \leftarrow t^i$
      $z2\_known \leftarrow TRUE$
    end if
    BREAK
  end if
  $i \leftarrow i + 1$
end while

(continued on next page)

(Line Generation)
if $w\_known = FALSE$ **then**
  (Explore for z^1 in upper-left:)
  $i \leftarrow 0$
  $t^i \leftarrow$ lexmin$\{(z_1, z_2) : z_2(x) \leq z_2^* + \delta_2,\ LP,\ ifs = z^*.solution\}$ is the UL nondominated point
end if
while $w\_known = FALSE$ **and** $z1\_known = FALSE$ **do**
  if $\min\left(|z_1^* - t_1^i|, |z_2^* - t_2^i|\right) < \epsilon$ **then**
    $z^1 \leftarrow z^*$
    $z1\_known \leftarrow TRUE$
    BREAK
  end if
  $\vec{w}^i \leftarrow slope(z^*, t^i)$
  $t^{i+1} \leftarrow \min\{(\vec{w}^i)^T z(x),\ LP,\ ifs = z^*.solution\}$
  if $(\vec{w}^i)^T t^{i+1} = (\vec{w}^i)^T z^*$ **then**
    $\vec{w} \leftarrow \vec{w}^i$
    $w\_known \leftarrow TRUE$
    if $i \geq 1$ **then**
      $z^1 \leftarrow t^i$
      $z1\_known \leftarrow TRUE$
    end if
    BREAK
  end if
  $i \leftarrow i + 1$
end while
(By the time the code has reached this point, either $\vec{w}$ is known OR *both* z^1 *and* z^2 are known)
if $z1\_known = FALSE$ **then**
  $z^1 \leftarrow \min\{z_1(x) : \vec{w}^T z(x) \leq \vec{w} z^* + \epsilon, z_2(x) \leq z_2^L,\ LP,\ ifs = z^*.solution\}$
end if
if $z2\_known = FALSE$ **then**
  $z^2 \leftarrow \min\{z_2(x) : \vec{w}^T z(x) \leq \vec{w} z^* + \epsilon, z_1(x) \leq z_1^R,\ LP,\ ifs = z^*.solution\}$
end if
if $w\_known = FALSE$ **then**
  $\vec{w} \leftarrow [-1, -1]$ clearly impossible gradient vector
end if
$\vec{w} \leftarrow \vec{w}/||\vec{w}||_1$
return $(z^1, z^2, \vec{w}, w\_known)$

A.1 Line Segment Trimming Subroutine

Let $L_\alpha := L(\alpha^1, \alpha^2)$, where $\alpha_1^1 < \alpha_1^2$ and $\alpha_2^1 > \alpha_2^2$, denote the current (or "child") line segment, which contains NDP z^α, and has gradient vector $\vec{w}^\alpha$. Suppose that a line segment from a parent call, $L_\beta := L(\beta^1, \beta^2)$, where $\beta_1^1 < \beta_1^2$ and $\beta_2^1 > \beta_2^2$, contains the NDP z^β and has gradient $\vec{w}^\beta$. The two line segments L_α and L_β can be extended to lines that intersect provided $\vec{w}^\alpha \neq \vec{w}^\beta$ (recall the gradient vectors are normalized). Their point of intersection, $\gamma \in \mathbb{R}^2$, is easily computed: it is the solution of the system of two linear equations given by $\vec{w}^\alpha \gamma = \vec{w}^\alpha z^\alpha$ and $\vec{w}^\beta \gamma = \vec{w}^\beta z^\beta$.

The resulting intersection point is contained in both line segments if and only if

$$\gamma_1 \in [\alpha_1^1, \alpha_1^2] \cap [\beta_1^1, \beta_1^2] \quad \text{and} \quad \gamma_2 \in [\alpha_2^2, \alpha_2^1] \cap [\beta_2^2, \beta_2^1]. \tag{A.1}$$

If γ satisfies (A.1), then the current line segment, L_α, is trimmed. Exactly one of the two subsets of L_α, $L(\alpha^1, \gamma)$ or $L(\gamma, \alpha^2)$, must be dominated by points from L_β. Since $z^\alpha \in L_\alpha$ is an NDP, L_α is trimmed so as to retain the portion of it that contains z^α: if $\gamma_1 < z_1^\alpha$, then update $\alpha^1 = \gamma$ and flag α^1 to be closed; otherwise, if $\gamma_1 > z_1^\alpha$, then update $\alpha^2 = \gamma$ and flag α^2 to be closed. In the unlikely case that $\gamma_1 = z_1^\alpha$, L_α is trimmed as follows: if $\gamma_1 < z_1^\beta$, then update $\alpha^2 = \gamma$ and flag α^2 to be closed; otherwise, if $\gamma_1 > z_1^\beta$, then update $\alpha^1 = \gamma$ and flag α^1 to be closed. Note that updating the appropriate endpoint of L_α to be the intersection point γ and flagging it as closed prevents rediscovery of any point in L_β that dominates any part of L_α.

A.2 Instance Generation

The generated instances have the objective functions $z_1(x) := x_1$ and $z_2(x) := x_2$, where $x \in \mathbb{R}^2$ is a vector of two continuous decision variables. Each instance's NDF includes some sections of the line segment $L_k = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 + x_2 = 0, -k \leq x_i \leq k \text{ for } i = 1, 2\}$ where $k \in (0, \infty)$ is a parameter. This line segment will be one slice in the instance. The instance has π other slices, where π is a parameter, all of which have their image in criterion space given by a pointed cone. The vertices of each cone lie on a line segment parallel to L_k but shifted vertically down by $d \in (0, \frac{2k}{\pi})$ units: they lie on

$$L_d = \{(x_1, x_2) : x_1 + x_2 = -d, -k \leq x_i \leq k - d \text{ for } i = 1, 2\},$$

where d is a parameter. The slice with vertex $(a, b) \in L_d$ is generated by the slice problem with feasible set

$$P_{(a,b)}(\theta_1, \theta_2) = \{x \in \mathbb{R}^2 \ : \ \theta_1 x_1 + (1 - \theta_1) x_2 \geq \theta_1 a + (1 - \theta_1) b \tag{A.2}$$

$$\theta_2 x_1 + (1 - \theta_2) x_2 \geq \theta_2 a + (1 - \theta_2) b\}, \tag{A.3}$$

where $\theta_1 \in [\frac{1}{2}, 1]$ and $\theta_2 \in [0, \frac{1}{2}]$ are parameters that control the width of the cone. These ranges for θ_1 and θ_2 ensure that the resulting cone contains $(a, b) + \mathbb{R}^2_+$ and is contained in the half-space that contains $L_d + \mathbb{R}^2_+$. Figure A.1 illustrates this structure, showing the image of the feasible set in criterion space for 4 slices: L_k and three pointed cones with vertices lying on L_d. Note that $\theta_1 = 1$ and $\theta_2 = 0$ imply that the cone is precisely

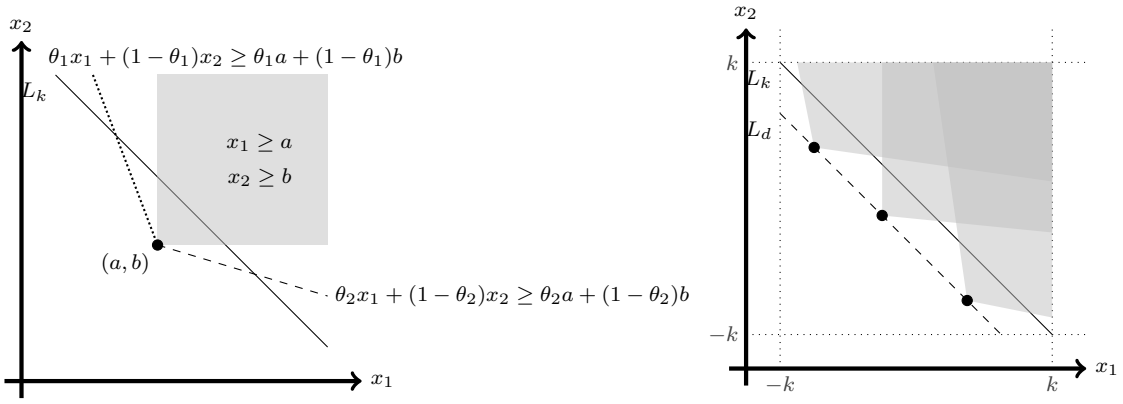


Figure A.1: (Left) The orthogonal pointed cone with vertex (a, b) is shaded. The generalized pointed cone is defined between the dotted and dashed lines. Different slices in an instance may have different values of θ_1 and θ_2. (Right) An example with 4 slices, including L_k and the boundary of three cones.

