EXACT METHODS IN FRACTIONAL COMBINATORIAL OPTIMIZATION

A Dissertation

by

OLEKSII URSULENKO

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2009

Major Subject: Industrial Engineering

EXACT METHODS IN FRACTIONAL COMBINATORIAL OPTIMIZATION

A Dissertation

by

OLEKSII URSULENKO

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Sergiy Butenko
Committee Members,	Wilbert E. Wilhelm
	Lewis Ntaimo
	Anxiao (Andrew) Jiang
	Oleg Prokopyev
Head of Department,	Brett A. Peters

December 2009

Major Subject: Industrial Engineering

ABSTRACT

Exact Methods in Fractional

Combinatorial Optimization. (December 2009)

Oleksii Ursulenko, B.S., Kyiv Polytechnical Institute;

M.S., Kyiv Polytechnical Institute

Chair of Advisory Committee: Dr. Sergiy Butenko

This dissertation considers a subclass of sum-of-ratios fractional combinatorial optimization problems (FCOPs) whose linear versions admit polynomial-time exact algorithms. This topic lies in the intersection of two scarcely researched areas of fractional programming (FP): sum-of-ratios FP and combinatorial FP. Although not extensively researched, the sum-of-ratios problems have a number of important practical applications in manufacturing, administration, transportation, data mining, etc.

Since even in such a restricted research domain the problems are numerous, the main focus of this dissertation is a mathematical programming study of the three, probably, most classical FCOPs: Minimum Multiple Ratio Spanning Tree (MMRST), Minimum Multiple Ratio Path (MMRP) and Minimum Multiple Ratio Cycle (MMRC). The first two problems are studied in detail, while for the other one only the theoretical complexity issues are addressed.

The dissertation emphasizes developing solution methodologies for the considered family of fractional programs. The main contributions include: (i) worst-case complexity results for the MMRP and MMRC problems; (ii) mixed 0–1 formulations for the MMRST and MMRC problems; (iii) a global optimization approach for the MMRST problem that extends an existing method for the special case of the sum of two ratios; (iv) new polynomially computable bounds on the optimal objective value of the considered class of FCOPs, as well as the feasible region reduction techniques

based on these bounds; (v) an efficient heuristic approach; and, (vi) a generic global optimization approach for the considered class of FCOPs.

Finally, extensive computational experiments are carried out to benchmark performance of the suggested solution techniques. The results confirm that the suggested global optimization algorithms generally outperform the conventional mixed 0–1 programming technique on larger problem instances. The developed heuristic approach shows the best run time, and delivers near-optimal solutions in most cases.

ACKNOWLEDGMENTS

I consider myself very fortunate to pursue my doctoral studies at Texas A&M University, where I met many wonderful collegues and admirable people. I am grateful to them for the part they took in my life and my studies. But above all I would like to express my sincere gratitude and respect to Dr. Sergiy Butenko, the chair of my advisory committee, for his continuous support, encouragement and, most importantly, patience.

Also, I would like to express my thanks to my committee members, Dr. Andrew Jiang, Dr. Lewis Ntaimo, Dr. Oleg Prokopyev and Dr. Wilbert Wilhelm for their guidance and support throughout the course of my research, and also for the knowledge and experience they have passed to me.

I would like to thank Dr. Svyatoslav Trukhanov, Dr. Balabhaskar "Baski" Balasundaram, Dr. Sera Kahruman, Reza Seyedshohadaie, Valentin Zingan for being such wonderful colleagues and friends at Texas A&M University.

My appreciation goes to all the faculty of Industrial and Systems Engineering Department, for creating a unique learning atmosphere and making my doctoral studies a great experience. In particular, I would like to thank Dr. Brett Peters and Dr. Guy Curry, for their guidance, and also Dr. Feldman and Dr. Smith.

I would like to thank all ISE staff and professional technical specialists: Dennis Allen, Mark Henry, Mark Hopcus, and others, for their assistance with the computational part of my research. I am sincerely greateful to Judy Meeks for her kindness and support.

Last but not least, I would like to thank my parents and my beautiful wife Zhanar for their constant support and love.

TABLE OF CONTENTS

LIST OF TABLES

TABLE Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Optimization is a phenomenon that we observe everywhere around us, starting with the laws of physics, making an object to assume a position with the minimum potential energy, to the process of evolution, that optimizes the characteristics and behavior of species so that they are adapted to its environment in the best possible way. Hence, it is not surprising that once a human being learns to perform an operation X, one of the first questions that follow is: "How to perform the operation X better ?". The optimization science is, therefore, in a general sense the formalization of one of the most natural human aspirations - the human's drive for perfection.

The development of foundations of the modern optimization techniques dates back to the times of Gauss, who invented the *steepest descent method*. Other early contributions were made by Fourier (1823) and de la Valee Poussin (1911). But it was not until late 1940s that the optimization science developed into the independent branch of applied mathematics. By that time the progress of the economy, industry and warfare advanced so far that the human society was compelled to formalize the ways of making "best possible" decisions in practical situations of great complexity.

In 1947 the theory of linear programming (LP) emerged, founded by George Dantzig and John von Neumann, and also Leonid Kantorovich who pioneered the subject in 1939 but whose works were not known on the West for almost 2 decades. Very soon the introduction of the computing machinery caused a burst in the advances of LP and its applications, since many practical problems became numerically solvable. The uses of optimization spread far beyond its initial applications - economy

---

The journal model is Mathematical Programming.

and military logistics.

Linear models are still very popular in practice, even though often applying a linear model requires considerable simplification of the underlying system. However, there are numerous practical problems that cannot be satisfactorily described via linear models, hence making them unlikely applications of LP. As a consequence, nonlinear programming (NLP) and its various branches started developing rapidly since early 1950s, when the famous Karush-Kuhn-Tucker conditions were established.

Fractional programming (FP), a subdomain of which we consider in this dissertation, is a branch of nonlinear optimization that studies the problems in which the objective function involves one or several ratios of functions. Such problems are often called *fractional problems* or *hyperbolic programs* in the literature. This terminology is derived from the seminal paper "Programming with Linear Fractional Functionals" published by A.Charnes and W.W.Cooper in 1962.

Three kinds of fractional programs are mainly considered in the literature: single ratio problems that seek to optimize only one ratio; min-max (max-min) multiple ratio problems, and sum-of-ratio problems. The single ratio models are particularly applicable when optimization of some kind of return-on-allocation ratio is required. They can also be viewed as a type of *multiobjective optimization*, when a compromise is sought between minimization of one function and maximization of another. Examples of single ratio FP applications include financial planning (debt/equity or return/investment ratio), production planning (inventory/sales ratio), fire power distribution on enemy targets, and many others.

Single ratio FP dominated the literature almost exclusively until early 1980s. A lot of important theoretical results were obtained, especially for the case of optimizing a ratio of a concave function and a convex function. It is interesting, however, that one of the earliest publications on FP, though not under this name, is the classical

1937 paper on a model of a general economic equilibrium by John von Neumann that analyzes a multi-ratio fractional program. Still, multi-ratio fractional models seem to be underappreciated in the literature. Among those the sum-of-ratios programs are the least researched, despite of the fact that they have a number of practical applications as well. These problems typically arise in decision making when a balance is sought between several weighted ratios that are to be optimized simultaneously. Examples of such situations include bond portfolio optimization as formulated by Konno and Inori [22]; layered matufacturing [25, 24], for instance material layout problems and cloth manufacturing [3]. A sum-of-ratios model is used in hospital administration in the State of Texas to distribute relative charge increases between medical procedures in various departments [37]. Another important application is data mining [20, 6]: if the objective of the data analysis is to minimize the sum of the average squared distances between the entities within the groups, then the problem becomes a minimum sum-of-ratios problem. Furthermore, when the data is represented as a graph this problem falls into the category of combinatorial fractional programs, which form yet another subdomain of FP that has received very little attention in the literature.

In this dissertation we focus on sum-of-ratios fractional combinatorial optimization problems (FCOPs), a topic that lies in the intersection of two scarcely researched areas of FP: sum-of-ratios FP and combinatorial FP. Although little is known about the properties of continuous sum-of-ratios problems, it is a fact that they are typically hard to solve even when the feasible region is convex. This is due to the fact that the objective function is, in general, multiextremal. Combinatorial problems of this kind are hence expected to be hard to solve as well. Therefore, it is unlikely that an efficient algorithm can be devised for fractional combinatorial problems in general. Yet it is a task of those working in the field of computational optimization to solve

these problems, however difficult they are. The traditional and, perhaps, *the only* way to succeed in such case is to break up a general class of problems into a number of special cases according to some specific properties pertinent to these problems in each of the cases, and exploit these properties to the benefit of the solver. This is the path that we take in this dissertation. We consider a subclass of sum-of-ratios FCOPs such that the linear versions of these problems can be easily solved, i.e., admit polynomial-time exact algorithms. Naturally, even in such restricted domain the problems are numerous. Hence, we concentrate our attention on the generalizations of the three, perhaps, most classical fractional combinatorial problems: *minimum ratio spanning tree (MRST)*, *minimum ratio shortest path (MRSP)*, and *minimum ratio shortest cycle (MRSC)*. The sum-of-ratios versions of these problems respectively have "*multiple* ratio" instead of "ratio" in their names.

In this work we establish some complexity results for the aforementioned problems and attempt to solve them with both traditional mixed integer programming techniques and global optimization approaches that exploit polynomial-time solvability of their linear versions.

The rest of the dissertation is organized as follows. The next chapter is devoted to the review of the related theoretical background and the work that has been done by the other researchers in the area so far. In Chapter III we establish several complexity results for the considered problems. Chapter IV contains the mixed binary formulations. In Chapter V we develop a global optimization approach for the Minimum Multiple Ratio Spanning Tree Problem based on the existing algorithm for a special case of this problem. An improved version of the algorithm is suggested in Chapter VI. Finally, Chapter VII summarizes our work and outlines possible directions for improvements and further research in this area.

CHAPTER II

BACKGROUND

This chapter provides formal definitions of the problems studied in this dissertation, briefly reviews their applications, and outlines the general ideas behind the algorithms proposed for solving the problems of interest.

A *fractional combinatorial optimization problem* is defined as follows:

$$\min_{\mathbf{x}\in\mathcal{X}} \frac{f(\mathbf{x})}{g(\mathbf{x})}, \tag{2.1}$$

where $\mathcal{X} \subseteq \{0,1\}^p$ is a set of certain combinatorial structures, and f and g are real-valued functions defined on $\mathcal{X}$. In addition, it is common to assume that $g(\mathbf{x}) > 0$ for all $\mathbf{x} \in \mathcal{X}$ [35].

One of the classical fractional combinatorial optimization problems is the *minimum ratio spanning tree* (MRST) problem [7], which is defined as follows. Consider a graph $G = (V, E)$ with the set V of n vertices and the set E of m edges. Given a pair of numbers (a_{ij}, b_{ij}) for each edge $(i, j) \in E$, find a spanning tree τ^*, which solves

$$\min_{\tau\in\mathcal{T}} \frac{\sum_{(i,j)\in\tau} a_{ij}}{\sum_{(i,j)\in\tau} b_{ij}}, \tag{2.2}$$

where $\mathcal{T}$ denotes the set of all spanning trees of G.

The practical applications of this problem include the *minimal cost-reliability ratio spanning tree* problem [8], where the functions in the numerator and the denominator of (2.2) represent the cost and the reliability of the spanning tree $\tau \in \mathcal{T}$, respectively. This problem can be solved in polynomial time using $O(|E|^{5/2} \log\log |V|)$ arithmetic operations [8, 9, 21]. Closely related classes of problems, where $\mathcal{X}$ is a cycle, a path, or a cut in graph G also admit polynomial time solution approaches

[2, 28, 34, 35]. An example of such a problem is the *minimum cost-to-time ratio cycle* problem, also known as the *tramp steamer problem* [2]. A short survey on fractional combinatorial optimization problems and related solution approaches can be found in [35].

Recently, Skiscim and Palocsay [39, 40] have introduced a generalization of the MRST problem, where the objective function is given by the sum of two ratios. The resulting *two ratio minimum spanning tree* (TRMST) problem is defined as follows. Consider a graph $G = (V, E)$ with the set V of n vertices and the set E of m edges. Given a set of 4 real positive numbers $(a_{ij}, b_{ij}, c_{ij}, d_{ij})$ for each edge $(i, j) \in E$, find a spanning tree τ^*, which solves

$$\min_{\tau \in \mathcal{T}} \frac{\sum_{(i,j)\in\tau} a_{ij}}{\sum_{(i,j)\in\tau} b_{ij}} + \frac{\sum_{(i,j)\in\tau} c_{ij}}{\sum_{(i,j)\in\tau} d_{ij}}, \tag{2.3}$$

where $\mathcal{T}$ denotes the set of all spanning trees of G.

A closely related class of combinatorial optimization problems is optimization of the ratio of two linear 0–1 functions:

$$\max_{\mathbf{x}\in\{0,1\}^n} f(\mathbf{x}) = \frac{a_0 + \sum_{i=1}^n a_i x_i}{b_0 + \sum_{i=1}^n b_i x_i}. \tag{2.4}$$

This problem is a special case of (2.1) and is usually referred to as a *single-ratio hyperbolic 0-1 programming problem* or *single-ratio fractional 0–1 programming problem* [5]. In a generalization of this problem one considers the sum of ratios of linear 0–1 functions in the objective:

$$\max_{\mathbf{x}\in\{0,1\}^n} f(\mathbf{x}) = \sum_{j=1}^{k} \frac{a_{j0} + \sum_{i=1}^n a_{ji} x_i}{b_{j0} + \sum_{i=1}^n b_{ji} x_i}, \tag{2.5}$$

This problem is known as the *multiple-ratio hyperbolic (fractional) 0-1 programming* problem [33, 41]. A short survey of the literature dealing with the fractional 0–1

programming problems can be found in [31]. Applications of constrained and unconstrained versions of these problems can be found in service systems design [14], facility location [41], query optimization in data bases and information retrieval [20], data mining [6], etc.

Both the minimum ratio spanning tree problem and the single-ratio hyperbolic 0–1 programming problem are polynomially solvable if the denominator is always positive, but become NP-hard if the denominator can take both positive and negative values [32, 39]. On the other hand, their multiple-ratio versions (2.3) and (2.5) are NP-hard for two ratios, even if all denominators are always positive [33, 39]. Some other complexity aspects of unconstrained single- and multiple-ratio fractional 0–1 programming problems, including complexity of uniqueness, approximability and local search, are addressed in [32, 33].

Generally speaking, multiple-ratio problems arise in case of multiple fractional performance metrics that need to be optimized, e.g., a fleet of cargo ships in the tramp steamer problem. Related discussion can be found in [10, 37] and references therein. Analogously with the definition of the multiple-ratio hyperbolic 0–1 programming problem, the *multiple-ratio fractional combinatorial optimization* (MRFCO) problem can be defined as

$$\min_{\mathbf{x}\in\mathcal{X}} \sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{g_i(\mathbf{x})}, \tag{2.6}$$

where $\mathcal{X} \subseteq \{0,1\}^p$ is a set of certain combinatorial structures, and f_i and g_i, $i = 1,\ldots,k$, are real-valued function defined on $\mathcal{X}$.

Obviously, the TRMST problem mentioned above is a simple example of the MRFCO problem. Then the multiple-ratio version of the MRST problem is formulated as follows. Let $G = (V, E)$ be a graph with the set V of n vertices and the set E of m edges. Given k pairs of real positive numbers (a_{ij}^1, b_{ij}^1), (a_{ij}^2, b_{ij}^2), …, (a_{ij}^k, b_{ij}^k) for

each edge $(i,j) \in E$, the *minimum multiple-ratio spanning tree* (MMRST) problem is to find a spanning tree τ^*, which solves

$$\min_{\tau \in \mathcal{T}} \sum_{r=1}^{k} \frac{\sum_{(i,j)\in\tau} a_{ij}^r}{\sum_{(i,j)\in\tau} b_{ij}^r}, \tag{2.7}$$

where $\mathcal{T}$ denotes the set of all spanning trees of G. Note that, similarly to [40], we assume that all the coefficients in the pairs (a_{ij}^1, b_{ij}^1), (a_{ij}^2, b_{ij}^2), ..., (a_{ij}^k, b_{ij}^k) are positive for each arc $(i,j) \in A$.