$(a, b) + \mathbb{R}^2_+$, and hence that (a, b) is an isolated NDP. Otherwise, if $\theta_1 < 1$, the left-hand boundary of the cone, from (a, b) to its intersection with L, is an NLS, while if $\theta_2 > 0$, the right-hand boundary of the cone, from (a, b) to its intersection with L_k, is an NLS. All instances are constructed to have the property that no two cones overlap within the band between L_k and L_d. This means that L_k alternates between a section that is part of the NDF and a section dominated by one cone.

Given any set of π polyhedra, say $P^i = \{x \in \mathbb{R}^m \ : \ A^i x \geq c^i\}$, for $i = 1, \ldots, \pi$, the MIP feasible set $\{(x, y) \in \mathbb{R}^m \times \{0, 1\}^\pi \ : \ A^i x \geq c^i - M_i(1-y_i),\ \forall i = 1, \ldots, \pi,\ \sum_{i=1}^{\pi} y_i = 1\}$ has (for appropriately chosen big-M values, $(M_i)_{i=1}^{\pi}$) slice problems with feasible sets $\{P^i\}_{i=1}^{\pi}$. We take the objective function vector to be $z(x) = (x_1, ...x_m)$ so that any polyehdron in criterion space is easily reverse-engineered to give a polyhedron in decision space (they have precisely the same description).

For the $\pi + 1$ polyhedra consisting of L_k and π pointed cones $P_{(a_i,b_i)}(\theta_1^i, \theta_2^i)$, for $i = 1, \ldots, \pi$, where (a_i, b_i) denotes the vertex of the ith cone, this gives the following BOMIP:

$$\text{minimize} \qquad (x_1, x_2) \tag{A.4}$$

$$\text{s.t.} \qquad x_1 + x_2 \geq -2k(1 - y_0) \tag{A.5}$$

$$\theta_1^i x_1 + (1-\theta_1^i)x_2 \geq \theta_1^i a_i + (1-\theta_1^i)b_i - 2k(1-y_i) \quad \forall i = 1, 2, ..., \pi \tag{A.6}$$

$$\theta_2^i x_1 + (1-\theta_2^i)x_2 \geq \theta_2^i a_i + (1-\theta_2^i)b_i - 2k(1-y_i) \quad \forall i = 1, 2, ..., \pi \tag{A.7}$$

$$\sum_{i=0}^{\pi} y_i = 1 \tag{A.8}$$

$$-k \leq x_i \leq k \qquad \forall i = 1, 2 \tag{A.9}$$

$$y \in \{0, 1\}^{\pi+1}. \tag{A.10}$$

This BOMIP has a number of variables and a number of constraints that is linear in π: it

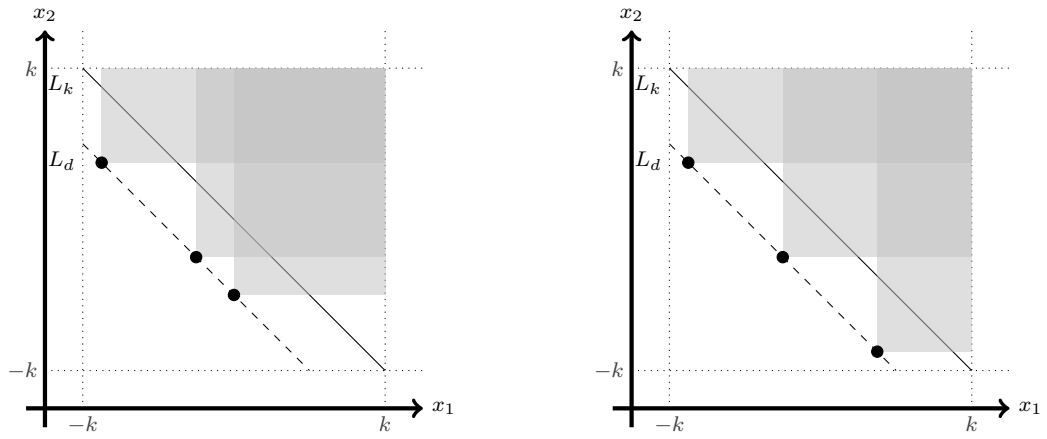


Figure A.2: NDPs and associated cones generated by the fixed cone-width class of instances. (Left) If the pointed cones from NDPs overlap on L_k, then there are fewer than $\pi + 1$ nondominated segments of L_k. (Right) Choosing the NDPs such that $-b_i < a_{i+1}$ prevents such overlapping, thus guaranteeing exactly $\pi + 1$ line segments of L_k are nondominated. All instances we generate are of the latter form.

has $\pi + 3$ variables and $4 + 2\pi$ constraints.

We generate two different classes of instances having the structure described above. One class has the π cone points distributed randomly along L_d, but has $\theta_1 = 1$ and $\theta_2 = 0$ for all cones. Since this means that all cones are orthogonal pointed cones, we refer to these as *fixed cone-width instances*. An illustration of these instances is given in Figure A.2. The other class has the cone points equally spaced along L_d, but has randomized values of θ_1 and θ_2; we call these *randomized cone-width instances*. These are illustrated by the right-hand side of Figure A.1.

All instances are carefully designed to ensure that each NLS, which is not a point, has length at least ϵ. In the case of the fixed cone-width instances, the only such NLSs are sections of L_k, and these can be guaranteed to have length at least ϵ by a careful choice of the randomized spacing between cone points, as a function of d. For fixed cone-width instances we take $d = k/4$, and we expect the choice of k to ensure that $d \geq \epsilon$. The latter condition also ensures that the horizontal and vertical gaps in the NDF are of length at least ϵ. In the case of random cone-width instances, an NLS may be either a section of L_k or a section of a cone boundary. The length of an NLS from L_k is ensured to be at least ϵ by restricting the values of θ_1 and θ_2 to lie within $[\frac{3}{4}, 1]$ and $[0, \frac{1}{2}]$, respectively, and choosing $d = k/(\pi + 1) - \frac{1}{2}$. The length of an NLS generated by a cone boundary is ensured to be at least ϵ by requiring $d \geq \epsilon$. Again, we expect the choice of k to guarantee that $d \geq \epsilon$, ensuring that the horizontal and vertical gaps in the NDF are also of length at least ϵ. We force a proportion of the cones in a randomized cone-width instance to be orthogonal[1].

Specifics of our procedure for generating fixed cone-width instances are given in Algorithm 9. By construction, the NDF associated with such an instance has exactly π isolated NDPs and $\pi + 1$ line segments (2 half-open at the ends of L_k and $\pi - 1$ open segments

[1]A cone is chosen to be orthogonal with probability ϕ, a parameter. In all our instances, we used $\phi = 0.05$.

in between isolated NDPs). Even large instances of this class of BOMIPs are solved quite quickly by most methods, so they are not ideal for testing computational efficiency. However, since their NDFs are known exactly, these instances are useful for validating the correctness of an algorithm.

Specifics of our procedure for generating random cone-width instances are given in Algorithm 10 and Algorithm 11, which generate the cone width parameters $\{(\theta_1^i, \theta_2^i)\}_{i=1}^{\pi}$ and cone vertices, $\{(a^i, b^i)\}_{i=1}^{\pi}$, respectively. The NDFs associated with the resulting BOMIP will have no more than $3\pi + 1$ line segments ($\pi + 1$ from L_k and at most 2 per cone), including open endpoints induced by any orthogonal cones and closed endpoints induced by intersecting slices. The random cone-width class of instances is more difficult to solve, in practice, than the fixed cone-width, because of the high frequency of intersecting slices in the NDF.

Together, these two structured sets of instances provide a useful way to study the accuracy and robustness of a BOMIP algorithm.