We consider two different types of approaches to solving the MMRST problem formulated above: mixed integer programming (MIP) and a global optimization algorithms based on representing the problem in the *image space* pioneered by Falk and Palocsay for general fractional programming [15, 16]. Two MIP formulations for the MMRST problem used in computational experiments are derived via Miller-Tucker-Zemlin (MTZ) subtour elimination constraints [29] and the single commodity flow-based formulation of the minimum spanning tree (MST) problem [23]. In both MIP models we also utilize linearization approaches for multiple-ratio fractional 0–1 programming [42]. The suggested global optimization algorithm has evolved from the ideas behind the work on two-ratio minimum spanning trees by Skiscim and Palocsay [39], who also employed the idea of the image space mentioned above.

The image space of the feasible set $\mathcal{T}$ [16] is obtained via introducing a mapping $M : \mathcal{T} \to \mathbb{R}^k$, such that

$$Y = \left\{ M(x) \equiv \left(\frac{a_1^T x}{b_1^T x}, \frac{a_2^T x}{b_2^T x}, \ldots, \frac{a_k^T x}{b_k^T x} \right)^T : \; x \in \mathcal{T} \right\}. \tag{2.8}$$

The idea of the image space became popular in research related to solving the problems involving the sum of ratios. One reason is that using the image space may significantly reduce the computational burden when $k << n$, which is usually the case in practical applications. This especially applies to our case, since for combinatorial

problems like MST the dimension of the original feasible region is often extremely large. Another reason is that, when translated to $\mathbb{R}^k$, the MMRST problem (2.7) is equivalent to the linear program

$$\begin{aligned} \min \quad & e^T y \\ \text{subject to} \quad & y \in conv(M(\mathcal{T})), \end{aligned} \tag{2.9}$$

where e denotes the corresponding vector of all ones. Unfortunately, neither we have a description of $conv(M(\mathcal{T}))$ nor there exists a systematic way of generating its facets or extreme points. It may be possible, however, to build a sort of an approximation of $conv(Y)$, which would be accurate enough in the neighborhood of an optimal extreme point y^* to guarantee a solution as close to y^* as needed. This is precisely the idea our global optimization algorithms utilize.

We will also consider the following related problems. Consider an acyclic directed graph $G = (N, A)$ with the set N of n nodes and the set A of m arcs, where a pair of real numbers (a_{ij}, b_{ij}) is given for each arc $(i, j) \in A$. Let two nodes $s, t \in N$ be given. The minimum ratio path (MRP) problem is to find a directed path p^* from node s to node t, which solves

$$\min_{p \in \mathcal{P}} \frac{\sum_{(i,j) \in p} a_{ij}}{\sum_{(i,j) \in p} b_{ij}}, \tag{2.10}$$

where $\mathcal{P}$ denotes the set of all directed paths from node s to node t of $G(N, A)$. The minimum multiple-ratio path (MMRP) problem generalizes the MRP problem by considering k pairs of real numbers (a^1_{ij}, b^1_{ij}), (a^2_{ij}, b^2_{ij}), ..., (a^k_{ij}, b^k_{ij}) for each arc $(i, j) \in A$. The objective is then to find a directed path p^* from node s to node t, which solves

$$\min_{p \in \mathcal{P}} \sum_{r=1}^{k} \frac{\sum_{(i,j) \in p} a^r_{ij}}{\sum_{(i,j) \in p} b^r_{ij}}, \tag{2.11}$$

where $\mathcal{P}$ denotes the set of all directed paths from node s to node t of $G(N, A)$. The

minimum cost-to-time ratio cycle (MRC) and minimum multiple cost-to-time ratio cycle (MMRC) problems are defined similarly, by considering cycles in the place of paths.

Table 1 summarizes the list of abbreviations used throughout this dissertation.

Table 1 The abbreviations used

CMRST	constrained minimum ratio spanning tree
DAG	directed acyclic graph
FCOP	fractional combinatorial optimization problem
FP	fractional programming
LP	linear programming
MIP	mixed integer programming
MMRST	minimum multiple-ratio spanning tree
MMRC	minimum multiple cost-to-time ratio cycle
MRC	minimum cost-to-time ratio cycle
MRFCO	multiple-ratio fractional combinatorial optimization
MRP	minimum ratio path
MRSC	minimum ratio shortest cycle
MRSP	minimum ratio shortest path
MRST	minimum ratio spanning tree
MST	minimum spanning tree
MTZ	Miller-Tucker-Zemlin
NLP	nonlinear programming
TRMST	two ratio minimum spanning tree

CHAPTER III

COMPLEXITY RESULTS

In order to be able to propose effective solution approaches for an optimization problem, it is essential to understand the problem's complexity [17]. This chapter discusses the known complexity results for some of the problems of interest and establishes the computational complexity for other related problems that have not been studied in the literature. The reader is referred to the classical text by Garey and Johnson [17] for introduction to computational complexity and the theory of NP-completeness.

It is well-known (see, e.g., [20]) that there exists a polynomial time algorithm for solving an single-ratio hyperbolic 0–1 programming problem (2.4), if the following condition holds:

$$b_0 + \sum_{i=1}^{n} b_i x_i > 0, \quad \forall x \in \{0,1\}^n. \tag{3.1}$$

Note that if the term $b_0 + \sum_{i=1}^{n} b_i x_i$ can take the value zero, then problem (2.4) may not have a finite optimum. If

$$b_0 + \sum_{i=1}^{n} b_i x_i \neq 0, \quad x \in \{0,1\}^n \tag{3.2}$$

holds, but the term $b_0 + \sum_{i=1}^{n} b_i x_i$ can take both negative and positive values, the single-ratio problem (2.4) is known to be NP-hard [20]. In other words, the sign of the denominator is "the borderline between polynomial and NP-hard cases" of the single-ratio problem (2.4) [20]. As for the multiple-ratio problem (2.5), the 2-ratio case of (2.5) becomes NP-hard, even if all denominators are restricted to be positive [33].

Similar results can be established for the spanning tree problems [40]. The MRST problem is NP-hard if we allow the denominator to take both positive and negative

values, and the TRMST problem is NP-hard even with both numerator and denominator restricted to be positive.

Next we show that these results hold for the minimum cost-to-time ratio cycle and minimum multiple cost-to-time ratio cycle problems, as well as for the minimum ratio path and minimum multiple-ratio path problems.

Recall the classical SUBSET SUM problem: Given a set of positive integers $S = \{s_1, \ldots, s_n\}$ and a positive integer K, does there exist a vector $x \in \{0,1\}^n$, such that

$$\sum_{i=1}^{n} s_i x_i = K? \tag{3.3}$$

This problem is known to be NP-complete [17]. We will use a polynomial-time reduction from SUBSET SUM to establish the NP-hardness of the MRP problem.

Proposition 1. *The MRP problem is NP-hard if we allow b_{ij} take both positive and negative values for all $(i,j) \in A$.*

Proof. The polynomial-time reduction we use in this and the next propositions follows the ideas used to prove the same result for the MRST problem [40] and single-ratio hyperbolic 0–1 programming problem [5]. Let an instance of the SUBSET SUM problem be given, i.e., we have a set of positive integers $S = \{s_1, \ldots, s_n\}$ and a positive integer K. We construct a directed graph $G = (N, A)$ with $|N| = 3n + 1$ nodes and $|A| = 4n$ arcs, where

$$N = \{v_0, v_1, \ldots, v_n\} \bigcup \{t_1, \ldots, t_n\} \bigcup \{\tilde{t}_1, \ldots, \tilde{t}_n, \}$$

and

$$A = \{(v_0, t_1), (v_1, t_2), \ldots, (v_{n-1}, t_n)\} \bigcup \{(v_0, \tilde{t}_1), (v_1, \tilde{t}_2), \ldots, (v_{n-1}, \tilde{t}_n)\} \bigcup$$

$$\bigcup \{(t_1, v_1), (t_2, v_2), \ldots, (t_n, v_n)\} \bigcup \{(\tilde{t}_1, v_1), (\tilde{t}_2, v_2), \ldots, (\tilde{t}_n, v_n)\}.$$

Let

$$\begin{aligned}
&a_{ij} = \tfrac{1}{2n} && \text{for all } (i,j) \in A,\\
&b_{ij} = -\tfrac{1+2K}{2n} && \text{for arcs } (v_0,t_1),\ldots,(v_i,t_{i+1}),\ldots,(v_{n-1},t_n),\\
&b_{ij} = -\tfrac{1+2K}{2n} && \text{for arcs } (t_1,v_1),\ldots,(t_i,v_i),\ldots,(t_n,v_n),\\
&b_{ij} = -\tfrac{1+2K}{2n} + s_i && \text{for arcs } (v_0,\tilde{t}_1),\ldots,(v_i,\tilde{t}_{i+1}),\ldots,(v_{n-1},\tilde{t}_n),\\
&b_{ij} = -\tfrac{1+2K}{2n} + s_i && \text{for arcs } (\tilde{t}_1,v_1),\ldots,(\tilde{t}_i,v_i),\ldots,(\tilde{t}_n,v_n).
\end{aligned} \tag{3.4}$$

Consider a set of directed paths $\mathcal{P}$ with v_0 and v_n as starting and end nodes respectively. Any directed path from v_0 to v_n goes through nodes $v_{i-1} - t_i - v_i$ or through nodes $v_{i-1} - \tilde{t}_i - v_i$ for all $i = 1,\ldots,n$. Define a 0–1 variable x_i, which is equal to 1 if the directed path goes through $v_{i-1} - \tilde{t}_i - v_i$ and 0, otherwise, i.e., if it goes through $v_{i-1} - t_i - v_i$. Then any directed path p from v_0 to v_n is determined by a vector $x = (x_1,\ldots,x_n) \in \{0,1\}^n$ and (2.10) can be rewritten as follows:

$$\min_{x\in\{0,1\}^n} \frac{1}{2(\sum_{i=1}^n s_i x_i - K) - 1} \tag{3.5}$$

This is the same 0–1 programming problem used in [5] to prove that (2.4) is NP-hard. It is easy to observe that the optimal objective function value of (3.5) is equal to -1 if and only if we have a "yes" instance of the SUBSET SUM problem. □

Using the same idea and results from [32] we can prove that if b_{ij} take both positive and negative values for all $(i,j) \in A$ then

(i) it is NP-hard to check if the solution of the MRP problem, i.e., the optimal directed path, is unique or not (see Lemma 1 of [32]),

(ii) the MRP problem remains NP-hard even the global solution is unique (see Lemma 4 of [32]).

Proposition 2. *The MMRP problem is NP-hard for any $k \geq 2$ even if the values of a_{ij}^r and b_{ij}^r are positive for all $(i,j) \in A$ and $r = 1, \ldots, k$.*

Proof. It is enough to prove this result for the case $k = 2$. In this case the objective function (2.11) is given as

$$\min_{p \in \mathcal{P}} \frac{\sum_{(i,j)\in p} a_{ij}}{\sum_{(i,j)\in p} b_{ij}} + \frac{\sum_{(i,j)\in p} c_{ij}}{\sum_{(i,j)\in p} d_{ij}}. \tag{3.6}$$

For a given instance of the SUBSET SUM problem, we construct the same same graph $G = (N, A)$ as in the previous proof. For each arc $(i,j) \in A$, the values of a_{ij}, b_{ij}, c_{ij} and d_{ij} should be assigned as follows:

$$\begin{array}{ll}
a_{ij} = \frac{1}{2n} & \text{for arcs } (v_0, t_1), \ldots, (v_i, t_{i+1}), \ldots, (v_{n-1}, t_n), \\
a_{ij} = \frac{1}{2n} & \text{for arcs } (t_1, v_1), \ldots, (t_i, v_i), \ldots, (t_n, v_n), \\
a_{ij} = \frac{1}{2n} + \frac{s_i}{2} & \text{for arcs } (v_0, \tilde{t}_1), \ldots, (v_i, \tilde{t}_{i+1}), \ldots, (v_{n-1}, \tilde{t}_n), \\
a_{ij} = \frac{1}{2n} + \frac{s_i}{2} & \text{for arcs } (\tilde{t}_1, v_1), \ldots, (\tilde{t}_i, v_i), \ldots, (\tilde{t}_n, v_n), \\
b_{ij} = \frac{1}{2n} & \text{for all } (i,j) \in A, \\
c_{ij} = \frac{(K+1)^2}{2n} & \text{for all } (i,j) \in A, \\
d_{ij} = \frac{1}{2n} & \text{for arcs } (v_0, t_1), \ldots, (v_i, t_{i+1}), \ldots, (v_{n-1}, t_n), \\
d_{ij} = \frac{1}{2n} & \text{for arcs } (t_1, v_1), \ldots, (t_i, v_i), \ldots, (t_n, v_n), \\
d_{ij} = \frac{1}{2n} + \frac{s_i}{2} & \text{for arcs } (v_0, \tilde{t}_1), \ldots, (v_i, \tilde{t}_{i+1}), \ldots, (v_{n-1}, \tilde{t}_n), \\
d_{ij} = \frac{1}{2n} + \frac{s_i}{2} & \text{for arcs } (\tilde{t}_1, v_1), \ldots, (\tilde{t}_i, v_i), \ldots, (\tilde{t}_n, v_n).
\end{array} \tag{3.7}$$

Then any directed path p from v_0 to v_n is determined by a vector $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ and (3.6) can be rewritten in terms of variables x_i as follows:

$$\min_{x \in \{0,1\}^n} 1 + \sum_{i=1}^{n} s_i x_i + \frac{(1+K)^2}{1 + \sum_{i=1}^{n} s_i x_i} \tag{3.8}$$

Performing some simple manipulations with the objective function in (3.8) we obtain

$$
\begin{aligned}
1+\sum_{i=1}^{n} s_i x_i + \frac{(1+K)^2}{1+\sum_{i=1}^{n} s_i x_i} &= \frac{1+2\sum_{i=1}^{n} s_i x_i + (\sum_{i=1}^{n} s_i x_i)^2 + K^2 + 2K + 1}{1+\sum_{i=1}^{n} s_i x_i} \\
&= \frac{(\sum_{i=1}^{n} s_i x_i)^2 + 2\sum_{i=1}^{n} s_i x_i + 2 + 2K + K^2}{1+\sum_{i=1}^{n} s_i x_i} \\
&= \frac{(\sum_{i=1}^{n} s_i x_i - K)^2 + (2K+2)(1+\sum_{i=1}^{n} s_i x_i)}{1+\sum_{i=1}^{n} s_i x_i} \\
&= \frac{(\sum_{i=1}^{n} s_i x_i - K)^2}{1+\sum_{i=1}^{n} s_i x_i} + 2(K+1),
\end{aligned} \tag{3.9}
$$

which implies that solution of (3.8) is equal to $2(K+1)$ if and only if the SUBSET SUM problem has a solution. □

These results can be also extended for the case of the minimum cost-to-time ratio cycle and minimum multiple cost-to-time ratio cycle problems. Indeed, we can modify the graph $G = (N, A)$ from Propositions 1 and 2 by adding one more node $\tilde{v}$ with arcs $(v_n, \tilde{v})$, $(\tilde{v}, v_0)$ and changing the values of a_{ij}, b_{ij}, c_{ij} and d_{ij} correspondingly, making sure that the discussed reductions to hyperbolic 0–1 programming problem (3.5) and (3.8) remain valid. We obtain the following statements.