Algorithm 9 Fixed Cone-Width NDP Generation

$d = k/4$ (distance by which L_d is shifted down)
$w = (2k - d)/n$ (the width of subintervals)
$a_1 = U(-k, -k + w)$
$b_1 = -a_1 - d$
for $i = 2, 3, ..., n$ **do**
    $a_i = U(\max\{-k + (i-1)w, -b_{i-1} + \epsilon\}, -k + iw)$
    $b_i = -a_i - d$
end for

Algorithm 10 Randomized Theta Generation

thetalist= $\emptyset$
for $i = 1, 2, ..., n$ **do**
    if $U(0, 1) \leq \pi$ **then**
        $\theta_1 = 1$
        $\theta_2 = 0$
    else
        $\theta_1 = U(\frac{3}{4}, 1)$
        $\theta_2 = U(0, \frac{1}{4})$
    end if
    thetalist.append((θ_1, θ_2))
end for

Algorithm 11 Randomized Cone-Width NDP Generation

$d = k/(n+1) - 0.5$
$a_1 = -k + 0.5d$
$b_1 = -a_1 - d$
for $i = 2, ..., n$ **do**
    $a_i = a_{i-1} + 2d + 1$
    $b_i = -a_i - d$
end for

A.3 Comparison between BLM variants

Algorithm	Ins	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	nMin	nScal	nGood	nLP	nBox	nSIS	nZL
Basic	16	2810	115	680.0	522.1	157.0	8591	2848	14	2881	-	22687	2839	2302	2830
	17	2986	110	865.4	668.1	196.4	9058	3003	7	3045	-	24920	2995	2442	2987
	18	2775	101	739.1	571.8	166.7	8399	2787	12	2813	-	22478	2778	2253	2769
	19	6204	181	2060.2	1650.6	407.2	18721	6217	18	6269	-	53756	6205	5145	6193
	20	3145	100	849.7	646.1	202.6	9596	3182	34	3198	-	26839	3168	2704	3154
Avg.		3584	121.4	1038.9	811.7	226.0	10873	3607.4	17	3641.2	-	30136	3597	2969.2	3586.6
SIS	16	2764	115	206.0	148.6	57.1	1934	517	14	551	335	8287	843	335	500
	17	2952	110	257.7	185.6	71.7	1960	521	7	562	349	8915	862	349	507
	18	2738	101	232.8	170.5	61.9	1909	506	12	535	350	8172	847	350	492
	19	6166	181	577.1	433.1	143.1	3805	1018	16	1080	673	18212	1681	673	1000
	20	3122	100	216.1	148.8	66.9	1748	454	33	471	336	8698	777	336	427
Avg.		3548.4	121.4	297.9	217.3	80.1	2271.2	603.2	16.4	639.8	408.6	10456.8	1002	408.6	585.2
Recursive	16	2810	115	639.7	485.7	153.2	8212	2586	-	3040	-	22340	2577	2220	2568
	17	2987	110	835.3	639.2	195.3	8757	2799	-	3159	-	24674	2791	2416	2783
	18	2775	101	691.6	527.2	163.6	8015	2512	-	2991	-	22161	2503	2138	2494
	19	6198	181	2006.9	1594.1	410.4	18023	5718	-	6587	-	53148	5706	4964	5694
	20	3144	100	824.9	618.7	205.2	9113	2828	-	3457	-	26537	2814	2458	2800
Avg.		3582.8	121.4	999.7	773	225.5	10424	3288.6	-	3846.8	-	29772	3278.2	2839.2	3267.8

Table A.1: Comparison between the different algorithms for historical instances, class C160. Times are reported in seconds.

Algorithm	Ins	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	nMin	nScal	nGood	nLP	nBox	nSIS	nZL
Basic	21	16850	294	37665.5	33307.6	4339.7	53514	17790	40	17894	-	171370	17803	15926	17766
	22	19778	410	42045.3	36927.4	5095.7	61766	20540	23	20663	-	191611	20510	18074	20476
	23	17319	343	38995.1	34570.3	4409.4	53859	17895	26	18043	-	171514	17884	15776	17871
	24	19898	460	46967.1	41632.8	5312.7	62338	20706	34	20892	-	200619	20696	17867	20682
	25	13682	337	24450.1	21268.9	3171.0	42196	14024	27	14121	-	130934	13994	12130	13964
Avg.		17505.4	368.8	38024.6	33541.4	4465.7	54734.6	18191	30	18322.6	-	173209.6	18177.4	15954.6	18151.8
SIS	21	15699	294	6106.2	4890.5	1211.5	6598	1741	40	1849	1227	43443	2956	1227	1722
	22	18840	410	7788.7	6297.5	1485.3	8613	2287	23	2421	1595	52662	3850	1595	2233
	23	16449	343	6977.4	5607.1	1366.8	7459	1982	26	2131	1338	46627	3309	1338	1961
	24	18546	460	9535.6	7899.2	1630.3	10070	2672	34	2884	1808	56219	4468	1808	2679
	25	13239	337	5329.1	4296.6	1029.2	6578	1753	26	1853	1193	37911	2916	1193	1700
Avg.		16554.6	368.8	7147.4	5798.2	1344.6	7863.6	2087	29.8	2227.6	1432.2	47372.4	3499.8	1432.2	2059
Recursive	21	16831	294	37201.4	32767.3	4417.0	52297	16979	-	18339	-	170040	16971	15611	16955
	22	19763	410	41650	36417.7	5210.1	60312	19600	-	21112	-	189879	19568	17830	19536
	23	17315	343	38684.6	34149.4	4521.7	52585	17042	-	18501	-	170082	17030	15546	17018
	24	19890	460	46196.8	40725.8	5449.5	60571	19576	-	21419	-	198351	19566	17509	19552
	25	13667	337	24635	21314.5	3310.4	40899	13143	-	14613	-	129295	13113	11834	13083
Avg.		17493.2	368.8	37673.6	33074.9	4581.7	53332.8	17268	-	18796.8	-	171529.4	17249.6	15666	17228.8

Table A.2: Comparison between the different algorithms for historical instances, class C320. Times are reported in seconds.

Algorithm	Ins	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	nMin	nScal	nGood	nLP	nBox	nSIS	nZL
Basic	A	15002	5001	3996.1	1803.6	1875.2	52010	14653	398	22306	-	104487	14613	3259	14567
	B	15002	5001	3853.1	1687.7	1849.7	51738	14561	493	22123	-	103984	14519	3154	14475
	C	15002	5001	3981.6	1850.5	1819.9	51695	14544	518	22089	-	103755	14495	3193	14446
	D	15002	5001	3842.0	1699.0	1824.9	51744	14601	449	22093	-	104201	14563	3153	14521
	E	15002	5001	3948.9	1767.4	1864.2	51745	14574	474	22123	-	104228	14542	3116	14504
Avg.		15002	5001	3924.3	1761.6	1846.8	51786.4	14586.6	466.4	22146.8	-	104131	14546.4	3175	14502.6
SIS	A	15002	5001	4454.9	2278.3	1852.4	53523	16157	326	20820	63	113195	16218	63	16064
	B	15002	5001	4909.8	2623.9	1976.3	53881	16322	386	20749	102	113185	16397	102	16191
	C	15002	5001	4267.5	1969.3	1973.5	53239	16081	422	20592	63	112113	16126	63	15954
	D	15002	5001	4334.7	2025.1	1985.4	53682	16209	373	20798	93	113018	16277	93	16092
	E	15002	5001	4642.9	2328.5	1982.3	53552	16187	391	20704	83	112541	16239	83	16065
Avg.		15002	5001	4522.0	2245.0	1954.0	53575.4	16191.2	379.6	20732.6	80.8	112810.4	16251.4	80.8	16073.2
Recursive	A	15002	5001	2861.8	1383.9	1184.6	29711	3638	-	22435	-	72937	3616	1	3594
	B	15002	5001	2788.4	1301.3	1192.8	29587	3614	-	22359	-	72826	3594	1	3574
	C	15004	5001	2827.3	1350.4	1184.6	29538	3597	-	22344	-	72824	3578	1	3559
	D	15002	5001	2788.5	1306.2	1187.8	29664	3630	-	22404	-	72986	3608	1	3586
	E	15002	5001	2851.8	1367.3	1189.4	29641	3689	-	22263	-	72909	3664	1	3639
Avg.		15002.4	5001	2823.6	1341.8	1187.8	29628.2	3633.6	-	22361	-	72896.4	3612	1	3590.4

Table A.3: Comparison between the different algorithms for generated instances, $n = 5000$. Times are reported in seconds.

Algorithm	Ins	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	nMin	nScal	nGood	nLP	nBox	nSIS	nZL
Basic	A	22502	7501	9268.1	4379.1	4008.0	81466	21893	669	37011	-	157416	21833	8149	21765
	B	22502	7501	9093.0	4228.6	3981.7	81333	21847	705	36934	-	157264	21796	8101	21741
	C	22502	7501	8912.3	4065.1	3984.0	81446	21847	704	37048	-	157339	21797	8192	21747
	D	22502	7501	8460.9	3818.5	3814.2	81301	21807	746	36941	-	157198	21757	8058	21705
	E	22502	7501	8672.5	4058.3	3786.0	81444	21840	708	37056	-	157230	21794	8240	21742
Avg.		22502	7501	8881.4	4109.9	3914.8	81398	21846.8	706.4	36998	-	157289.4	21795.4	8148	21740
SIS	A	22502	7501	11346.9	6662.2	3791.6	79742	24039	595	30955	114	168338	24123	114	23916
	B	22502	7501	12163.8	7132.0	4131.0	79662	23994	591	30960	123	167968	24073	123	23854
	C	22501	7501	12520.1	7754.7	3879.3	79925	24106	595	31013	105	168628	24178	105	23968
	D	22502	7501	12395.5	7320.7	4155.7	79593	23959	628	30944	103	168028	24034	103	23817
	E	22503	7501	11874.0	6862.9	4129.8	79772	24007	621	31027	110	168178	24086	110	23889
Avg.		22502	7501	12060.1	7146.5	4017.5	79738.8	24021	606	30979.8	111	168228	24098.8	111	23888.8
Recursive	A	22502	7501	6544.4	3209.7	2535.2	44466	5305	-	33856	-	109695	5278	1	5251
	B	22502	7501	6329.9	3016.2	2517.2	44485	5382	-	33721	-	109677	5345	1	5308
	C	22502	7501	6341.6	2964.6	2562.0	44445	5257	-	33931	-	109532	5228	1	5199
	D	22502	7501	6172.0	2822.3	2538.8	44415	5136	-	34143	-	109698	5099	1	5062
	E	22505	7501	6345.6	2997.1	2538.9	44451	5284	-	33883	-	109541	5252	2	5220
Avg.		22502.6	7501	6346.7	3002.0	2538.4	44452.4	5272.8	-	33906.8	-	109628.6	5240.4	1.2	5208

Table A.4: Comparison between the different algorithms for generated instances, $n = 7500$. Times are reported in seconds.

APPENDIX B
WEIGHT SPACE DECOMPOSITION

B.1 Running Example

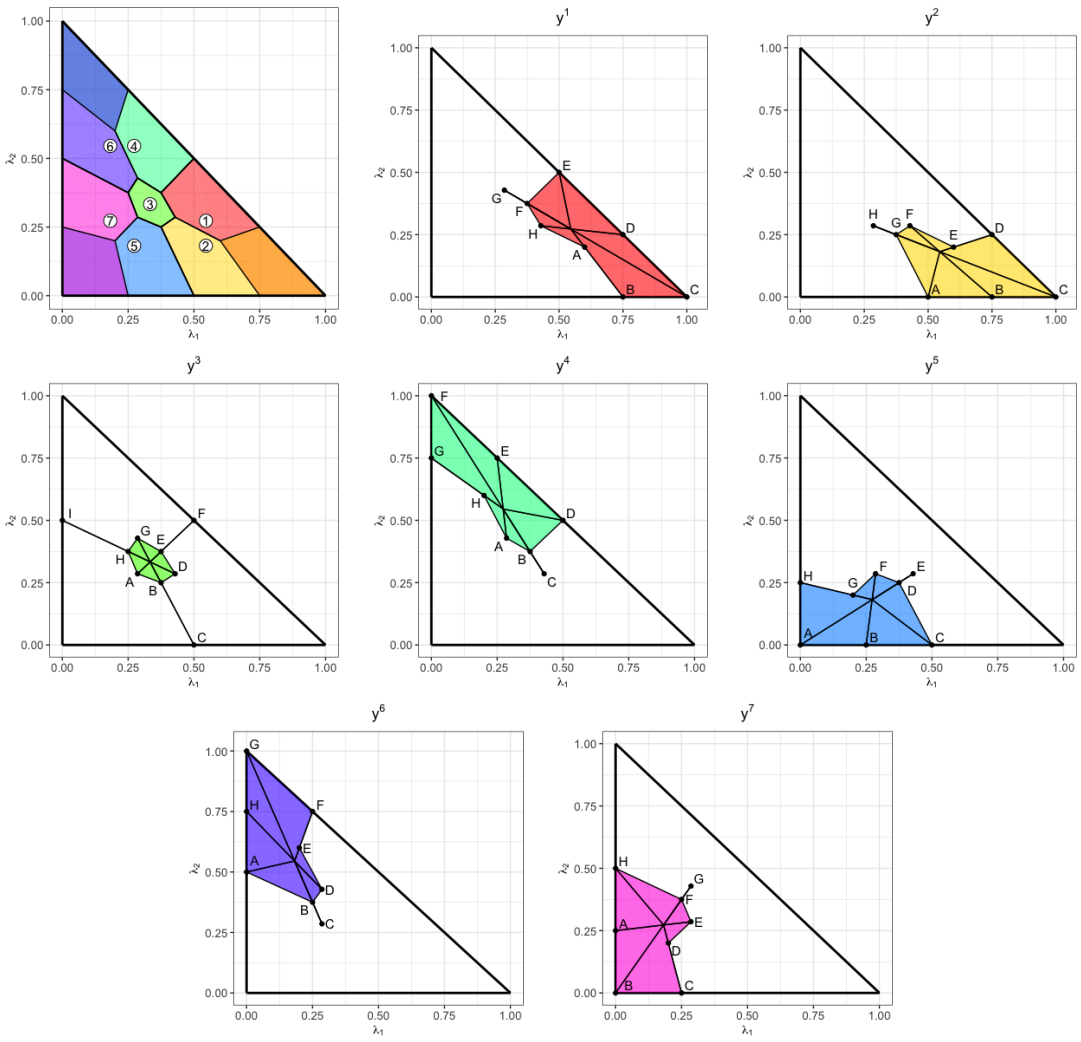
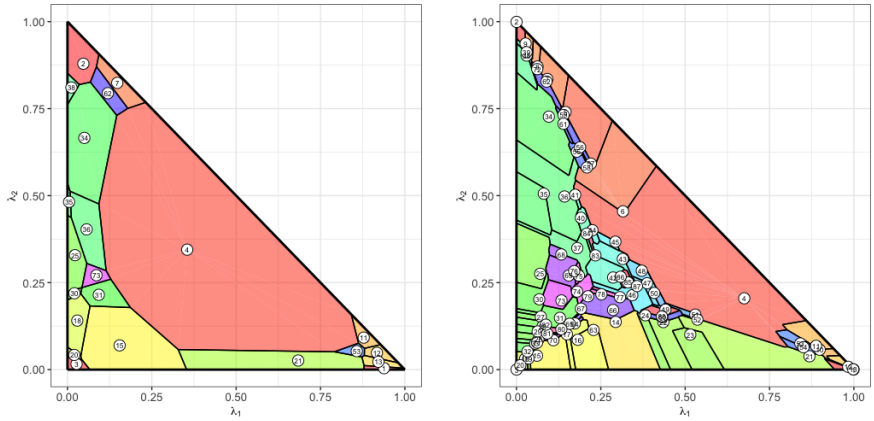


Figure B.1: All weight set components for Example 10. Labeled local nadir weights correspond to Table B.1, which gives the contributing set for each.

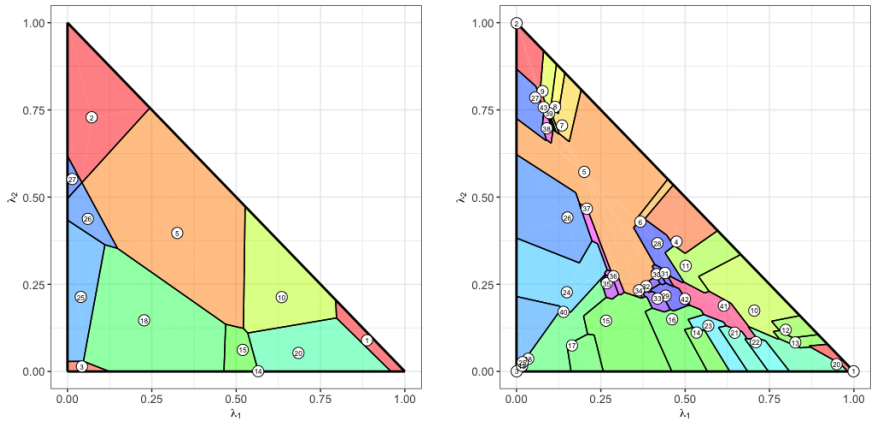
Table B.1: Each element indicates the labels (e.g., $i, j, b(k)$ for $y^i, y^j, y^{b(k)}$, respectively) for the complete contributing set per local nadir weight. Each column is associated with one weight set component, and each row corresponds to labels given in Figure B.1.