Proposition 3. *The MRC problem is NP-hard if we allow b_{ij} take both positive and negative values for all $(i, j) \in A$.*

Proposition 4. *The MMRC problem is NP-hard for any $k \geq 2$ even if the values of a_{ij}^r and b_{ij}^r are positive for all $(i, j) \in A$ and $r = 1, \ldots, k$.*

CHAPTER IV

LINEAR MIXED 0–1 PROGRAMMING FORMULATIONS

A mathematical programming formulation often gives an important insight into the structure of the considered problem. With the advent of powerful MIP solver engines, such as CPLEX, Xpress, GLPK, etc., such formulations gain even more importance. Indeed, in cases when the software can deliver a solution in reasonable time based on the mathematical description of a problem, the need for developing problem-specific solution techniques is questionable.

In this chapter we develop mixed 0–1 formulations for the MMRST and MMRP problems and perform experiments to test their efficiency from the computational point of view. The first section is allotted to the MMRST problem. There, we develop an MST formulation based on the directed-out spanning tree via Miller-Tucker-Zemlin constraints, suggest several valid inequalities, and transform it into the MMRST formulation using the results of Wu [42]. We use this transformation throughout the chapter to linearize our sum-of-ratios problems, and therefore we elaborate on the relevant details when we use it for the first time. Also, in the first section we provide another, flow-based, MMRST formulation. The second section contains the MMRP formulation that is derived from the classic network flow model for the Shortest (s, t)-path problem. The chapter is concluded by a section where we present and discuss the numerical results.

IV.1. Formulations for the MMRST problem

IV.1.1. Formulation via Miller-Tucker-Zemlin constraints

Consider an instance of the MMRST problem, which is given by a simple undirected graph $G = (V, E)$ with the set V of n vertices and the set E of m edges, where each edge $(i, j) \in E$ is associated with k pairs of numbers $(a_{ij}^r, b_{ij}^r), r = 1, \ldots, k$. Then the directed version of G is defined as a network $\tilde{G} = (N, A)$, where the set of nodes N coincides with V, the set of arcs A has two directed arcs (i, j) and (j, i) for every undirected edge (i, j) in E, and each arc is associated with the same k pairs of numbers as the corresponding edge in E, i.e., $a_{ij}^r = a_{ji}^r$ and $b_{ij}^r = b_{ji}^r$ for all $r = 1, \ldots, k$. Therefore, we obtained a directed graph $\tilde{G} = (N, A)$ with $|N| = n$ and $|A| = 2m$.

The formulation is designed based on the observation that any feasible solution to MMRST problem corresponds to a *directed-out spanning tree* rooted at node 1 (i.e., there is a unique directed path in the resulting tree from node 1 to every other node of the graph), and vice versa. Let x_{ij}, $i \neq j$, be 0–1 variables such that $x_{ij} = 1$ if an arc (i, j) appears in the optimal tree sought, and $x_{ij} = 0$, otherwise. The total number of variables x_{ij} is equal to $n(n-1)$, however, if $(i, j) \notin A$ we can assign $x_{ij} = 0$, with only $|A| = 2m$ variables remaining. Then the objective function for the MMRST problem can be formulated as follows:

$$\min_{x_{ij} \in \{0,1\}} \sum_{r=1}^{k} \frac{\sum_{(i,j) \in A} a_{ij}^r x_{ij}}{\sum_{(i,j) \in A} b_{ij}^r x_{ij}}. \tag{4.1}$$

Next we need to find a mathematical description of the set of all directed-out spanning trees rooted at node 1 in terms of the introduced decision variables. This can be done using the following set of linear constraints. Since the tree is rooted at node 1, we

have

$$\sum_{(1,j)\in A} x_{1j} \geq 1. \tag{4.2}$$

The tree is directed out, therefore the in-degree of each of the nodes $2, \ldots, n$ is exactly 1, and the in-degree of node 1 is 0:

$$\sum_{i:(i,j)\in A} x_{ij} = 1, \quad j = 2, \ldots n, \tag{4.3}$$

$$x_{(i,1)} = 0, \quad \forall i : (i,1) \in A. \tag{4.4}$$

The total number of arcs in the spanning tree is $n-1$:

$$\sum_{(i,j)\in A} x_{ij} = n - 1. \tag{4.5}$$

To ensure the connectivity, we will use the classical *Miller-Tucker-Zemlin (MTZ)* constraints that were originally proposed in [29] in order to prevent subtours in the *traveling salesman problem (TSP)*:

$$u_i - u_j + (n-1)x_{ij} \leq n - 2, \quad \forall i, j \geq 2,\ i \neq j,\ (i,j) \in A, \tag{4.6}$$

where the variables u_i, $i = 1, \ldots, n$ satisfy

$$u_1 = 0 \ \text{ and } \ 1 \leq u_i \leq n-1,\ i = 2, \ldots, n. \tag{4.7}$$

These constraints (4.6)-(4.7) are usually referred to as *Miller-Tucker-Zemlin (MTZ) constraints*.

Proposition 5. *Constraints (4.2)-(4.7) define a directed-out spanning tree rooted at node* 1.

Proof. Observe that since we require condition (4.3)-(4.4) to be satisfied then we may have only directed cycles and do not have cycles, which involve node 1. MTZ con-

straints (4.6)-(4.7) prevent any other directed cycle in the final solution [29]. Therefore, we need to show only the existence of the solution. Let the values x_{ij} correspond to some directed-out spanning tree rooted at node 1. If $x_{ij} = 0$ then

$$u_i - u_j \leq n - 2, \tag{4.8}$$

which is obviously true due to (4.7). If $x_{ij} = 1$ then

$$u_i - u_j \leq -1,$$

that is

$$u_j \geq u_i + 1. \tag{4.9}$$

Condition (4.9) implies that for any node j of the subtree rooted at node i the value of corresponding u_j should be greater than u_i. It is easy to notice that if we numerate the nodes of the directed-out spanning tree rooted at node 1 according to the order we visit the nodes of the tree in *depth-first*, or *breadth-first* search procedure than constraints (4.9) will be satisfied for all arcs (i, j) in the tree. □

Remark. As we mentioned above the values of u_i imply that for any node j of the subtree rooted at node i the value of corresponding u_j should be greater than u_i. Therefore, indices u_i define some *topological ordering* of the final tree. This fact is not surprising since we know that a graph has a topological ordering if and only if it is acyclic [2]. Therefore, we can conclude that

Corollary 1. *Subgraph induced by nodes* 1, 2, ..., n *and the constraints (4.6)-(4.7) is topologically ordered, where the values of* u_1, u_2, ..., u_n *define the order labels of the nodes.*

Next, following a standard procedure for multiple-ratio fractional 0–1 programming introduced in [42] (see also [32, 41] for some additional discussion), we define k

new variables y_r as follows:

$$y_r = \frac{1}{\sum_{(i,j)\in A} b_{ij}^r x_{ij}}, \quad r = 1, \ldots, k, \tag{4.10}$$

which, assuming that the denominator in (4.10) cannot be equal to zero, is equivalent to

$$\sum_{(i,j)\in A} b_{ij}^r x_{ij} y_r = 1, \quad r = 1, \ldots, k. \tag{4.11}$$

We define $2km$ new variables z_{ij}^r such that

$$z_{ij}^r = x_{ij} y_r. \tag{4.12}$$

The nonlinear constraints (4.12) can be equivalently replaced by 4 linear inequalities

$$\begin{aligned} z_{ij}^r &\le U^r x_{ij}, \quad z_{ij}^r \le y_r - L^r(1 - x_{ij}), \\ z_{ij}^r &\ge L^r x_{ij}, \quad z_{ij}^r \ge y_r - U^r(1 - x_{ij}), \end{aligned} \tag{4.13}$$

where U^r and L^r are upper and lower bounds on variable y_r, respectively. These bounds can be defined, for example, as

$$U^r \ge \max_{\tau\in\mathcal{T}} \frac{1}{\sum_{(i,j)\in\tau} b_{ij}^r x_{ij}} \quad \text{and} \quad L^r \le \min_{\tau\in\mathcal{T}} \frac{1}{\sum_{(i,j)\in\tau} b_{ij}^r x_{ij}}. \tag{4.14}$$

If $b_{ij}^r \ge 0$ for $(i,j) \in A$ and $r = 1, \ldots, k$, then these bounds can be defined as follows

$$U^r \ge \frac{1}{\sum_{(i,j)\in\tau_{min}} b_{ij}^r} \quad \text{and} \quad L^r \le \frac{1}{\sum_{(i,j)\in\tau_{max}} b_{ij}^r}, \tag{4.15}$$

where τ_{min} and τ_{max} are minimum and maximum, respectively, weight spanning trees of the initial undirected graph G with weights b_{ij}^r assigned to its edges.

The resulting linear mixed 0–1 formulation of the MMRST problem is given as:

$$\min \sum_{r=1}^{k} \sum_{(i,j)\in A} a_{ij}^r z_{ij}^r. \tag{4.16a}$$

subject to

$$\sum_{i:(i,j)\in A} x_{ij} = 1, \quad j = 2, \dots n; \tag{4.16b}$$

$$\sum_{j:(1,j)\in A} x_{1j} \geq 1 \quad \text{and} \quad x_{i1} = 0, \quad \forall i : (i,1) \in A; \tag{4.16c}$$

$$\sum_{(i,j)\in A} x_{ij} = n - 1; \tag{4.16d}$$

$$u_i - u_j + (n-1)x_{ij} \leq n - 2, \quad \forall i, j \geq 2,\ (i,j) \in A; \tag{4.16e}$$

$$\sum_{(i,j)\in A} b_{ij}^r z_{ij}^r = 1, \quad r = 1, \dots, k; \tag{4.16f}$$

$$z_{ij}^r \leq U^r x_{ij},\ z_{ij}^r \leq y_r - L^r(1 - x_{ij}), \quad \forall (i,j) \in A,\ r = 1, \dots, k; \tag{4.16g}$$

$$z_{ij}^r \geq L^r x_{ij},\ z_{ij}^r \geq y_r - U^r(1 - x_{ij}), \quad \forall (i,j) \in A,\ r = 1, \dots, k; \tag{4.16h}$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A; \tag{4.16i}$$

$$u_1 = 0 \ \text{ and } \ 1 \leq u_i \leq n-1, \quad i = 2, \dots, n, \quad L^r \leq y_r \leq U^r, \quad r = 1, \dots, k; \tag{4.16j}$$

$$z_{ij}^r \geq 0, \quad \forall (i,j) \in A,\ r = 1, \dots, k; \tag{4.16k}$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A. \tag{4.16l}$$

The bounds U^r and L^r are defined by (4.14) (or by (4.15) in case of positive b_{ij}^r). The total number of variables is $O(mk+n)$, out of which $2m$ are binary, and the number of constraints is $O(mk+n)$.

IV.1.1.1. Valid inequalities

In [11] the following valid inequalities were developed for the improvement of MTZ constraints in the **TSP**:

$$u_i - u_j + (n-1)x_{ij} + (n-3)x_{ji} \leq n - 2, \quad i \neq j,\ i, j = 2, \dots, n, \tag{4.17}$$

$$u_i \geq 1 - (n-3)x_{i1} + \sum_{j=2, j\neq i}^{n} x_{ji}, \quad i = 2, \ldots, n, \tag{4.18}$$

$$u_i \leq n - 1 - (n-3)x_{1i} - \sum_{j=2, j\neq i}^{n} x_{ij}, \quad i = 2, \ldots, n, \tag{4.19}$$

Constraints (4.17), (4.18)-(4.19) are lifted versions of (4.6) and (4.7), respectively.

Proposition 6. *Constraint (4.17) is a valid inequality for the directed-out spanning tree formulation and the value of u_j ($j = 1, \ldots, n$) defines the depth of node j in the final tree, i.e., the length of the path from the root node 1 to the node j.*

Proof. The proof just follows the arguments in [11]. It is obviously true for $x_{ji} = 0$. In case of $x_{ji} = 1$ constraints (4.17) force $u_i = u_j + 1$, which is true if and only if we numerate the nodes according to their distance from the root node 1.

□

In our case, constraints (4.18)-(4.19) should be replaced by

$$u_i \geq 2 - x_{1i}, \quad i \geq 2, (1, i) \in A, \tag{4.20}$$

$$u_i \leq n - 1 - \sum_{(i,j)\in A, j\neq 1} x_{ij}, \quad i \geq 2, \ldots, n. \tag{4.21}$$

Proposition 7. *Constraints (4.20)-(4.21) are valid inequalities for the directed-out spanning tree formulation.*

Proof. The proof just follows the arguments from the proof above taking into account that the considered graph structure a directed-out spanning tree. □

IV.1.2. Flow-based formulation

We consider a variation of the flow-based linear mixed 0–1 formulation for the minimum spanning tree problem proposed in [23]. Assume that node 1 serves as a source

node that sends a unit of some flow to every other node. We will denote by f_{ij} the variable representing the flow on edge (i, j) in the direction i to j. For each edge $(i, j), i < j$, we define a variable x_{ij} that indicates whether (i, j) is a part of the tree sought. Then we have:

$$\min \sum_{(i,j)\in E, i<j} w_{ij} x_{ij}. \tag{4.22a}$$

subject to

$$\sum_{i:(1,i)\in E} (f_{1i} - f_{i1}) = n - 1; \tag{4.22b}$$

$$\sum_{i:(v,i)\in E} (f_{vi} - f_{iv}) = 1, \quad \forall v \in V, v \neq 1; \tag{4.22c}$$

$$f_{ij} \leq (n-1)x_{ij}, \quad f_{ji} \leq (n-1)x_{ij}, \quad \forall (i,j) \in E, i < j; \tag{4.22d}$$

$$\sum_{(i,j)\in E, i<j} x_{ij} = n - 1; \tag{4.22e}$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i,j) \in E, i < j; \tag{4.22f}$$

$$f_{ij} \geq 0 \ \forall (i,j) \in E. \tag{4.22g}$$

Following (4.16) and the respective discussion in the previous subsection, formulation (4.22) can be easily modified to model the MMRST problem. As a result, we obtain the following formulation:

$$\min \sum_{r=1}^{k} \sum_{(i,j)\in E, i<j} a_{ij}^r z_{ij}^r. \tag{4.23a}$$

subject to

$$\sum_{i:(i,1)\in E} (f_{1i} - f_{i1}) = n - 1; \tag{4.23b}$$

$$\sum_{i:(v,i)\in E} (f_{vi} - f_{iv}) = 1, \quad \forall v \in V, v \neq 1; \tag{4.23c}$$

$$f_{ij} \leq (n-1)x_{ij}, \quad f_{ji} \leq (n-1)x_{ij}, \quad \forall (i,j) \in E, \ i < j; \tag{4.23d}$$

$$\sum_{(i,j)\in E, i<j} x_{ij} = n - 1; \tag{4.23e}$$

$$\sum_{(i,j)\in E, i<j} b_{ij}^r z_{ij}^r = 1, \quad r = 1, \ldots, k; \tag{4.23f}$$

$$z_{ij}^r \leq U^r x_{ij}, \quad z_{ij}^r \leq y_r - L^r(1 - x_{ij}), \quad \forall (i,j) \in E,\ i < j,\ r = 1, \ldots, k; \tag{4.23g}$$

$$z_{ij}^r \geq L^r x_{ij}, \quad z_{ij}^r \geq y_r - U^r(1 - x_{ij}), \quad \forall (i,j) \in E,\ i < j,\ r = 1, \ldots, k; \tag{4.23h}$$

$$L^r \leq y_r \leq U^r, \quad r = 1, \ldots, k; \tag{4.23i}$$

$$z_{ij}^r \geq 0, \quad \forall (i,j) \in E,\ i < j,\ r = 1, \ldots, k; \tag{4.23j}$$

$$f_{ij} \geq 0, \quad \forall (i,j) \in E; \tag{4.23k}$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i,j) \in E,\ i < j. \tag{4.23l}$$

The flow-based formulation (4.23) contains $O(mk + n)$ variables and $O(mk + n)$ constraints as well as the formulation (4.16). However, the number of binary variables is m instead of $2m$ in (4.16), since only one binary variable per edge is introduced.

IV.2. Formulation for the MMRP problem

Consider a directed graph $G = (N, A)$ with $|N| = n$ nodes and $|A|$ arcs, and weights $w_{ij} \in \mathbb{R}_+$ associated with each arc (i, j). Several classical mathematical programming formulations for the Shortest (s, t)-path problem are well-known in the literature (e.g., see [2]). One of the most popular is the following network-flow-like formulation, where the arcs (i, j) are associated with the binary variables x_{ij}.

$$\min \sum_{(i,j)\in A} w_{ij} x_{ij} \tag{4.24a}$$

subject to

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ij} = 0, \quad \forall i \neq s, t,\ i \in N; \tag{4.24b}$$

$$\sum_{i:(i,s)\in A} x_{is} - \sum_{i:(s,i)\in A} x_{si} = -1; \tag{4.24c}$$

$$\sum_{i:(i,t)\in A} x_{it} - \sum_{i:(t,i)\in A} x_{ti} = 1; \tag{4.24d}$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A. \tag{4.24e}$$

To transform (4.24) model into a mixed 0–1 formulation for the MMRP problem, we again resort to the transformation by Wu, that we used for the MMRST problem. Assuming that k pairs of real numbers (a_{ij}^r, b_{ij}^r), $r = 1, \ldots, k$ are associated with each arc $(i,j) \in A$, and that none of the denominators may be 0, we define k new variables y_r and km new variables z_{ij}^r in the same way as in (4.10) and (4.12), respectively. After adding the constraints (4.13) along with the bounds (4.14) on y^r we arrive at the final formulation:

$$\min \sum_{r=1}^{k} \sum_{(i,j)\in A} a_{ij}^r z_{ij}^r \tag{4.25a}$$

subject to

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ij} = 0, \quad \forall i \neq s,t,\ i \in N; \tag{4.25b}$$

$$\sum_{i:(i,s)\in A} x_{is} - \sum_{i:(s,i)\in A} x_{si} = -1; \tag{4.25c}$$

$$\sum_{i:(i,t)\in A} x_{it} - \sum_{i:(t,i)\in A} x_{ti} = 1; \tag{4.25d}$$

$$\sum_{(i,j)\in A} b_{ij}^r z_{ij}^r = 1, \quad r = 1, \ldots, k; \tag{4.25e}$$

$$z_{ij}^r \leq U^r x_{ij}, \quad z_{ij}^r \leq y_r - L^r(1 - x_{ij}), \quad \forall (i,j) \in A, r = 1, \ldots, k; \tag{4.25f}$$

$$z_{ij}^r \geq L^r x_{ij}, \quad z_{ij}^r \geq y_r - U^r(1 - x_{ij}), \quad \forall (i,j) \in A, r = 1, \ldots, k; \tag{4.25g}$$

$$L^r \leq y_r \leq U^r, \quad r = 1, \ldots, k; \tag{4.25h}$$

$$z_{ij}^r \geq 0, \quad \forall (i,j) \in A,\ i < j,\ r = 1, \ldots, k; \tag{4.25i}$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A. \tag{4.25j}$$

This formulation contains $O(mk)$ variables and $O(mk+n)$ constraints. This is not surprising, since the transformation that we use always spawns $k(m+1)$ new continuous variables, whenever we have m binary variables involved in the fractional objective function.

IV.3. Computational results

The computational experiments were carried out for $k = 1, \ldots, 5$. For all graphs the parameters $a_1^e, \ldots, a_k^e$, $b_1^e, \ldots, b_k^e$ for each edge e of a graph are uncorrelated and follow standard uniform distribution. We rely on the Mersenne Twister MT19937 [26] random number generator implementation from the Boost random number library [27] to create random variates for our instances. For MMRST problem we consider two types of test instances: complete graphs and connected random graphs.

We use CPLEX 9.13 solver with default settings to test performance of the MIP models (4.16) and (4.23). The experiments were performed on a computer with Intel® Core™ 2 Duo 3.16 GHz CPU and 3.23 GB of RAM.

Tables 2 and 3 summarize the computational results for MMRST instances with complete and sparse connected random graphs, respectively. Probability of an edge in the latter type of instances is set to 0.1 when $|V| = 20$, and 0.05 otherwise. We tested a batch of 3 instances for each reported pair (k, n). For each approach the target gap value is set to 1%, and computation time is limited to 1 hour. When the average gap is not available, it means that no feasible solution was found for *any* instance in the batch by the method within the allotted time. If the solver was unable to find a feasible solution for *some* instances in the batch, the gap value is marked with (*).

Table 2 Performance of MMRST MIP models on complete graph instances

k	n	MTZ 1		MTZ 2		FLOW	
		run time (sec.)	gap (%)	run time (sec.)	gap (%)	run time (sec.)	gap (%)
2	10	1.7	0.9	1.9	0.9	0.6	1.0
	15	27.0	1.0	115.7	1.0	128.0	1.0
	20	2468.0	5.4	2713.0	5.2	1337.0	2.6
	30	3600.0	22.3	3600.0	20.9	3600.0	17.2
	40	3600.0	25.5	3600.0	30.7	3600.0	33.8
	50	3600.0	34.2	3600.0	32.4	3600.0	29.9*
	80	3600.0	50.5	3600.0	43.7	3600.0	n/a
	100	3600.0	n/a	3600.0	45.7	3600.0	n/a
3	10	2.4	1.0	2.7	1.0	0.6	0.9
	15	1603.9	2.9	1528.7	2.9	504.6	1.0
	20	3600.0	15.5	3600.0	13.3	3600.0	8.0
	30	3600.0	29.5	3600.0	29.5	3600.0	25.0
	40	3600.0	35.6	3600.0	32.4	3600.0	29.9
	50	3600.0	41.8	3600.0	37.5	3600.0	n/a
4	10	15.3	1.0	14.3	1.0	5.4	1.0
	15	3600.0	9.7	3600.0	9.9	3002.0	5.2
	20	3600.0	23.3	3600.0	23.0	3600.0	19.3
	30	3600.0	27.1	3600.0	30.5	3600.0	31.5
5	10	65.0	1.0	56.7	1.0	17.3	1.0
	15	3600.0	13.5	3600.0	13.3	3600.0	7.8
	20	3600.0	n/a	3600.0	20.1	3600.0	16.3

Table 3 Performance of MMRST MIP models on sparse random graph instances

k	n	MTZ 1		MTZ 2		FLOW	
		run time (sec.)	gap (%)	run time (sec.)	gap (%)	run time (sec.)	gap (%)
2	20	0.1	0.7	0.1	0.6	0.1	0.6
	40	0.4	0.9	0.5	0.8	0.1	0.7
	60	2291.0	2.6	1296.0	1.7	146.1	0.9
	80	3600.0	7.8	3600.0	4.6	2493.7	1.8
	100	3600.0	14.7	3600.0	10.0	3600.0	7.0
	120	3600.0	24.7	3600.0	16.3	3600.0	n/a
	140	3600.0	29.1	3600.0	21.7	3600.0	27.8*
	160	3600.0	28.4	3600.0	23.0	3600.0	n/a
3	20	0.1	0.9	0.2	0.7	0.1	0.6
	40	11.4	0.7	8.9	0.9	0.1	0.7
	60	3600.0	7.5	3600.0	6.3	1806.0	1.4
	80	3600.0	10.9*	3600.0	8.2	3600.0	3.5
	100	3600.0	20.3	3600.0	17.4	3600.0	12.4*
4	20	0.1	0.6	0.1	0.5	0.1	0.4
	40	4.4	0.9	1.58	0.9	0.1	0.6
	60	3600.0	4.0	2806.7	2.6	1002.6	0.9
	80	3600.0	12.8	3600.0	9.36	3600.0	4.6
5	20	0.8	0.6	0.7	0.8	0.1	0.9
	40	24.1	1.0	17.2	1.0	0.3	0.9
	60	3600.0	5.5	3600.0	4.8	1052.7	1.0

The results for the MIP formulations are reported in the columns under 'MTZ 1',

'MTZ 2' and 'FLOW'. The MTZ columns respectively refer to the formulation (4.16) with no added valid inequalities, and (4.16) augmented by the inequalities (4.17). Experiments with added constraints (4.20) and (4.21) were also performed, but only adding (4.17) consistently yields a noticeable benefit in terms of run time, mostly for sparse graphs. In fact, the constraints (4.21) *increase* the run time significantly.

The directed acyclic graphs (DAGs) $G = (N, A)$ that we use for our MMRP instances have the following structure. The undirected version of such a graph is a complete $(l + 1)$-partite graph with subsets $N_1, \ldots, N_{l+1} \subset N$ forming the partition. Each subset N_i has w nodes with the exception of $N_1 = \{s\}$ and $N_{l+1} = \{t\}$. In a directed graph the nodes are connected in a way so that $(v, u) \in A$ if and only if $v \in N_i, u \in N_j$ such that $j > i$. Therefore an (l,w) DAG would have

$$|N| = (l-1)w + 2 \quad \text{and} \quad |A| = \frac{(|N|-2)(|N|-w)}{2} + 2|N| - 3.$$

Such graph structure allows to completely describe it with a pair of positive integers (l, w). The MMRP instances for the computational experiment were chosen to approximately match the graph sizes of the complete MMRST instances with 15, 30, 50, 100 and 160 vertices. Table 4 summarizes the results in a self-explanatory manner.

Performance of the MMRST MIP models is affected by two main factors: the number of ratios in the objective and the number of edges in the graph. The latter factor translates directly into the number of binary variables involved in the considered MIP formulations, and has a strong effect on the performance. In fact, comparison of the data from Tables 2 and 3 suggests that, given the computation resources, both models fail to reach the target gap value in the allotted time frame when the number of binaries approaches 180. This roughly corresponds to a 15-vertex complete or 60-vertex random instance for the MTZ model, that has 2 binary variables per edge. These numbers are respectively 20 and 80 for the flow-based model, that has a single

Table 4 Performance of the MMRP MIP model on tested instances

(l, w)	$k = 2$		$k = 3$		$k = 4$		$k = 5$	
	run time (sec.)	gap (%)	run time (sec.)	gap (%)	run time (sec.)	gap (%)	run time (sec.)	gap (%)
(5,3)	0.0	0.0	0.0	0.0	0.3	0.0	1.0	0.0
(7,5)	10.7	0.1	52.0	0.1	98.7	0.1	247.3	0.1
(9,7)	270.3	0.1	2646.0	5.2	3373.3	33.5	3600.0	52.6
(12,10)	3600.0	68.2	3600.0	79.6	3600.0	81.0	3600.0	81.1
(14,12)	3600.0	86.4	3600.0	84.1	3600.0	85.9	3600.0	85.5

binary per edge, and, as expected, performs slightly better. The flow-based model, however, consistently shows difficulty finding a feasible solution for larger instances.

The same general observations apply to the MMRP model as well. However, it shows somewhat better performance if we solely take into consideration instance graph size. This is probably due to the fact that an MMRP instance of the given structure contains fewer feasible solutions than a complete MMRST instance of similar size.

CHAPTER V

A GLOBAL OPTIMIZATION APPROACH FOR THE MMRST PROBLEM

The optimization algorithm presented in this chapter is, to some extent, a generalization of the approach used by Sciskim and Palocsay in [39] for a Minimum Two Ratio Spanning Tree Problem. Just like the latter algorithm, it is based on the representation of the problem in the *image space* mentioned in Chapter II.

In order to proceed with description of the algorithm, let us introduce some additional notation that we use throughout the rest of the paper. Recall the definition of the image $Y \triangleq M(\mathcal{T})$ of the feasible set $\mathcal{T}$ of the MMRST problem introduced in (2.8), where $M : \mathcal{T} \to \mathbb{R}^k$ is given by

$$M(x) \equiv \left(\frac{a_1^T x}{b_1^T x}, \frac{a_2^T x}{b_2^T x}, \ldots, \frac{a_k^T x}{b_k^T x} \right)^T$$

for any $x \in \mathcal{T}$. Given $x \in \mathcal{T}$, we will denote by $M_r(x)$ the r-th ratio $a_r^T x / b_r^T x$. Given $y \in Y$, we will denote by $M^{-1}(y)$ the inverse image $\{x \in \mathcal{T} : M(x) = y\}$. Note that since $\mathcal{T}$ is finite, Y is also finite. For a rectangular region $Q = \{y \in \mathbb{R}^k : l_r \leq y_r \leq u_r, r = 1, \ldots, k\}$, we denote the vector $l = (l_1, \ldots, l_k)$ by $L(Q)$, and its r-th component by $L_r(Q)$. Similarly, $U(Q)$ and $U_r(Q)$ denote u and u_r, respectively.

On each step j of our algorithm, an approximation of the portion of $conv(Y)$ containing optimal solution y^* is given by a set of rectangular regions $S^j = \{Q_1^j, \ldots, Q_t^j\}$, such that for all steps j and i, where $j < i$, we have

1. $y^* \in \bigcup_{Q \in S^j} Q$;
2. $\bigcup_{Q \in S^i} Q \subseteq \bigcup_{Q \in S^j} Q$;

3. $\bar{y}^j \in \bigcup_{Q \in S^j} Q$ and $\bar{y}^i \in \bigcup_{Q \in S^i} Q$ are available s.t. $e^T y^* \leq e^T \bar{y}^i \leq e^T \bar{y}^j$.

Note that $e^T L(Q_p^j)$ provides a lower bound on the optimal objective of (2.9) over the rectangle Q_p^j. Without loss of generality, we can assume that on every step j the rectangular regions in the set S^j are sorted in the nondecreasing order of such lower bounds, i.e., we have $e^T L(Q_p^j) \leq e^T L(Q_q^j)\ \forall p < q$. Then $e^T L(Q_1^j)$ provides the lower bound on (2.9) available from the approximation S^j. Let us denote this lower bound by $\underline{z}^j$, and the current upper bound, which is the best feasible solution found so far, by $\bar{z}$. S^{j+1} is obtained from S^j by reducing Q_1^j and/or partitioning it into two subregions. The reduction is done similarly to [39], by solving the following subproblem for a particular ratio $r \in \{1, \ldots, k\}$:

$$\min\{y_r : y \in Y \cap Q_1^j, y_s \leq u_s, s = 1, \ldots, k\} \tag{5.1}$$

where

$$u_s = \max\{y_s : y \in Q_1^j, e^T y \leq \bar{z}\} = \bar{z} - \underline{z}^j + L_s(Q_1^j),\ s = 1, \ldots, k.$$

Let $\tilde{y}$ be an optimal solution to (5.1). Then Q_1^j may be reduced to

$$P = \{y \in Q_1^j : y_s \leq u_s, s = 1, \ldots, k, s \neq r, y_r \geq \tilde{y}_r\}$$

without discarding any $y \in Y \cap Q_1^j$ that are no worse than the best incumbent solution to (2.9). If $\tilde{y}_r > L_r(Q_1^j)$, then $\underline{z}^{j+1}$ will be a better lower bound than $\underline{z}^j$. Certainly, P is discarded from further consideration if $e^T L(P) \geq \bar{z}$. Otherwise, $\tilde{y}$ may improve on the current incumbent solution. If $\tilde{y}_r = L_r(Q_1^j)$, then P is partitioned into $P' = \{y \in P : y_h \leq (L_h(P) + \tilde{y}_h)/2\}$ and $P'' = \{y \in P : y_h \geq (L_h(P) + \tilde{y}_h)/2\}$, where

$$h = \arg\max\{|\tilde{y}_s - L_s(P)| : s = 1, \ldots, k\}. \tag{5.2}$$

Thus $\tilde{y}$ becomes separated from $L(P')$, making the next iteration likely to improve $\underline{z}$. Of course, (5.1) does not have to be solved when $\tilde{y} \in P$ such that $\tilde{y}_r = L_r(Q_1^j)$ is already known from previous steps of the algorithm.