Label	y^1	y^2	y^3	y^4	y^5	y^6
A	1,2	2,3,5,b(2)	2,3,5,6,7	1,3,4,6,7	5,b(1),b(2)	3,6,7,b(1)
B	1,2,b(2)	1,2,b(2)	2,3,5	1,3,4	5,7,B(2)	3,6,7
C	1,2,b(2),b(3)	1,2,b(2),b(3)	2,3,5,b(2)	1,2,3,4,5	2,5,b(2)	2,5,6,7
D	1,2,b(3)	1,2,b(3)	1,2,3,4,5	1,3,4,b(3)	2,3,5	1,3,4,6,7
E	1,3,4,b(3)	1,2	1,3,4	4,6,b(3)	1,3,4,5	4,6
F	1,3,4	1,2,4,5	1,3,4,b(3)	4,b(1),b(3)	2,3,5,6,7	4,6,b(3)
G	1,3,4,6,7	2,3,5	1,3,4,6,7	4,6,b(1)	5,7	6,b(1),b(3)
H	1,2,4,5	2,3,5,6,7	3,6,7	4,6	5,7,b(1)	4,6,b(1)
I			3,6,7,b(1)			

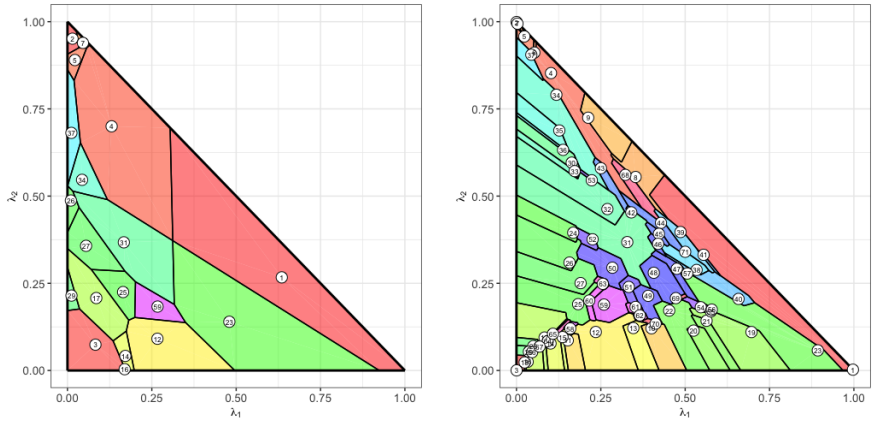
B.2 Comparing Decompositions



(a) Sample set with 88 ND images, including 21 ESND images.



(b) Sample set with 43 ND images, including 11 ESND images.



(c) Sample set with 71 ND images, including 18 ESND images.

Figure B.2: Comparing weight set decompositions with respect to weighted sum scalarization (left) and weighted Tchebychev scalarization (right) for sample ND sets. Color key is consistent between pairs. Continued from Figure 4.7.

GLOSSARY

ϵ-frontier Define an *ϵ-approximation of the ND frontier $\mathcal{N}$*, or simply an *ϵ-frontier*, $\mathcal{N}_\epsilon$, to be a set of points in criteria space such that, for fixed $\epsilon > 0$, (1) for all $z \in \mathcal{N}$, there exists $\bar{z} \in \mathcal{N}_\epsilon$ with $||z - \bar{z}||_2 \leq \epsilon$, and (2) for all $\bar{z} \in \mathcal{N}_\epsilon$, there exists $z \in \mathcal{N}$ with $||z - \bar{z}||_2 \leq \epsilon$. 51

ith dimensional contributing set Given LNP $n := y^N(\bar{Y})$ and $i \in \{1, \ldots, p\}$, the *ith dimensional contributing set of n*, denoted $C^i(n)$, is the subset of $\bar{Y}$ that contributes to LNP n in component i. 118

biobjective mixed integer linear program (MOP) with the added restriction that $p = 2$ and $\mathcal{X} \subseteq \mathbb{Z}^{n_I} \times \mathbb{R}^{n_C}$ for finite n_I and n_C. 4, 15

complete For a given LNP, its contributing set $\bar{Y}$ is *complete* with respect to Y_N if for all $y' \in Y_N \setminus \bar{Y}$, $y^N(\bar{Y}) \neq y^N(\bar{Y} \cup \{y'\})$. 118

component-wise orderings For $y, \bar{y} \in \mathbb{R}^p$, $y \leqq \bar{y}$ is defined by $y_i \leq \bar{y}_i$ for all $i = 1; \ldots, p$; $y \leq \bar{y}$ is defined by $y \leqq \bar{y}$ and $y \neq \bar{y}$; and $y < \bar{y}$ is defined by $y_i < \bar{y}_i$ for all $i = 1, \ldots, p$. 4

compromise region For $\lambda^1, \ldots, \lambda^N \in \Lambda$, $N \geq 3$ conv$(\lambda^1, \ldots, \lambda^N)$ is called the *compromise region*. 150

decision space $\mathbb{R}^n$ where n is the number of variables. 3

dichotomic search Given a biobjective (MOP) ($p = 2$), *dichotomic search* iteratively solves the weighted sum scalarization (1.4) to discover all ESND images. At each

step of the procedure, ESND images are compared as a pair, and they are either certified as adjacent on the convex hull of $\mathcal{Y}$, or a new ESND image is found which proves them to be nonadjacent. 7

dimensional weight set component Let $y \in \mathcal{Y}_{wN}$. We define for $i = 1, \ldots, p$ the *ith dimensional weight set component* by

$$\Lambda(y, i) := \{\lambda \in \Lambda(y) : \lambda_i y_i \geq \lambda_k y_k \text{ for all } k = 1, \ldots, p\}$$

. 76

dominates (strictly dominates) For minimization, a feasible solution x *(strictly) dominates* another solution x' if $f(x) \leq f(x')$ ($f(x) < f(x')$). 4

efficient (weakly efficient) For minimization, a feasible solution $x^* \in X$ is *(weakly) efficient* if there does not exist another feasible solution $x \in X$ such that $f(x) \leq f(x^*)$ ($f(x) < f(x^*)$). 4

extreme supported Supported ND images that are also extreme points of the convex hull of $\mathcal{Y}$ are called *extreme supported nondominated (ESND) images*. 6, 15, 70, 108

image space $\mathbb{R}^p$ where p is the number of objectives, also known as the criterion or objective space. 3

inner approximation An *inner approximation*, $\{\Lambda^-(y)\}_{y \in Y'}$ is such that the approximated weight set components are subsets of the true components, i.e., $\Lambda^-(y) \subseteq \Lambda(y)$ for all $y \in Y'$. 146

kernel weight For $y \in \mathcal{Y}_{wN}$ we denote the *kernel weight* or *kernel vertex* (also known as

T-vertex [64]) of y by $\lambda(y)$ and define it by

$$\lambda_i(y) := \frac{1}{y_i} \frac{1}{\sum_{j=1}^{p} 1/y_j} \text{ for } i = 1, \ldots, p$$

. 74, 113

lexicographic scalarization The *lexicographic scalarization* hierarchically minimizes the objectives in turn. We define only the biobjective case: the case of minimizing $f_1(x)$ and then $f_2(x)$ is denoted by

$$\eta = \text{lexmin}\{(f_1(x), f_2(x)) : \ x \in \mathcal{X}\}$$

. 5, 20

local dimension Let a weight λ and a set S be given such that $\lambda \in S \subseteq \Lambda$. Further, we denote with $\mathcal{P}$ the set of all polytopes in $\mathbb{R}^p$. Then, the *local dimension of λ with respect to S* is given by

$$\dim(\lambda, S) := \max_{P \in \mathcal{P}, \lambda \in P \subseteq S} \dim(P)$$

. 96

local nadir point Given a subset of weakly ND images, $\bar{Y} = \{y^1, \ldots, y^{\bar{R}}\} \subseteq \mathcal{Y}_{wN}$, we define the *local nadir point* $y^N(\bar{Y})$ by

$$y_i^N(\bar{Y}) = \max_{r=1,\ldots,\bar{R}} y_i^r \text{ for } i = 1, \ldots, p$$

. 82, 118

local nadir weight We call the kernel weight of $y^N(\bar{Y})$, $\lambda^N(\bar{Y}) := \lambda(y^N(\bar{Y}))$, the *local nadir weight* . 82, 120

maximal A given LNP with contributing set $\bar{Y}$ is *maximal* with respect to Y_N if $y^N(\bar{Y})$ is not strictly dominated by any $y' \in Y_N$, but $y^N(\bar{Y} \cup \{y'\})$ is strictly dominated by some $y' \in Y_N$. 118

multiobjective discrete optimization problem (MOP) with the added restriction that $\mathcal{X} \subseteq \mathbb{Z}^n$ for finite n. 4, 67

multiobjective optimization problem A multiobjective optimization problem, with $p \in \mathbb{N}, p \geq 2$ objectives can be stated as

$$\begin{aligned} \min f(x) &= \min \left(f_1(x), \ldots, f_p(x)\right)^\top \\ &\text{s.t. } x \in \mathcal{X}, \end{aligned}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$, for $n \in \mathbb{N}$, is called the *feasible set*, and $f = (f_1, \ldots, f_p)^\top : \mathbb{R}^n \to \mathbb{R}^p$ is the (vector-valued) *objective function*. 3

nondominated An image $y = f(x)$ is *nondominated (ND)* if x is efficient and denote by $\mathcal{Y}_N$ the set of ND images. 4, 15, 67

nondominated frontier The set of ND images, denoted $\mathcal{Y}_N$. 4

nondominated line segment Define S to be the index set of all feasible integer solutions in $\mathcal{X}_I$. For a slice problem with index $s \in S$, we denote its slice by N^s. If a slice problem with index $s \in S$ contributes to the ND frontier of the BOMIP, so $N^s \cap \mathcal{N}$ is nonempty, then we write $N^s \cap \mathcal{N} = \{L_1^s, L_2^s, ..., L_{n(s)}^s\}$, where $n(s)$ is the number of line segments contributed to the ND frontier by the slice problem. (If $N^s \cap \mathcal{N}$ is empty, then $n(s) = 0$.) Each of the line segments, L_i^s for some $s \in S$ and $i = 1, \ldots, n(s)$, is a *nondominated line segment*. 19

nonnegative orthant $\mathbb{R}^p_{\leqq} := \{t \in \mathbb{R}^p : t \leqq 0\}$. The sets $\mathbb{R}^p_{\leq}$ and $\mathbb{R}^p_{>}$ are defined analogously. 4

outer approximation An *outer approximation*, $\{\Lambda^+(y)\}_{y\in Y'}$ for $Y' \subset Y_N$, for a weight set decomposition $\{\Lambda(y)\}_{y\in Y_N}$ is an approximation for a proper subset of the images where the approximated weight set components are supersets of the true components, i.e., $\Lambda(y) \subseteq \Lambda^+(y)$ for all $y \in Y'$. 146

polytopal complex A *polytopal complex* $\mathcal{C}$ is a finite collection of polytopes in $\mathbb{R}^d$ such that (1) the empty polytope is in $\mathcal{C}$, (2) if $P \in \mathcal{C}$, then all the faces of P are also in $\mathcal{C}$, and (3) the intersection $P \cap Q$ of two polytopes $P, Q \in \mathcal{C}$ is a face of both P and Q . 72

polytopal subdivision A *polytopal subdivision* of a set $S \subseteq \mathbb{R}^d$ is a polytopal complex $\mathcal{C}$ with the underlying set $\bigcup_{P\in\mathcal{C}} P = S$. 72

slice The ND frontier of a BOLP slice problem consists of a (connected) set of (closed) line segments; we call this a *slice*. 19

slice problem Given $x_I \in \mathcal{X}_I$, the biobjective linear program obtained from fixing the integer variables to x_I is called the *slice problem for* x_I [53]. 19

star-shaped A set $S \subseteq \mathbb{R}^p$ is *star-shaped*, if there exists a $y \in S$ such that $\theta y+(1-\theta)\bar{y} \in S$ for all $\bar{y} \in S$ and all $\theta \in (0, 1)$. The set of all such images y is called *kernel* of S and is denoted by $\ker(S)$. 72, 113

supported Weighted sum scalarizations yield *supported* ND images, which are located on the convex hull of the set of images. 6

vertical gap We define a *vertical gap* as an interval $(y^-, y^+) \subset \mathbb{R}$ such that no ND image p exists with $p_2 \in (y^-, y^+)$ but where there does exist a ND image p^- with $p_2^- = y^-$ and either a ND image p^+ with $p_2^+ = y^+$ or a sequence of ND images $\{p^0, p^1, ...\}$ with $\lim_{n\to\infty} p_2^n = y^+$ (or both). A *horizontal gap* can be defined similarly. 19

weakly nondominated An image $y = f(x)$ is *weakly nondominated* if x is weakly efficient and denote by $\mathcal{Y}_{wN}$ the set of weakly ND images. 4

weight set Defined as $\Lambda := \left\{\lambda \in \mathbb{R}^p_{\geqq} : \sum_{k=1}^p \lambda_k = 1\right\}$. 73, 108

weight set component for $y \in Y$, we define *weight set components* as $\Lambda^{WS}(y) := \left\{\lambda \in \Lambda : \lambda^\top y \leq \ \lambda^\top \bar{y}\right.$ for the weighted sum scalarization and $\Lambda(y) := \left\{\lambda \in \Lambda : \|y\|^\lambda_\infty \leq \ \|\bar{y}\|^\lambda_\infty \text{ for all } \bar{y} \in Y\right\}$ for the weighted Tchebychev scalarization . 74, 108, 190

weighted Tchebychev scalarization Let $\|y\|^\lambda_\infty := \max_{i=1,\ldots,p}\{|\lambda_i\ y_i|\}$. With a *reference point* $s \in \mathbb{R}^p$ and a given weight vector $\lambda \in \mathbb{R}^p_{\geq}$, the *weighted Tchebychev scalarization with respect to* λ can be stated as

$$\min\ \{\|f(x) - s\|^\lambda_\infty\ :\ x \in \mathcal{X}\}$$

. 6, 68, 108

weighted sum scalarization For vector $\lambda \in \mathbb{R}^p_{\geq}$, we refer to

$$\min\{\lambda^T f(x) :\ x \in \mathcal{X}\}$$

as the *weighted sum scalarization with respect to* λ. 6, 107

ACRONYMS

εTCM ε,Tabu-Constraint Method. 16

BBM Balanced Box Method. 16

BLM Boxed Line Method. 16

BOLP Biobjective Linear Program. 19

BOMIP Biobjective Mixed Integer Linear Program. 4, 15

ESND Extreme Supported Nondominated. 6, 15, 70, 108

IP Integer Program, including mixed integer linear programs. 5

LNP Local Nadir Point. 118

LP Linear Program. 15

MODO Multiobjective Discrete Optimization Problem. 4, 16, 67

MOP Multiobjective Optimization Problem. 3, 156

ND Nondominated. 4, 67

NDLS Nondominated Line Segment. 19

TOMIP Triobjective Mixed Integer Linear Program. 156

TSA Triangle Splitting Algorithm. 15

REFERENCES

[1] D. Lei, "Multi-objective production scheduling: A survey," *The International Journal of Advanced Manufacturing Technology*, vol. 43, no. 9-10, pp. 926–938, 2009.

[2] J. Malczewski, "GIS-based multicriteria decision analysis: A survey of the literature," *International Journal of Geographical Information Science*, vol. 20, no. 7, pp. 703–726, 2006.

[3] R. Z. Farahani, M. SteadieSeifi, and N. Asgari, "Multiple criteria facility location problems: A survey," *Applied Mathematical Modelling*, vol. 34, no. 7, pp. 1689–1709, 2010.

[4] A. Rais and A. Viana, "Operations research in healthcare: A survey," *International Transactions in Operational Research*, vol. 18, no. 1, pp. 1–31, 2011.

[5] D. White, "A bibliography on the applications of mathematical programming multiple-objective methods," *Journal of the Operational Research Society*, vol. 41, no. 8, pp. 669–691, 1990.

[6] C. A. Coello, "An updated survey of GA-based multiobjective optimization techniques," *ACM Computing Surveys (CSUR)*, vol. 32, no. 2, pp. 109–143, 2000.

[7] K. Deb, *Multi-objective optimization using evolutionary algorithms*. New York: John Wiley & Sons, 2001, vol. 16.

[8] A. Zhou, B. Y. Qu, H. Li, S. Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multi-objective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

[9] ——, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

[10] Q. Gu, Q. Wang, N. N. Xiong, S. Jiang, and L. Chen, "Surrogate-assisted evolutionary algorithm for expensive constrained multi-objective discrete optimization problems," *Complex & Intelligent Systems*, pp. 1–20, 2021.

[11] N. H. Luong, H. La Poutré, and P. A. Bosman, "Multi-objective gene-pool optimal mixing evolutionary algorithm with the interleaved multi-start scheme," *Swarm and Evolutionary Computation*, vol. 40, pp. 238–254, 2018.

[12] G Mavrotas and D Diakoulaki, "A branch and bound algorithm for mixed zero-one multiple objective linear programming," *European Journal of Operational Research*, vol. 107, pp. 530–541, 1998.

[13] G. Mavrotas and D. Diakoulaki, "Multi-criteria branch and bound: A vector maximization algorithm for mixed 0-1 multiple objective linear programming," *Applied Mathematics and Computation*, vol. 171, pp. 53–71, 2005.

[14] T. Vincent, F. Seipp, S. Ruzika, A. Przybylski, and X. Gandibleux, "Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for biobjective case," *Computers and Operations Research*, vol. 40, pp. 498–509, 1 2013.

[15] M. Ehrgott and X. Gandibleux, "A survey and annotated bibliography of multiobjective combinatorial optimization," *OR Spectrum*, vol. 22, no. 4, pp. 425–460, 2000.

[16] ——, *Multiple Criteria Optimization: state of the art annotated bibliographic surveys*, ser. International Series in Operations Research and Management Science. Boston, MA: Kluwer Academic Publilshers, 2002, vol. 52.

[17] M. Ehrgott, *Multicriteria optimization*, 2nd. Berlin, Heidelberg: Springer-Verlag, 2005, ISBN: 3540213988.

[18] L. Zadeh, "Optimality and non-scalar-valued performance criteria," *IEEE Transactions on Automatic Control*, vol. 8, no. 1, pp. 59–60, 1963.

[19] A. M. Geoffrion, "Proper efficiency and the theory of vector maximization," *Journal of Mathematical Analysis and Applications*, vol. 22, no. 3, pp. 618 –630, 1968.