The formal description of the algorithm is provided in Algorithm 1. Note that $\mathcal{T}$ is passed to the main procedure implicitly through the description of graph G. We discuss how the subproblems (5.1) are solved along with some other important details in the following subsections. Figure 1 illustrates the steps of Algorithm 1 in detail on a small numerical example with two ratios.

Theorem 1. *Algorithm 1 converges in a finite number of steps.*

Proof. Let $\bar{z}^j$ denote the value of $\bar{z}$ after j steps of the algorithm. The algorithm terminates when $\bar{z}^j - \underline{z}^j \leq \epsilon \bar{z}$. Suppose that the stopping criterion is not satisfied in a finite number of steps, i.e., the algorithm generates infinite sequences of bounds $\{\underline{z}^j : j \geq 1\}$ and $\{\bar{z}^j : j \geq 1\}$. Since $\{\underline{z}^j : j \geq 1\}$ is monotonously nondecreasing, $\{\bar{z}^j : j \geq 1\}$ is monotonously nonincreasing, and $\underline{z}^j \leq \bar{z}^j$ for any $j \geq 1$, both sequences must converge:

$$\lim_{j\to\infty} \underline{z}^j = \underline{z}^*, \lim_{j\to\infty} \bar{z}^j = \bar{z}^*, \text{ and } \underline{z}^* < \bar{z}^*.$$

The last inequality is strict because of the assumption that we do not have a finite convergence of the algorithm. Consider an arbitrary $\delta > 0$. We will show that there exists $\hat{j}$ such that for any $j \geq \hat{j} : \bar{z}^j - \underline{z}^j \leq \delta$, thus obtaining a contradiction.

Note that finiteness of Y guarantees that $\underline{z}$ improves after a finite number of steps, and the lower bound can increase only due to one of the following two reasons:

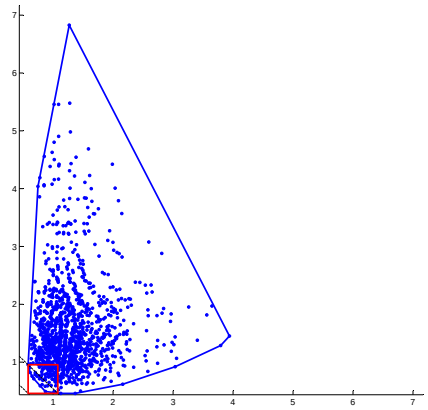
1. $\tilde{y}_r > L_r(P)$, in which case the lower bound increases by $y_r - L_r(P)$;
2. P' is not added to S, i.e., $e^T L(P') \geq \bar{z}$, in which case the increase in lower bound value would be $(\tilde{y}_h - L_h(P))/2$.

Algorithm 1

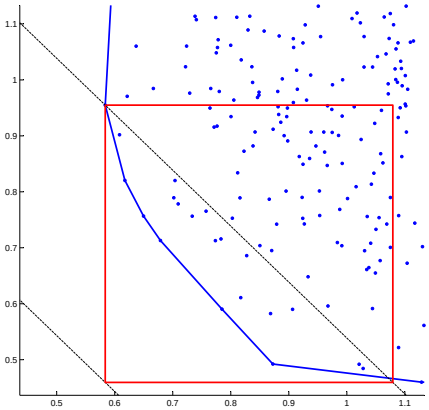
Require: G; $a_r, b_r \in \mathbb{R}^n, r = 1, \ldots, k$; $0 < \epsilon < 1$.

Ensure: $\bar{x}$, an ϵ-optimal solution to (2.7).

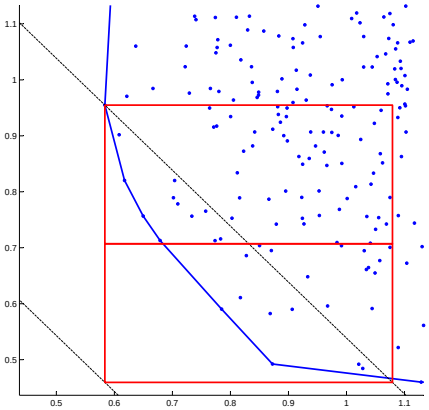
$y^r \leftarrow \arg\min\{y_r : y \in Y\}, r = 1, \ldots, k$;
$\bar{y} \leftarrow \arg\min\{e^T y : y \in \{y^1, \ldots, y^k\}\}$;
$\bar{z} \leftarrow e^T \bar{y}$;
$Q \leftarrow \{y \in \mathbb{R}^k : y_r \geq y_r^r, r = 1, \ldots, k\}$;
$S =\leftarrow \{Q\}$;
Choose $r \in \{1, \ldots, k\}$;
repeat
    $Q \leftarrow$ the first set in S;
    Remove Q from S;
    $\underline{z} \leftarrow e^T L(Q)$;
    $P \leftarrow \{y \in Q : y_s \leq \bar{z} - \underline{z} + L_s(Q), s = 1, \ldots, k\}$;
    $\tilde{y} = \arg\min\{y_r : y \in Y \cap P\}$;
    if $e^T \tilde{y} < \bar{z}$ **then**
        $\bar{z} \leftarrow e^T \tilde{y}$;
        $\bar{y} \leftarrow \tilde{y}$;
    end if
    if $\tilde{y}_r = L_r(P)$ **then**
        Choose $h \in \{1, \ldots, k\} (h \neq r)$ that maximizes $\tilde{y}_h - L_h(P)$;
        $P' \leftarrow \{y \in P : y_h \leq (L_h(P) + \tilde{y}_h)/2\}$;
        $P'' \leftarrow \{y \in P : y_h \geq (L_h(P) + \tilde{y}_h)/2\}$;
        if $e^T L(P'') < \bar{z}$ **then**
            $S \leftarrow S \cup \{P''\}$;
        end if
    else
        $P' \leftarrow \{y \in P : y_r \geq \tilde{y}_r\}$;
    end if
    if $e^T L(P') < \bar{z}$ **then**
        $S \leftarrow S \cup \{P'\}$;
    end if
until $\bar{z} - \underline{z} \leq \epsilon \bar{z}$
return $\bar{x} \in M^{-1}(\bar{y})$;

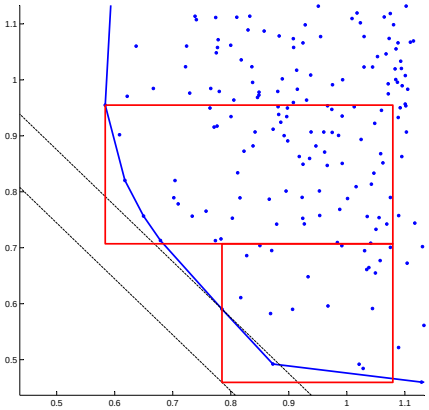


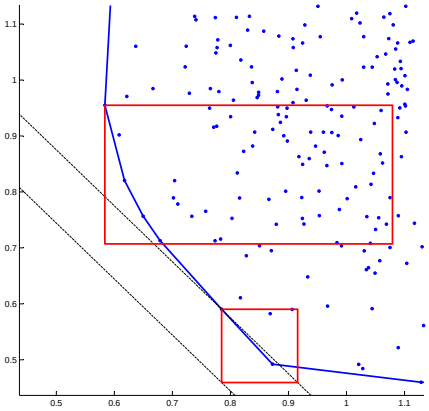
complete image space



initial Q







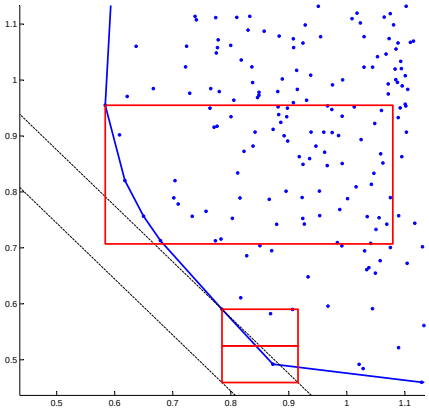


Fig. 1 An illustrative example: complete image space, initial region Q and the first several steps of the algorithm.

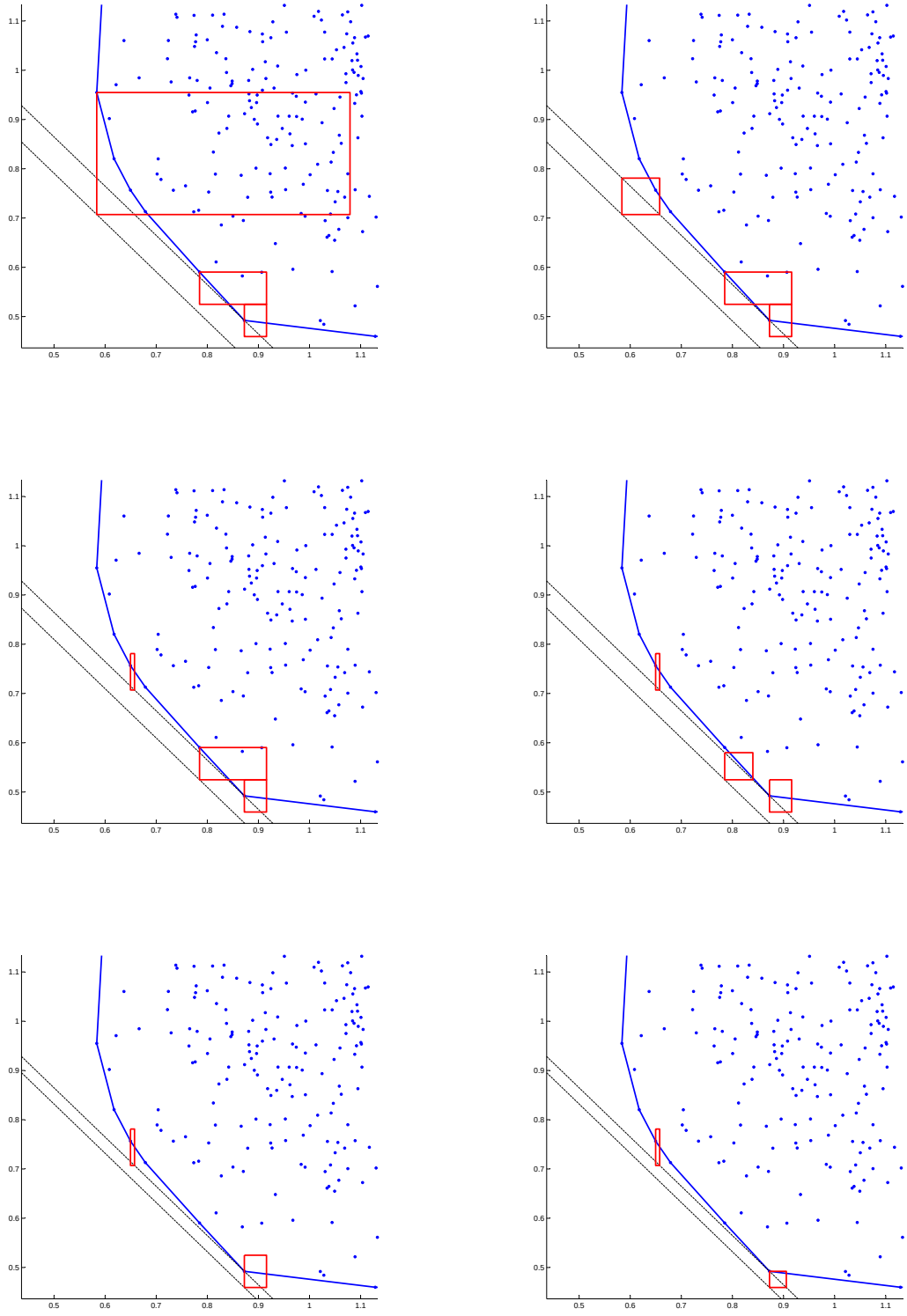


Fig. 1 Continued.

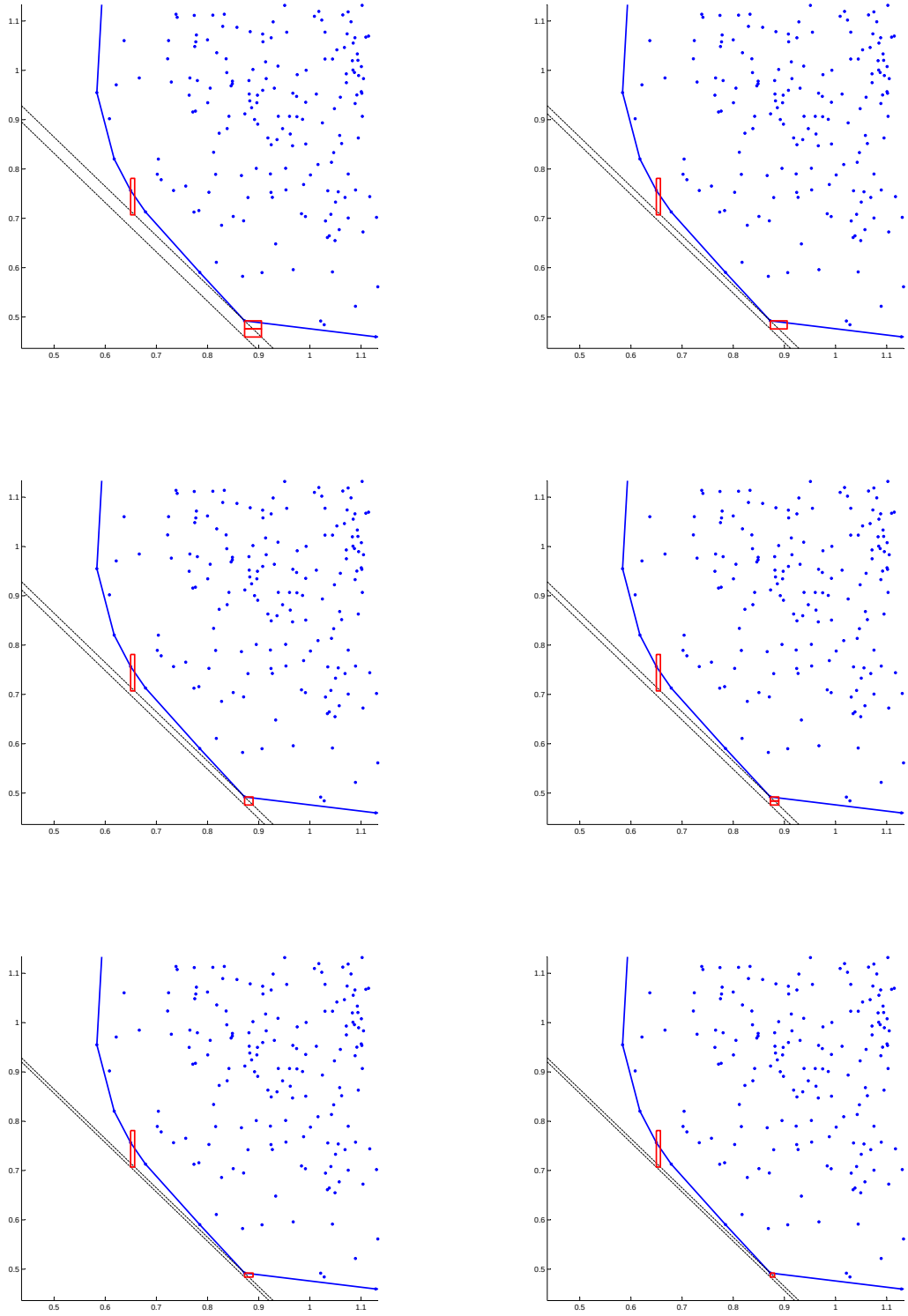


Fig. 1 Continued.