[20] R. E. Steuer and E.-U. Choo, "An interactive weighted tchebycheff procedure for multiple objective programming," *Mathematical Programming*, vol. 26, no. 3, pp. 326–344, 1983.

[21] J. L. Cohon, *Multiobjective programming and planning*, ser. Math. Sci. Eng. New York, NY: Academic Press, 1978.

[22] Y. P. Aneja and K. P. Nair, "Bicriteria transportation problem," *Management Science*, vol. 25, no. 1, pp. 73–78, 1979.

[23] A. Przybylski, X. Gandibleux, and M. Ehrgott, "A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme," *INFORMS Journal on Computing*, vol. 22, no. 3, pp. 371–386, 2010.

[24] R. Steuer, "Multiple criteria optimization: Theory, computation and application. john wliey & sons," *Inc.: New York, NY, USA*, 1986.

[25] B. Soylu and H. Katip, "A multiobjective hub-airport location problem for an airline network design," *European Journal of Operational Research*, vol. 277, no. 2, pp. 412–425, 2019.

[26] B. Soylu and G. B. Yıldız, "An exact algorithm for biobjective mixed integer linear programming problems," *Computers & Operations Research*, vol. 72, pp. 204–213, 2016.

[27] R. Raeesi and K. G. Zografos, "The multi-objective steiner pollution-routing problem on congested urban road networks," *Transportation Research Part B: Methodological*, vol. 122, pp. 457–485, 2019.

[28] T. Stidsen, K. A. Andersen, and B. Dammann, "A branch and bound algorithm for a class of biobjective mixed integer programs," *Management Science*, vol. 60, no. 4, pp. 1009–1032, 2014.

[29] N. Boland, H. Charkhgard, and M. Savelsbergh, "The quadrant shrinking method: A simple and efficient algorithm for solving tri-objective integer programs," *European Journal of Operational Research*, vol. 260, no. 3, pp. 873–885, 2017.

[30] J. Hu and S. Mehrotra, "Robust and stochastically weighted multiobjective optimization models and reformulations," *Operations research*, vol. 60, no. 4, pp. 936–953, 2012.

[31] N. Boland, H. Charkhgard, and M. Savelsbergh, "A criterion space search algorithm for biobjective mixed integer programming: The triangle splitting method," *INFORMS Journal on Computing*, vol. 27, no. 4, pp. 597–618, 2015.

[32] T. Perini, N. Boland, D. Pecin, and M. Savelsbergh, "A criterion space method for biobjective mixed integer programming: The boxed line method," *INFORMS Journal on Computing*, vol. 32, no. 1, pp. 16–39, 2020.

[33] A. Fattahi and M. Turkay, "A one direction search method to find the exact nondominated frontier of biobjective mixed-binary linear programming problems," *European Journal of Operational Research*, vol. 266, no. 2, pp. 415–425, 2018.

[34] B. Soylu, "The search-and-remove algorithm for biobjective mixed-integer linear programming problems," *European Journal of Operational Research*, vol. 268, no. 1, pp. 281–299, 2018.

[35] D. Emre, "Exact solution algorithms for biobjective mixed integer programming problems," Ph.D. dissertation, Bilkent University, 2020.

[36] J. Sylva and A. Crema, "A method for finding the set of non-dominated vectors for multiple objective integer linear programs," *European Journal of Operational Research*, vol. 158, no. 1, pp. 46–55, 2004.

[37] M. Laumanns, L. Thiele, and E. Zitzler, "An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method," *European Journal of Operational Research*, vol. 169, no. 3, pp. 932–942, 2006.

[38] A. Przybylski, X. Gandibleux, and M. Ehrgott, "A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives," *Discrete Optimization*, vol. 7, no. 3, pp. 149–165, 2010.

[39] M. Ozlen, B. A. Burton, and C. A. MacRae, "Multi-objective integer programming: An improved recursive algorithm," *Journal of Optimization Theory and Applications*, vol. 160, no. 2, pp. 470–482, 2014.

[40] G. Kirlik and S. Sayın, "A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems," *European Journal of Operational Research*, vol. 232, no. 3, pp. 479–488, 2014.

[41] K. Klamroth, R. Lacour, and D. Vanderpooten, "On the representation of the search region in multi-objective optimization," *European Journal of Operational Research*, vol. 245, no. 3, pp. 767 –778, 2015.

[42] K. Dächert and K. Klamroth, "A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems," *Journal of Global Optimization*, vol. 61, Apr. 2015.

[43] N. Boland, H. Charkhgard, and M. Savelsbergh, "The l-shape search method for tri-objective integer programming," *Mathematical Programming Computation*, vol. 8, no. 2, pp. 217–251, 2016.

[44] K. Dächert, K. Klamroth, R. Lacour, and D. Vanderpooten, "Efficient computation of the search region in multi-objective optimization," *European Journal of Operational Research*, vol. 260, no. 3, pp. 841 –855, 2017.

[45] O. Turgut, E. Dalkiran, and A. E. Murat, "An exact parallel objective space decomposition algorithm for solving multi-objective integer programming problems," *Journal of Global Optimization*, vol. 75, no. 1, pp. 35–62, 2019.

[46] S. Tamby and D. Vanderpooten, "Enumeration of the nondominated set of multiobjective discrete optimization problems," *INFORMS Journal on Computing*, 2020.

[47] H. P. Benson and E. Sun, "Outcome space partition of the weight set in multiobjective linear programming," *Journal of Optimization Theory and Applications*, vol. 105, no. 1, pp. 17–36, 2000.

[48] H. P. Benson and E. Sun, "A weight set decomposition algorithm for finding all efficient extreme points in the outcome set of a multiple objective linear program," *European Journal of Operational Research*, vol. 139, no. 1, pp. 26–41, 2002.

[49] M. J. Alves and J. P. Costa, "Graphical exploration of the weight space in three-objective mixed integer linear programs," *European Journal of Operational Research*, vol. 248, no. 1, pp. 72–83, 2016.

[50] P. Halffmann, T. Dietz, A. Przybylski, and S. Ruzika, "An inner approximation method to compute the weight set decomposition of a triobjective mixed-integer problem," *Journal of Global Optimization*, Mar. 2020.

[51] P. Halffmann, "Advances in multiobjective optimisation - scalarisation, approximation, and complexity," Ph.D. dissertation, Technische Universität Kaiserslautern, 2021.

[52] N. Boland, H. Charkhgard, and M. Savelsbergh, "A criterion space search algorithm for biobjective integer programming: The balanced box method," *INFORMS Journal on Computing*, vol. 27, no. 4, pp. 735–754, 2015.

[53] P. Belotti, B. Soylu, and M. M. Wiecek, "A branch-and-bound algorithm for biobjective mixed-integer programs," *Optimization Online*, 2013.

[54] C. H. Papadimitriou and M. Yannakakis, "On the approximability of trade-offs and optimal access of web sources," in *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, IEEE, 2000, pp. 86–92.

[55] S. Vassilvitskii and M. Yannakakis, "Efficiently computing succinct trade-off curves," *Theoretical Computer Science*, vol. 348, no. 2-3, pp. 334–356, 2005.

[56] F. Bökler and P. Mutzel, "Output-sensitive algorithms for enumerating the extreme nondominated points of multiobjective combinatorial optimization problems.," in *Algorithms – ESA 2015. 23rd annual European symposium, Patras, Greece, September 14–16, 2015. Proceedings*, Berlin: Springer, 2015, pp. 288–299, ISBN: 978-3-662-48349-7/pbk; 978-3-662-48350-3/ebook.

[57] O. Özpeynirci and M. Köksalan, "An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer programs," *Management Science*, vol. 56, pp. 2302–2315, Dec. 2010.

[58] A. Przybylski, X. Gandibleux, and M. Ehrgott, "A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme," *INFORMS Journal on Computing*, vol. 22, no. 3, pp. 371–386, 2010.

[59] A. M. Geoffrion, "Solving bicriterion mathematical programs," *Operations Research*, vol. 15, pp. 39–54, Feb. 1967.

[60] V. J. Bowman, "On the relationship of the tchebycheff norm and the efficient frontier of multiple-criteria objectives," in *Multiple Criteria Decision Making*, H. Thiriez and S. Zionts, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1976, pp. 76–86, ISBN: 978-3-642-87563-2.

[61] A. P. Wierzbicki, "The use of reference objectives in multiobjective optimization," in *Multiple Criteria Decision Making Theory and Application*, G. Fandel and T. Gal, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1980, pp. 468–486, ISBN: 978-3-642-48782-8.

[62] T. Holzmann and J. Cole Smith, "Solving discrete multi-objective optimization problems using modified augmented weighted tchebychev scalarizations," *European Journal of Operational Research*, vol. 271, May 2018.

[63] K. Miettinen, F. Ruiz, and A. P. Wierzbicki, *Introduction to Multiobjective Optimization: Interactive Approaches*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 27–57, ISBN: 978-3-540-88908-3.

[64] M. Luque, F. Ruiz, and R. E. Steuer, "Modified interactive chebyshev algorithm (mica) for convex multiobjective programming," *European Journal of Operational Research*, vol. 204, no. 3, pp. 557 –564, 2010.

[65] M. J. Alves and J. Clímaco, "An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound," *European Journal of Operational Research*, vol. 124, no. 3, pp. 478 –494, 2000.

[66] B. Bozkurt, J. W. Fowler, E. S. Gel, B. Kim, M. Köksalan, and J. Wallenius, "Quantitative comparison of approximate solution sets for multicriteria optimization problems with weighted tchebycheff preference function," *Operations Research*, vol. 58, no. 3, pp. 650–659, 2010.

[67] P.-L. Yu and M. Zeleny, "The set of all nondominated solutions in linear cases and a multicriteria simplex method," *Journal of Mathematical Analysis and Applications*, vol. 49, no. 2, pp. 430–468, 1975.

[68] M. J. Alves and J. P. Costa, "Graphical exploration of the weight space in three-objective mixed integer linear programs," *European Journal of Operational Research*, vol. 248, no. 1, pp. 72–83, 2016.

[69] B. Schulze, K. Klamroth, and M. Stiglmayr, "Multi-objective unconstrained combinatorial optimization: A polynomial bound on the number of extreme supported solutions," *Journal of Global Optimization*, vol. 74, no. 3, pp. 495–522, 2019.

[70] P. K. Eswaran, A. Ravindran, and H. Moskowitz, "Algorithms for nonlinear integer bicriterion problems," *Journal of Optimization Theory and Applications*, vol. 63, no. 2, pp. 261–279, 1989.

[71] T. K. Ralphs, M. J. Saltzman, and M. M. Wiecek, "An improved algorithm for solving biobjective integer programs," *Annals of Operations Research*, vol. 147, no. 1, pp. 43–70, 2006.

[72] F. P. Preparata and M. I. Shamos, *Computational geometry*, Corr. and expanded 2. print. New York u.a.: Springer, 1988, ISBN: 3-540-96131-3.

[73] G. M. Ziegler, *Lectures on polytopes*, ser. Graduate texts in mathematics 152. New York u.a.: Springer, 1995, ISBN: 3-540-94329-3.

[74] H. E. Romeijn and J. F. Dempsey, "Intensity modulated radiation therapy treatment plan optimization," *TOP*, vol. 16, no. 2, p. 215, 2008.

[75] T. C. Y. Chan, T. Craig, T. Lee, and M. B. Sharpe, "Generalized inverse multiobjective optimization with application to cancer therapy," *Operations Research*, vol. 62, no. 3, pp. 680–695, 2014.

[76] H. Charkhgard, M. Savelsbergh, and M. Talebian, "A linear programming based algorithm to solve a class of optimization problems with a multi-linear objective function and affine constraints," *Computers & Operations Research*, vol. 89, pp. 17 –30, 2018.

[77] ——, "Nondominated nash points: Application of biobjective mixed integer programming," *4OR*, vol. 16, no. 2, pp. 151–171, 2018.

[78] A. Przybylski, X. Gandibleux, and M. Ehrgott, "A two phase method for multiobjective integer programming and its application to the assignment problem with three objectives," *Discrete Optimization*, vol. 7, no. 3, pp. 149 –165, 2010.

[79] K. Dächert, P. Halffmann, K. Klamroth, S. Ruzika, and L. Schäfer, *Exact algorithms for multiobjective optimization problems with integer variables - a state of the art survey and discussion*, submitted, 2021.

[80] F. Heyde and A. Löhne, "Geometric duality in multiple objective linear programming," *SIAM Journal on Optimization*, vol. 19, pp. 836–845, Jun. 2008.

[81] H. Kaplan, N. Rubin, M. Sharir, and E. Verbin, "Efficient colored orthogonal range counting," *SIAM J. Comput.*, vol. 38, pp. 982–1011, Jan. 2008.

[82] H. P. Benson and E. Sun, "A weight set decomposition algorithm for finding all efficient extreme points in the outcome set of a multiple objective linear program," *European Journal of Operational Research*, vol. 139, pp. 26–41, Feb. 2002.

[83] G. Ceyhan, "A study of the day-ahead energy market auctions from a multi-objective perspective," 2020.

[84] S. A. B. Rasmi and M. Türkay, "Gondef: An exact method to generate all non-dominated points of multi-objective mixed-integer linear programs," *Optimization and Engineering*, vol. 20, no. 1, pp. 89–117, 2019.

[85] S. Ruzika and M. M. Wiecek, "Approximation methods in multiobjective programming," *Journal of optimization theory and applications*, vol. 126, no. 3, pp. 473–501, 2005.

[86] R Armann, "Solving multiobjective programming problems by discrete representation," *Optimization*, vol. 20, no. 4, pp. 483–492, 1989.

[87] N. Boland, H. Charkhgard, and M. Savelsbergh, "A new method for optimizing a linear function over the efficient set of a multiobjective integer program," *European journal of operational research*, vol. 260, no. 3, pp. 904–919, 2017.

[88] I. Herszterg, "Efficient algorithms for solving multi-objective optimization and large-scale transportation problems," Ph.D. dissertation, Georgia Institute of Technology, 2020.

[89] L. M. Hvattum, A. Løkketangen, and F. Glover, "Comparisons of commercial mip solvers and an adaptive memory (tabu search) procedure for a class of 0-1 integer programming problems," *Algorithmic Operations Research*, vol. 7, no. 1, pp. 13–20, 2012.

[90] A. Gleixner, L. Eifler, T. Gally, G. Gamrath, P. Gemander, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. Miltenberger, *et al.*, "The scip optimization suite 5.0," 2017.

[91] G. Gamrath, D. Anderson, K. Bestuzheva, W.-K. Chen, L. Eifler, M. Gasse, P. Gemander, A. Gleixner, L. Gottwald, K. Halbig, *et al.*, "The scip optimization suite 7.0," 2020.

[92] S. Enayati and O. Y. Özaltın, "Optimal influenza vaccine distribution with equity," *European Journal of Operational Research*, vol. 283, no. 2, pp. 714–725, 2020.

[93] J. P. Dickerson, A. D. Procaccia, and T. Sandholm, "Price of fairness in kidney exchange," CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, Tech. Rep., 2014.

[94] B. D. Bradley, T. Jung, A. Tandon-Verma, B. Khoury, T. C. Chan, and Y.-L. Cheng, "Operations research in global health: A scoping review with a focus on the themes of health equity and impact," *Health research policy and systems*, vol. 15, no. 1, pp. 1–24, 2017.

[95] K. Huang, Y. Jiang, Y. Yuan, and L. Zhao, "Modeling multiple humanitarian objectives in emergency response to large-scale disasters," *Transportation Research Part E: Logistics and Transportation Review*, vol. 75, pp. 1–17, 2015.

[96] W. J. Gutjahr and S. Fischer, "Equity and deprivation costs in humanitarian logistics," *European Journal of Operational Research*, vol. 270, no. 1, pp. 185–197, 2018.

[97] W. Fan and R. B. Machemehl, "Bi-level optimization model for public transportation network redesign problem: Accounting for equity issues," *Transportation Research Record*, vol. 2263, no. 1, pp. 151–162, 2011.

[98] T. Feng, J. Zhang, and A. Fujiwara, "Comparison of transportation network optimization with different equity measures using bilevel programming approach," Tech. Rep., 2009.

[99] S. Gupta, A. Jalan, G. Ranade, H. Yang, and S. Zhuang, "Too many fairness metrics: Is there a solution?" *Available at SSRN*, 2020.

[100] D. Bertsimas, V. F. Farias, and N. Trichakis, "The price of fairness," *Operations research*, vol. 59, no. 1, pp. 17–31, 2011.

[101] R. Cole and V. Gkatzelis, "Approximating the nash social welfare with indivisible items," in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, 2015, pp. 371–380.

[102] J. Garg, M. Hoefer, and K. Mehlhorn, "Approximating the nash social welfare with budget-additive valuations," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, 2018, pp. 2326–2340.

[103] H. Charkhgard, K. Keshanian, R. Esmaeilbeigi, and P. Charkhgard, "The magic of nash social welfare in optimization: Do not sum, just multiply," *Optimization Online*, 2020.

VITA

Tyler Perini was born in Atlanta, Georgia on December 2, 1993 and raised in the suburb of Lawrenceville. He graduated Peachtree Ridge High School in 2012. He attended The College of Charleston to earn his Bachelor of Science degree in Applied Mathematics in 2016. He entered Georgia Institute of Technology's PhD program later that year. He received the National Science Foundation's Graduate Research Fellowship in 2018.