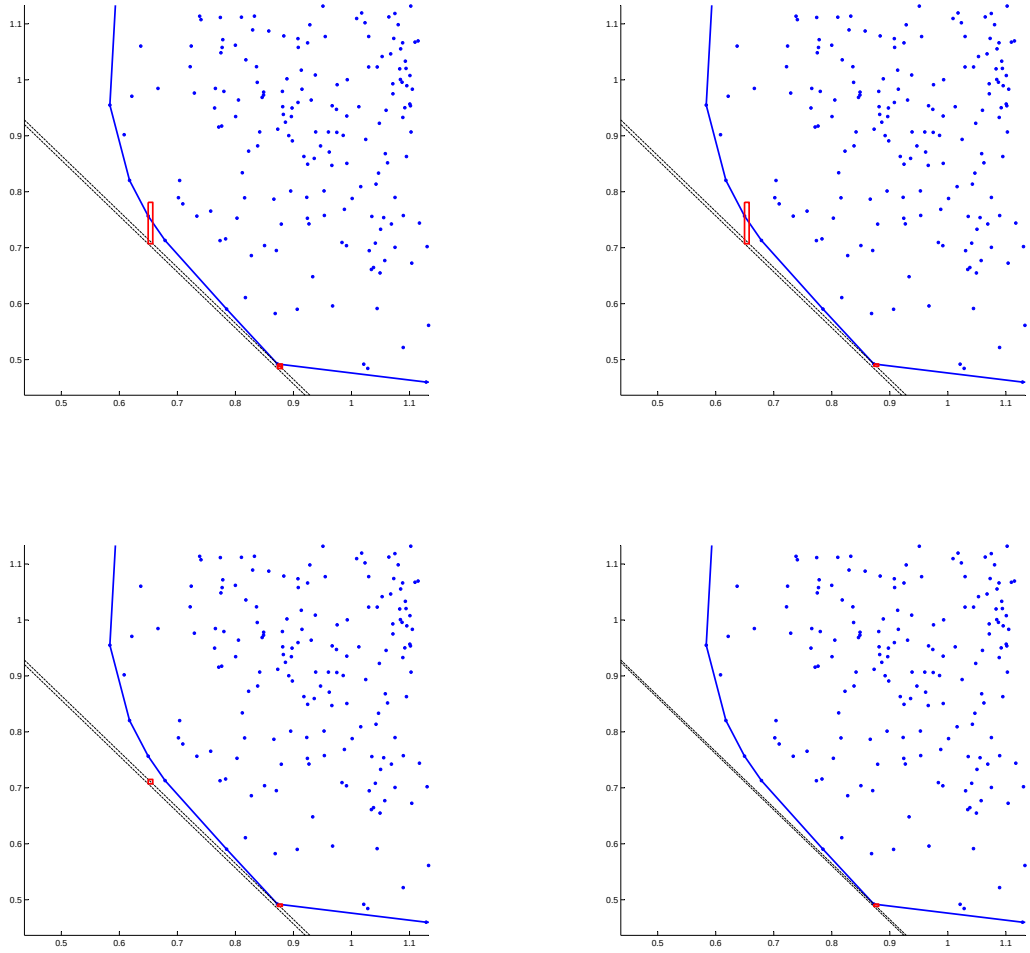


Fig. 1 Continued.

Due to finiteness of Y it is possible to choose $\delta_1 < \min\{|y'_r - y''_r| : y', y'' \in Y, y'_r \neq y''_r\}$, $\delta_2 \leq \min\{\delta_1, \delta/(2k)\}$, and due to convergence of $\{\underline{z}^j : j \geq 1\}$ there exists $\hat{j}$ such that for any $j \geq \hat{j}$ we have $|\underline{z}^j - \underline{z}^{j+1}| < \delta_2$. On the other hand, if $\underline{z}^j$ in the algorithm increases because $\tilde{y}_r > L_r(P)$ then the increase must be at least δ_2. Thus, if $j \geq \hat{j}$, $\underline{z}^j$ can increase only due to the second reason, and the corresponding increase $(\tilde{y}_h - L_h(P))/2$ must be less than δ_2. Since h maximizes $\tilde{y}_s - L_s(P), s = 1, \ldots, k$ and

y_h is a feasible solution, we have

$$\bar{z}^j - \underline{z}^j \leq e^T \tilde{y} - e^T L(P) \leq k(\tilde{y}_h - L_h(P)) < 2k\delta_2 \leq \delta.$$

Thus, $\bar{z}^* = \underline{z}^*$, and we obtain the contradiction with our assumption that the stopping criterion is not satisfied in a finite number of steps. The finite convergence follows. □

V.1. Solving the subproblem

Computational complexity of each iteration of the algorithm described above is defined by the complexity of solving the subproblem (5.1), therefore it is imperative to solve this problem effectively. Returning to the original variable x, for a rectangular region $Q \in \mathbb{R}^k$ it is formulated as

$$\min \ a_r^T x / b_r^T x \tag{5.3a}$$

$$\text{subject to} \quad x \in \mathcal{T} \cap \mathcal{B}, \tag{5.3b}$$

where $\mathcal{B}$ defines Q in terms of x:

$$(a_i - U_i(Q) b_i)^T x \leq 0, \quad i = 1, \ldots, k; \tag{5.3c}$$

$$(L_i(Q) b_i - a_i)^T x \leq 0, \quad i = 1, \ldots, k. \tag{5.3d}$$

The *constrained minimum ratio spanning tree (CMRST)* problem (5.3) above is a generalization of the capacity-constrained version of the MST problem. Unfortunately, the latter problem is NP-hard, as shown by Aggarwal et al. [1], even in the case of one constraint. Unless we specifically mention otherwise, $L_r(Q)$ and $U_r(Q)$ in (5.3c)-(5.3d) should be assumed $-\infty$ and ∞, respectively, i.e., $k = 2$ refers to a single constraint case of (5.3).

An effective branch-and-bound approach is suggested in [1] for the MST problem

with a single capacity constraint. This approach can be directly extended to our problem when $k = 2$, but because it heavily exploits the ability to obtain solutions that satisfy the capacity constraint, further generalization for $k > 2$ is difficult, if at all possible. In fact, the case of multiple capacity constraints in such classical combinatorial optimization problems as the MST problem and the shortest path problem is not addressed in the literature. Therefore, we have developed our own branch-and-bound approach for solving the general CMRST problem when $k \geq 2$.

Each node $\mathcal{N}$ of our branch-and-bound tree is characterized by the sets $F^0_{\mathcal{N}} = \{e \in E(G) : x_e \text{ is fixed to } 0\}$ and $F^1_{\mathcal{N}} = \{e \in E(G) : x_e \text{ is fixed to } 1\}$. To obtain a good lower bound on the objective in each node, we dualize the constraints (5.3c) and (5.3d) and solve the fractional Lagrangian dual introduced by Gol'stein in [18]. Assuming, without loss of generality, that $r = 1$, the Lagrangian dual problem to (5.3) is defined as

$$\max_{v \geq 0} \min_{x \in \mathcal{T}} \mathcal{L}(x, v), \tag{5.4}$$

where $v \in \mathbb{R}^{2k-2}$ and

$$\mathcal{L}(x, v) = \max_{v \geq 0} \min_{x \in \mathcal{T}} \left(\frac{a_1^T x}{b_1^T x} + \sum_{r=2}^{k} \frac{v_{r-1}(a_r - U_r(Q)b_r)^T x}{b_1^T x} + \sum_{r=2}^{k} \frac{v_{k+r-2}(L_r(Q)b_r - a_r)^T x}{b_1^T x} \right). \tag{5.5}$$

We can solve (5.4), e.g., via some subgradient optimization algorithm [30]. We employ the Kelley's cutting plane method, since for moderate k it converges fast for piecewise linear functions in practice. Each new cutting plane generated by the Kelley's method corresponds to a tree $x \in \mathcal{T}$, which may or may not be feasible for our CMRST problem. Depending on whether this is the case, they will be used differently in computing a lower bound for the CMRST problem.

We adopt a branching rule similar to the one introduced in [1]. Suppose that solving the dual problem in node $\mathcal{N}$ yields a solution $\hat{x} \in \mathcal{T}$ feasible to (5.3), and let

$e_1, \ldots, e_p$ ($p \leq m-1$) be all edges of the tree corresponding to $\hat{x}$ that are not fixed in node $\mathcal{N}$. We produce p child nodes $\mathcal{M}_1, \ldots, \mathcal{M}_p$ of $\mathcal{N}$ by additionally fixing some of those edges at each child node. Specifically, a child node $\mathcal{M}_j$ ($j = 1, \ldots, p$) is created by additionally fixing j edges out of $e_1, \ldots, e_p$ according to the rule:

$$\begin{aligned} F^0_{\mathcal{M}_j} &= F^0_{\mathcal{N}} \cup \{e_j\}; \\ F^1_{\mathcal{M}_j} &= F^1_{\mathcal{N}} \cup \{e_1,\ e_2,\ \ldots,\ e_{j-1}\}. \end{aligned} \tag{5.6}$$

Note that if $F^1_{\mathcal{N}}$ is a forest, then it is guaranteed that $F^1_{\mathcal{M}_j}$ is a forest. If several trees feasible to (5.3) are available in $\mathcal{N}$, then we choose a tree that yields the best objective value.

However, it is possible that the procedure that solves (5.4) does not encounter a solution feasible to (5.3). Then we use a different criterion for choosing the edges to branch upon. Let $\bar{v}$ be the optimal solution to (5.4), and the trees $t_1, \ldots, t_w$ define the hyperplanes that are tangent to the lower epigraph of $\mathcal{L}(v) = \min_{x \in \mathcal{T}} \mathcal{L}(x, v)$ at $\bar{v}$ for some $w \geq 1$. Since $epi\mathcal{L} \subset \mathbb{R}^{2k-1}$, to define $\bar{v}$ uniquely we need at least $2k-1$ hyperplanes. Thus, eliminating $w - 2k + 2$ of the w trees guarantees increase in the optimal value of $\mathcal{L}(v)$. Therefore, the branching should be performed on the edges of those particular trees.

However, it may be difficult to obtain *all* hyperplanes tangent to the lower epigraph of $\mathcal{L}(v)$ at $\bar{v}$. Instead, we branch on the edges of the trees $t_1, \ldots, t_\alpha$, corresponding to the *last* α hyperplanes produced by Kelley's method to approximate the epigraph of $\mathcal{L}(v)$. We choose p' edges occurring most frequently in $t_1, \ldots, t_\alpha$, that are not yet fixed, and produce p' child nodes according to the rule (5.6). Clearly, because in this case there is no guarantee for a child node $\mathcal{M}_j$ that $F^1_{\mathcal{M}_j}$ is a forest, we have to check this fact, and discard the node if it is not.

Solving (5.4) via the Kelley's method, in turn, involves solving a sequence of

problems of the form

$$\min_{x \in \mathcal{T}} a^T x / b^T x, \tag{5.7}$$

which is polynomially solvable. We solve (5.7) using the Dinkelbach's method [12], which, again, involves solving a sequence of MST problems. Consequently, to derive a lower bound for (5.3), we examine a sequence of spanning trees of G, each of them being a feasible solution to the original MMRST problem (2.7). Therefore, as we obtain each spanning tree, we examine the value of the original objective that it yields, and improve the upper bound $\bar{z}$ whenever possible.

V.2. Partitioning the feasible region

There is a subtle, yet extremely important from the computational perspective, difference between the cases $k = 2$ and $k > 2$. To solve (2.9) for $k = 2$, one can take advantage of the fact that alternating $r = 1$ and $r = 2$ in

$$\min \ y_r \tag{5.8a}$$

$$\text{subject to} \quad L_i(Q) \leq y_i \leq U_i(Q), \quad i \neq r \tag{5.8b}$$

$$y \in Y \tag{5.8c}$$

virtually rules out the necessity to partition $Q \subset \mathbb{R}^2$. Note also that $L_i(Q)$ may be set to $-\infty$, and thus efficient procedures suggested in [1, 19] for solving (5.8) with only one side constraint may be utilized. In fact, this is the strategy used in the algorithm by Skiscim and Palocsay [39]. Indeed, suppose that y^1 is the solution to (5.8) for $r = 1$ and

$$Q = \{y \in \mathbb{R}^2 : (l_1, l_2) \leq y \leq (u_1, u_2)\}.$$

Now Q is reduced to

$$Q' = \{y \in \mathbb{R}^2 : (y_1^1, l_2) \leq y \leq (e^T y^1 - l_2, y_2^1)\}.$$

Let y^2 be a solution to (5.8) for $r = 2$ and $Q = Q'$.

If $e^T y^2 < e^T y^1$, then we can further reduce Q' to

$$Q'' = \{y \in \mathbb{R}^2 : (y_1^1, y_2^2) \leq y \leq (y_1^2, e^T y^2 - y_1^1)\}$$

thus forcing $y^1 \notin Q''$, since $e^T y^2 < e^T y^1 \Rightarrow e^T y^2 - y_1^1 < y_2^1$. Now that y^1 is separated from $L(Q'')$, we can again solve (5.8) for $r = 1$ and $Q = Q''$ to further improve the bounds on the optimal objective of (2.9).

If $e^T y^2 > e^T y^1$, then we can reduce Q' to

$$Q''' = \{y \in \mathbb{R}^2 : (y_1^1, y_2^2) \leq y \leq (e^T y^1 - y_2^2, y_2^1)\}$$

forcing $y^2 \notin Q'''$, and we can proceed with solving (5.8) for $r = 2$ and $Q = Q'''$.

The only case when the algorithm cannot proceed is $e^T y^2 = e^T y^1$. In [39] the authors restart the procedure by improving the upper bound, thus reducing Q' and forcing both y^1 and y^2 outside of the resulting rectangle. To achieve this, either a local search is performed, or, if the local search fails to improve an incumbent solution, the procedure is applied recursively to $\{y \in Q' : y_s \leq (L_s(Q) + U_s(Q))/2\}$, $s = 1, 2$ until either a better incumbent is found or ϵ-optimality of the current incumbent is proved.

Consider now the case $k > 2$. Let $Q \subset \mathbb{R}^k$ such that

$$L_s(Q) = \min\{y_s \in \mathbb{R} : y \in Y \cap Q\},$$

with y^s being the respective optimal image point, $s = 1, \ldots, k$; and

$$U_s(Q) = \bar{z} - \sum_{s'=1, s' \neq s}^{k} L_{s'}(Q), s = 1, \ldots, k,$$

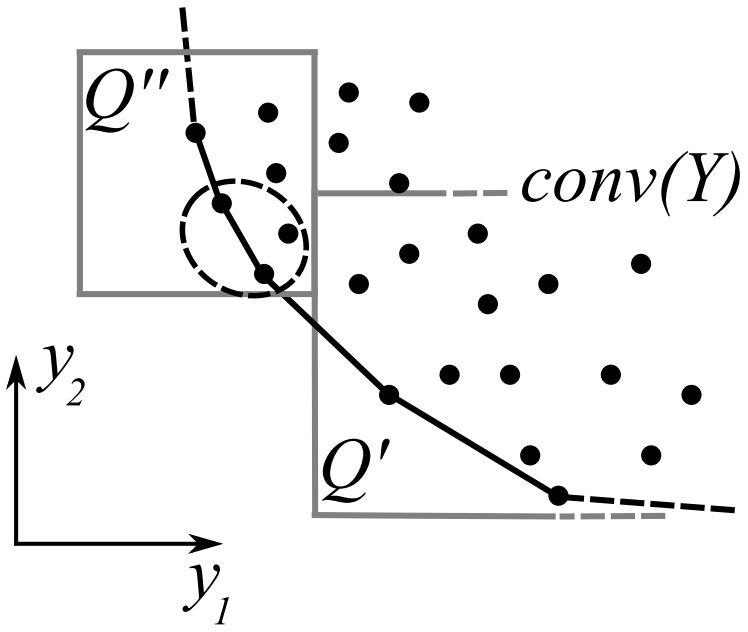


Fig. 2 A two-dimensional illustration of condition (5.9) where $r = 1$.

where $\bar{z} = \min\{e^T y^s, s = 1, \ldots, k\}$.

It is likely that $y^s \in Q'$ for all $s = 1, \ldots, k$ when $k > 2$. This case is analogous to $e^T y^2 = e^T y^1$ for $k = 2$ above, and the procedure by Sciskim and Palocsay [39] outlined above stalls. Improvement of the upper bound, unless it is large enough (which cannot be guaranteed), does not restart the procedure. Therefore, for $k > 2$ partitioning Q is a vital step for the algorithm to proceed. Moreover, it turns out that the way the feasible region is partitioned has a significant impact on the computational performance of the algorithm. In particular, we would like to avoid solving (5.3) with finite $L_r(Q)$. Suppose such subproblem may have to be solved and

$$\exists\, Q', Q'' \in S:\ L_r(Q') \geq U_r(Q''),\ L_s(Q') \leq L_s(Q'') \leq U_s(Q') \ \text{ for some } s \neq r, \quad (5.9)$$

i.e., the regions Q' and Q'' are positioned as shown in Figure 2 with $r = 1$ and $s = 2$.

As the following proposition implies, the situation described by the condition

(5.9) may lead to extremely inefficient computations. Assume that $\mathcal{B}$ is defined as in (5.3c)-(5.3d), $L_r(Q) > -\infty$, and $\mathcal{L}(x, v)$ is the fractional Lagrangian function of (5.3) obtained via dualizing the constraints defining $\mathcal{B}$. Then the following proposition is true.

Proposition 8. *Let $conv(\mathcal{T}) \cap \mathcal{B} \neq \emptyset$, and $\tilde{x} \in \mathcal{T}$ be such that $a_r^T \tilde{x}/b_r^T \tilde{x} < L_r(Q)$, and all other inequalities that define $\mathcal{B}$ are satisfied in $\tilde{x}$. Then*

$$\sup_{v \geq 0} \inf_{x \in \mathcal{T}} \mathcal{L}(x, v) = L_r(Q).$$

Proof. Take some $\bar{x} \in conv(\mathcal{T}) \cap \mathcal{B}$. Let $\hat{x} = \alpha \bar{x} + (1 - \alpha)\tilde{x}$ for some $0 \leq \alpha < 1$ such that $a_r^T \hat{x}/b_r^T \hat{x} = L_r(Q)$. Since $\mathcal{B}$ is convex, such $\hat{x}$ exists. Moreover, it is an optimal solution to the linear relaxation of (5.3)

$$\min \ a_r^T x / b_r^T x \tag{5.10a}$$

$$\text{subject to} \quad x \in conv(\mathcal{T}) \cap \mathcal{B}. \tag{5.10b}$$

Indeed, $x \in \mathcal{B}$ enforces the lower bound of $L_r(Q)$ on the objective, and this bound is achieved at $\hat{x}$. On the other hand, it follows from the results in fractional duality [4, 18, 36] that

$$L_r(Q) = \inf\{a_r^T x/b_r^T x : x \in conv(\mathcal{T}) \cap \mathcal{B}\} = \sup_{v \geq 0} \inf\{\mathcal{L}(x, v) : x \in conv(\mathcal{T})\}.$$

Since $\mathcal{L}(x, v)$ is quasiconcave for any fixed $v \geq 0$, it achieves its minimum over $conv(\mathcal{T})$ in some x^* that is a vertex of $conv(\mathcal{T})$. Thus $x^* \in \mathcal{T}$ and

$$\sup_{v \geq 0} \inf\{\mathcal{L}(x, v) : x \in \mathcal{T}\} = \sup_{v \geq 0} \inf\{\mathcal{L}(x, v) : x \in conv(\mathcal{T})\} = L_r(Q).$$

□

Suppose that Q', Q'' are defined as in (5.9), and $T \subset \mathcal{T}$ is such that

$$\forall x \in T \quad M_r(x) \in Q'' \text{ and } L_s(Q') \leq M_s(x) \leq U_s(Q'), s = 1, \ldots, k, s \neq r.$$

The example displayed on Figure 2 shows $M(T)$ as the image points encircled by a dash line. Then, if the CMRST subproblem (5.3) with the box constraints defined by Q' is solved via the procedure described in subsection V.1, the lower bound on the optimal value of (5.3) obtained in *all* nodes of the branch-and-bound tree will be equal to $L_r(Q')$ until at least one edge is excluded *for each* $x \in T$. Not only this may be a weak bound; what is worse, it leaves the branching process without direction for choosing the next node to process, thus dramatically increasing run time. To rule out the possibility of such a situation to occur, we do not alternate the index r, but choose it to be fixed in Algorithm 1. This way, the boxes can only be split by hyperplanes that are parallel to the r-th coordinate axis. Hence, since we start with a single box, the projections of boxes in S^j at any step j of the algorithm onto any coordinate axes other than r-th never overlap.

It should be clear, that the run time of the main algorithm does depend on the choice of r, as it depends on the shape of $conv(Y)$. It may be chosen, for example, by running a few iterations of the algorithm for every $r = 1, \ldots, k$, and choosing the ratio along which the lower bound progresses faster.

As a side note, it is clear from the proof above, that (5.4) will yield the same bound as (5.10) due to the integrality of $conv(\mathcal{T})$. However, even "compact" descriptions of the MST polytope (see, for example [23]) may be huge for moderate instances. Moreover, solving a linear program would neither provide numerous incumbent solutions for MMRST, nor would it give information for branching as valuable as (5.4) does.

V.3. Computational experiments

In order to be consistent, we test the global optimization algorithm on the same batch of the MMRST instances that was used in numerical experiments in Chapter IV.

All algorithms are implemented in C++ using Microsoft Visual Studio 2003 environment[1]. We rely on the Boost Graph Library [38] implementation of adjacency matrix to represent graphs. The experiments were performed on a computer with Intel® Core™ 2 Duo 3.16 GHz CPU and 3.23 GB of RAM.

Tables 5 and 6 summarize the computational results for complete graph instances and sparse connected random graphs, respectively. As in Chapter IV, we tested a batch of 3 instances for each reported pair (k, n). For each approach the target gap value is set to 1%, and computation time is limited to 1 hour. Average run time, average number of steps (*i.e.*, subproblems solved), as well as average final gap values are reported for each batch.

It is evident that performance of this optimization approach depends on both k and $|\mathcal{T}|$. On one hand, both of these factors have their impact on how difficult it is to build the approximation of $conv(Y)$ that is accurate enough. On the other, computational complexity of each iteration also depends on both of these factors. As expected, the results suggest that k primarily affects the number of iterations, and that $|\mathcal{T}|$ mostly affects the time per iteration. A less expected (and encouraging) empirical conclusion can be drawn from Figure 3, which presents convergence of bounds on the optimal objective value for the hardest tested instances. It turns out that an optimal or near-optimal solution is found by the algorithm early, and most of the time is spent on proving quality of an incumbent. This tendency is even more obvious for easier instances. Most likely this should be contributed to a large number

[1]The source code is available upon request.

Table 5 Performance of Algorithm 1 on complete graph instances

k	n	steps	run time (sec.)	gap (%)
2	10	12.4	0.1	0.6
	15	20.0	0.6	0.9
	20	26.0	2.6	0.9
	30	37.0	16.8	0.9
	40	28.6	37.0	0.9
	50	39.0	87.0	0.9
	80	65.4	1053.0	1.0
	100	65.8	2941.0	3.9
3	10	51.0	1.1	0.9
	15	104.2	14.6	0.9
	20	190.4	108.0	1.0
	30	495.6	1801.0	1.0
	40	322.0	3465.0	1.9
	50	46.0	3600.0	16.2
4	10	318.0	19.8	0.9
	15	999.4	441.0	1.0
	20	1773.2	3424.0	1.9
	30	198.6	3600.0	17.7
5	10	1534.0	181.0	0.9
	15	4101.2	3600.0	2.9
	20	710.0	3600.0	8.9

Table 6 Performance of Algorithm 1 on sparse random graph instances

k	n	steps	run time (sec.)	gap (%)
2	20	5.8	0.1	0.8
	40	6.0	0.3	0.8
	60	16.6	8.9	0.9
	80	20.6	46.9	0.9
	100	50.6	371.0	0.9
	120	54.0	996.3	1.0
	140	63.6	2677.0	1.2
	160	50.0	3600.0	3.5
3	20	19.4	0.25	0.7
	40	34.0	6.8	0.9
	60	141.2	387.7	0.9
	80	184.6	3000.6	1.1
	100	49.6	3600.0	6.3
4	20	23.4	0.6	0.8
	40	210.0	87.0	0.9
	60	386.2	2610.0	1.7
	80	39.6	3600.0	10.6
5	20	154.0	3.3	1.0
	40	610.0	208.0	1.0
	60	243.6	3600.0	6.0

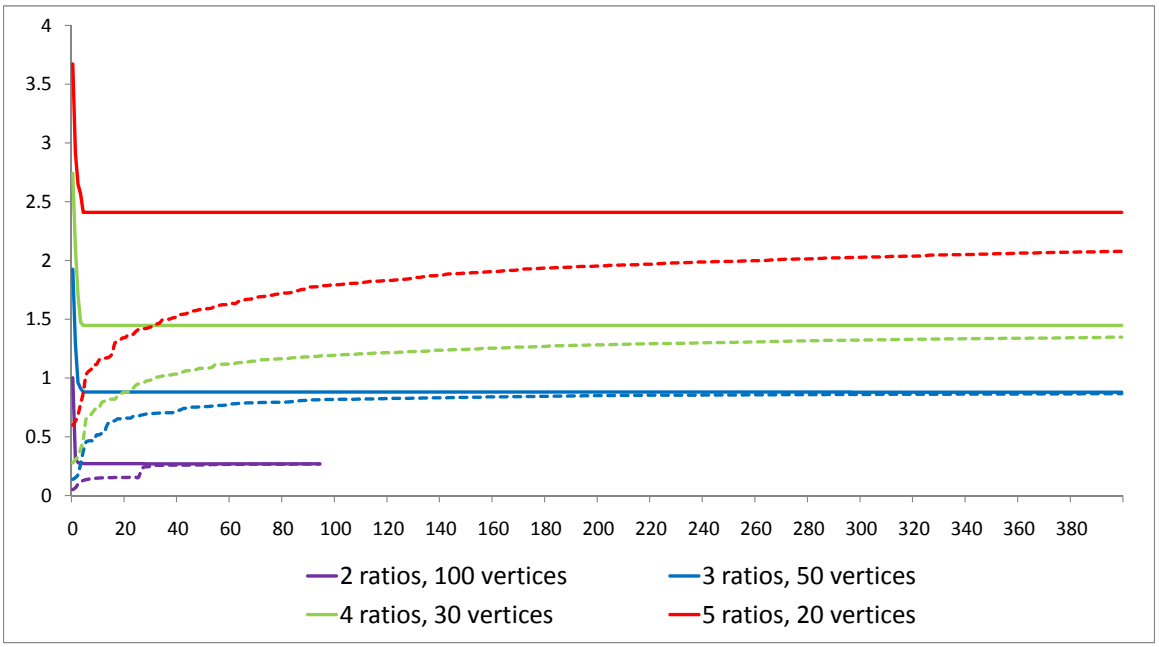


Fig. 3 Convergence of bounds for the hardest complete graph instances.

of trees examined on each step of the algorithm. Therefore, when the size of the instance does not allow to prove near-optimality in a reasonable time, the suggested algorithm may still be used as a good heuristic.

CHAPTER VI

A GLOBAL OPTIMIZATION APPROACH FOR THE CLASS OF FRACTIONAL COMBINATORIAL PROBLEMS

This chapter inherits most of the notation from Charter V. Whenever we depart from the notation adopted there, we specifically mention this. In the previous chapter we solved MMRST via approximating the portion of the problem's image polytope $conv(Y)$ that contained the optimal vertex. Specifically, we did this by building a sequence of sets

$$S^j = \{Q_1^j, \ldots, Q_j^t\}, t \leq j.$$

The sets $Q_i^j \subset \mathbb{R}^k$ are rectangular regions, and for some $M \in \mathbb{N}$ the sequence satisfied

$$e^T\bar{y} - \inf\{e^T L(Q) : Q \in S^m\} \leq \epsilon e^T\bar{y}\ \forall \epsilon > 0, m > M,$$

where $\bar{y}$ is the image of an incumbent solution to the original problem. Although the algorithm can be easily extended to other FCOPs of the considered class, it has an inherent drawback. In order to construct S^m we had to solve m NP-hard subproblems. Taking into account that m may be large, however effective the procedure for solving a subproblem may be, this design is hardly suitable for large-scale instances.

Another way to tackle this class of problems is to use the approach that is traditional in linear integer programming. That is, to solve the problem via branch-and-bound using linear relaxation for underestimation of the optimal value in a tree node. Unfortunately, following this approach directly leads us to the same pitfall, since the objective function

$$f(x) = \sum_{i=1}^{k} \frac{a_i^T x}{b_i^T x}$$

is in general multiextremal, and thus finding its global extremum over a convex set is a hard problem. Consequently, in each node of the branch-and-bound tree we still would have to solve a hard problem, ultimately performing exponential number of iterations, each having exponential complexity. In order to reduce the computational load in each node of the branch-and-bound tree, several approaches may be used. For instance, one could use an underestimator of $f(x)$ that is polynomially computable. The success in this case depends on how tight such an underestimation is. Another conceptually related way is to obtain an initial partition of the search space in the root node, and underestimate $f(x)$ in each region separately in every node. This should significantly reduce the computational load as the depth of the tree increases, as well as tighten the overall approximation.

In this chapter we develop the global optimization algorithm for a class of FCOPs that have a polynomially solvable single-ratio version. Thus we refer to the search space as $\mathcal{X}$ rather than $\mathcal{T}$ used for the set of characteristic vectors of trees in a graph G in order to emphasize more diverse nature of the combinatorial objects. Our approach here is generally based on a branch-and-bound with an underestimation of $f(x)$ over $conv(\mathcal{X})$ guiding the search. In order to mitigate the computational complexity of underestimation, we use a combination of the approaches mentioned above. The resulting optimization technique is the main contribution of the research described in this chapter.

VI.1. Obtaining outer approximation of $conv(\mathcal{X})$

In order to underestimate the sum-of-ratios objective f in $conv(\mathcal{X})$ and partition a feasible region, we use an algorithm that, similarly to Algorithm 1 in Chapter V,

$\forall \epsilon > 0$ builds a set S of rectangles in the image space $\mathbb{R}^k$ such that

$$\bar{f} - \inf\{e^T L(Q) : Q \in S\} \leq \epsilon \inf\{e^T L(Q) : Q \in S\}. \tag{6.1}$$

By $\bar{f}$ we denote here an upper bound for $\inf\{f(x) : x \in conv(\mathcal{X})\}$. The difference between the algorithms is that Algorithm 1 approximates a portion of $conv(Y)$, while Algorithm 2 that we describe in this section does the same thing to $\mathcal{Y} \equiv M(conv(\mathcal{X}))$. It takes $Q_0 \subset \mathbb{R}^k$ as an initial approximation of a portion of $\mathcal{Y}$ that is interesting to us, and continues refining the approximation until the desired accuracy is achieved. The starting rectangle Q_0 may be computed as in the initial steps of Algorithm 1. Specifically, we can put

$$\begin{aligned} L_i(Q_0) &= \inf\{y_i : y \in Y\} \quad i = 1 \ldots, k; \\ U_i(Q_0) &= \bar{u} - \sum_{\substack{j=1 \\ j \neq i}}^{k} L_j(Q_0), \quad i = 1 \ldots, k; \end{aligned} \tag{6.2}$$

provided we have an upper bound $\bar{u}$ for $\inf\{e^T y : y \in Y\}$. For the class of problems considered here such Q_0 is guaranteed to contain global minimizers of $e^T y$ over both Y and $\mathcal{Y}$ (the former may not be true when $conv(\mathcal{X})$ is not integral).

We use the following additional notation in the listing of the algorithm to express the stopping criterion (6.1). Let $Q \subset \mathbb{R}^k$ be a rectangle; we put

$$\pi_r(Q) = \sum_{\substack{i=1 \\ i \neq r}}^{k} U_i(Q) - L_i(Q).$$

In coherence with the previous chapter we also denote $M^{-1}(Q)$, i.e., the polyheadral cone in $\mathbb{R}^n$ that corresponds to a rectangle $Q \subset \mathbb{R}^k$, as $C(Q)$.

Proposition 9. *Algorithm 2 terminates after finite number of steps and returns a*

Algorithm 2

Require: $\mathcal{X}$; $Q_0 \in \mathbb{R}^k$; $r \in \{1, \ldots, k\}$; $a_i, b_i \in \mathbb{R}^n, i = 1, \ldots, k$; $0 < \epsilon < 1$.
Ensure: $S = \{Q_1, \ldots, Q_m\}$ that satisfies (6.1), $\underline{z} = \inf\{e^T L(Q) : Q \in S\}$

1: $Q \leftarrow \{y \in Q_0 : y_r \geq \inf_{y \in \mathcal{Y} \cap Q_0} y_r\}$;
2: **if** $L_r(Q) = \infty$ **then**
3: **return** ∞;
4: **end if**
5: $t \leftarrow 1$;
6: $S^t \leftarrow \{Q\}$;
7: $\underline{z} \leftarrow e^T L(Q)$;
8: **while** $\pi_r(Q) > \epsilon \underline{z}$ **do**
9: Choose $j \in \{1, \ldots, k\}, j \neq r$ that maximizes $U_j(Q) - L_j(Q)$;
10: $Q' \leftarrow \{y \in Q : (L_j(Q) + U_j(Q))/2 \geq y_j\}$;
11: $Q'' \leftarrow \{y \in Q : (L_j(Q) + U_j(Q))/2 \leq y_j\}$;
12: $Q' \leftarrow \{y \in Q' : y_r \geq \inf_{y \in \mathcal{Y} \cap Q'} y_r\}$;
13: $Q'' \leftarrow \{y \in Q'' : y_r \geq \inf_{y \in \mathcal{Y} \cap Q''} y_r\}$;
14: $S^{t+1} \leftarrow S^t \setminus \{Q\} \cup \{Q', Q''\}$;
15: $t \leftarrow t + 1$;
16: $Q \leftarrow \arg\min\{e^T L(q) : q \in S^t\}$;
17: $\underline{z} \leftarrow e^T L(Q)$;
18: **end while**;
19: **return** $S^t, \underline{z}$;

lower bound $\underline{z}$ *for* $z^* = \inf\{f(x) : x \in C(Q) \cap conv(\mathcal{X})\}$ *such that*

$$z^* - \underline{z} \leq \epsilon \underline{z} \; \forall \epsilon > 0.$$

Proof. On each step t the algorithm maintains an invariant

$$\mathcal{Y} \cap Q \subset \cup q : q \in S^t,$$

and by the definition of the image space

$$\inf\{f(x) : x \in C(Q) \cap conv(\mathcal{X})\} \equiv \inf\{e^T y : y \in Q \cap \mathcal{Y}\}.$$

Hence $\underline{z} \leq z^*$. On the other hand, the steps 1,12 and 13 of the algorithm guarantee

that $\forall Q \in S^t, t > 0, \exists y \in \mathcal{Y} \cap Q$ such that $y_r = L_r(Q)$. Consequently, denoting $\arg\min\{e^T L(Q) : Q \in S^t\}$ by Q_t, we have

$$e^T L(Q_t) \leq z^* \leq \sum_{i=1, t \neq r}^{k} U_i(Q_t) + L_r(Q_t) \quad \forall t > 0.$$

Therefore, after iteration t we have

$$z^* - \underline{z} \leq \sum_{i=1, i \neq r}^{k} U_i(Q_t) + L_r(Q_t) - e^T L(Q_t) = \pi_r(Q_t).$$

Since the partitioning rule (lines 9-11 of the algorithm) ensures that $\pi_r(Q_t) \to 0$ as $t \to \infty$. Assuming without loss of generality that $\underline{z} \neq 0$, the stopping condition is met for $\epsilon > 0$ in finite number of steps. □

Algorithm 2, in fact, is a special case of a more general sum-of-ratios optimization scheme suggested by Dur et al. in [13]. It is interesting to observe how $\mathcal{Y}$ and its outer approximation given by the algorithm compare to $conv(Y)$, the "image polytope" that we attempted to approximate in chapter V via Algorithm 1. Figures 4 and 5 visualize these comparisons respectively for a random complete MMRST instance with 6 vertices and 2 ratios. The initial rectangle Q_0 was chosen as in (6.2). These figures provide some empiric insight of how well $\mathcal{Y}$ may approximate $conv(Y)$, thus emphasizing importance of representing $\mathcal{Y}$ adequately via S for the sake of computational efficiency of the global optimization algorithm that we discuss in Section VI.4 of this chapter. As a side note, the instance with the *least* resemblance between $\mathcal{Y}$ and $conv(Y)$ has been chosen for illustration from among multiple ones. The size of the graph has to be small, since total enumeration of the combinatorial objects is required to visualize $conv(Y)$. Obviously, Algorithm 2 has exponential computational complexity, since it solves an NP-hard problem with any given accuracy. Let us, nevertheless, assess its performance with more scrutiny. Observe that the algorithm

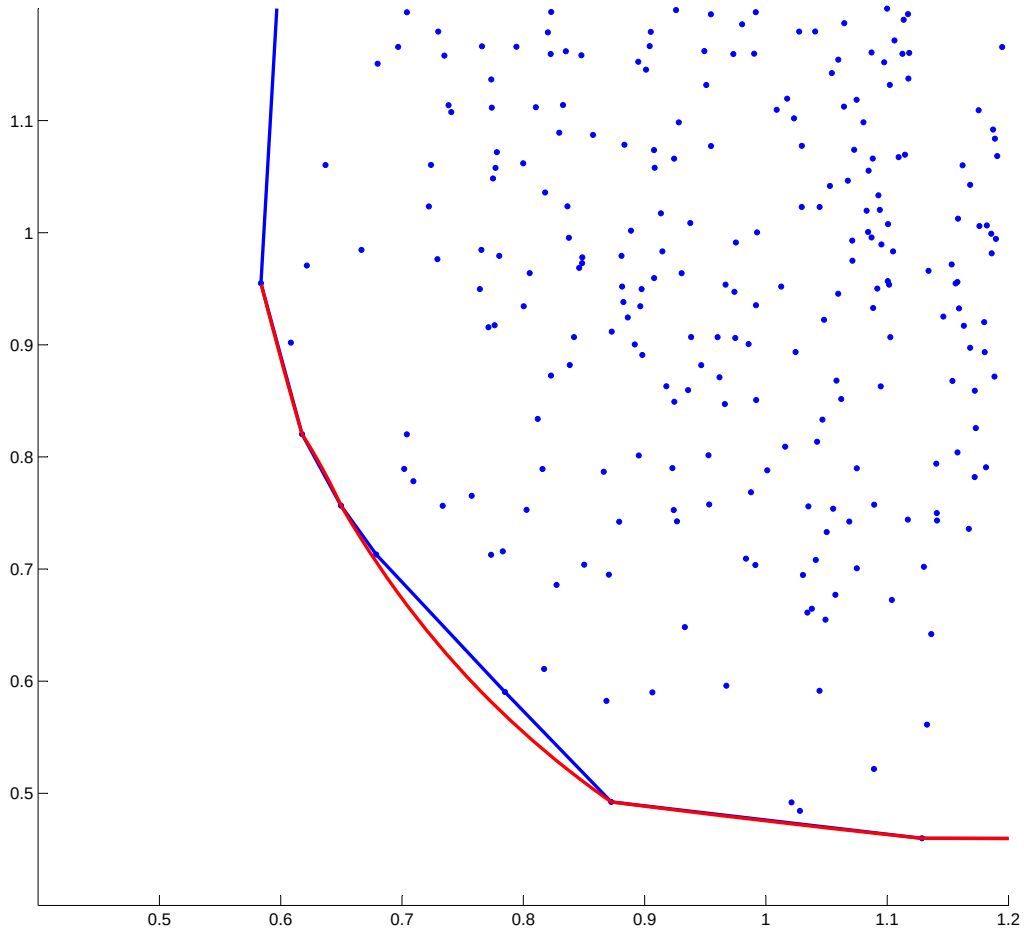


Fig. 4 Comparison of $conv(Y)$ vs. $\mathcal{Y}$.

explores the given initial rectangle Q_0 in a branch-and-bound fashion, and the value of $\pi_r(Q), Q \in S^t$, uniquely determines the branch-and-bound tree depth at which Q was obtained. The following facts follow from this observation.

Proposition 10. *Let Q_0, r and ϵ be inputs to Algorithm 2. Denote the value of $\pi_r(Q)$ obtained at the branch-and-bound tree depth d by $\pi_r^d(Q_0)$.*

1. *Algorithm 2 terminates after at most 2^d steps, where d is the minimum depth*

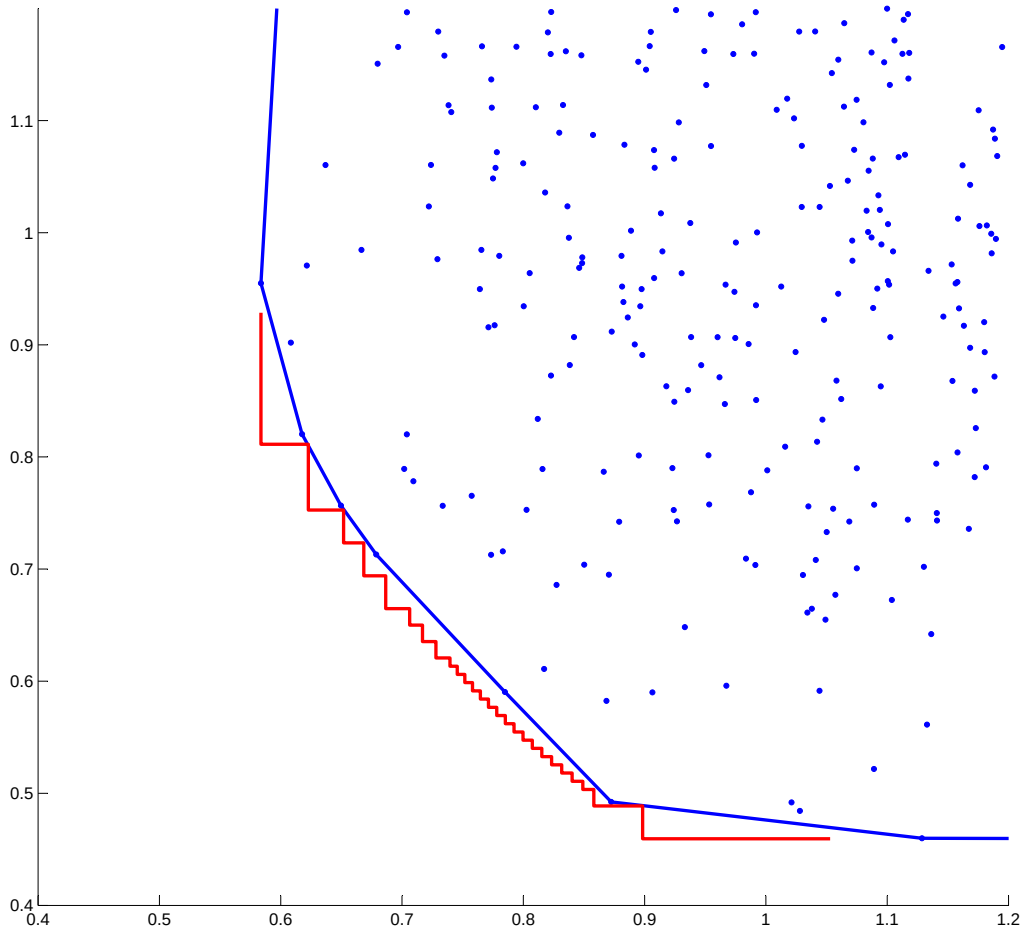


Fig. 5 Comparison of $conv(Y)$ vs. approximation of $\mathcal{Y}$ via Algorithm 2 after 25 steps.

such that

$$\pi_r^d(Q_0) \leq \epsilon e^T L(Q_0).$$

2. *After m steps, Algorithm 2 guarantees accuracy no worse than*

$$\epsilon = \frac{\pi_r^{\lfloor \log_2 m \rfloor}(Q_0)}{e^T L(Q_0)}.$$

Proof. (1) Suppose we change the stopping criterion of the algorithm to $\pi_r(Q) \leq \epsilon e^T L(Q_0)$. The convergence result still holds, since it is true for any $\epsilon > 0$.

Moreover, since $e^T L(Q_0)$ is the minimum possible value for the lower bound, the number of iterations performed by the algorithm does not decrease with this stopping criterion. The algorithm in this case obviously terminates when the branch-and-bound tree reaches the depth d, hense the maximum number of iterations in is bounded by 2^d.

(2) After m steps the branch-and-bound tree is guaranteed to reach the depth $\lceil \log_2 m \rceil$. Consequently, a rectangle Q obtained at the depth at least $\lfloor \log_2 m \rfloor$ has been found to provide the smallest lower bound among the members of S. Underestimating $\bar{z}$ by $e^T L(Q_0)$ gives the minimum guaranteed accuracy.

□

The facts above give some justification for our choice of partitioning the search region over the uniform grid approach. Also, they express quantitatively the tradeoff between the size of partition and accuracy of the approvimation of $\mathcal{Y}$ in the vicinity of an optimal solution. These objectives are conflicting, but each is important for reducing the computational burden as we look for an ϵ-optimal solution to a FCOP. Yet several ways for improvement along *both* are evident:

- Try to reduce the initial region Q_0 as much as possible;
- Attempt to improve upon the 'easy' lower bound of $e^T L(Q)$;
- Choose r appropriately.

The first two items are discussed in the next sections. A justified choice for the input r is revealed in the discussion above. In order to reduce worst-case computational complexity it is reasonable to choose r that minimizes $\pi_r(Q_0)$. This remark should seem irrelevant now, since Q_0 calculated as in (6.2) is cubic, i.e., $\pi_1(Q_0) = \pi_2(Q_0) = \ldots = \pi_k(Q_0)$. It will gain more meaning in section VI.3.

The amount of computations in Algorithm 2 can be reduced. Notice that in the lines 12-13 in order to find

$$\begin{aligned} l' = &\quad \inf\{y_r : y \in \mathcal{Y} \cap Q'\} \quad \text{and} \\ l'' = &\quad \inf\{y_r : y \in \mathcal{Y} \cap Q''\} \end{aligned}$$

we do not have to always solve optimization problems in both cases, because $Q' \cup Q'' = Q$, and $L(Q)$ is known. Therefore, if we find that $l' < L(Q)$, we can set $l'' = L(Q)$. The following fact allows us to further reduce the computational load in the special case of 2 ratios by eliminating the lower bound constraint when solving constraint ratio subproblems in Algorithm 2.

Proposition 11. *Let $a_i, b_i \in \mathbb{R}^n$, $i = 1, 2$, and $l, u \in \mathbb{R}$ such that $l \leq u$. Let x^* be a minimizer of $a_1^T x / b_1^T x$ over $conv(\mathcal{X})$. Then $\forall l \leq a_2^T x^* / b_2^T x^*$ the problems*

$$P_1: \begin{array}{ll} \min & a_1^T x / b_1^T x \\ s.t.: & a_2^T x / b_2^T x \leq u; \\ & a_2^T x / b_2^T x \geq l; \\ & x \in conv(\mathcal{X}); \end{array} \quad \text{and} \quad P_2: \begin{array}{ll} \min & a_1^T x / b_1^T x \\ s.t.: & a_2^T x / b_2^T x \leq u; \\ & x \in conv(\mathcal{X}); \end{array}$$

have the same optimal objective value.

Proof. Observe that both problems must be either feasible or not simultaneously. The case when both have no solutions is trivial. Otherwise, because $a_1^T x / b_1^T x$ is monotonic along every direction in $\mathbb{R}^n$, $\forall x' \in conv(\mathcal{X})$ such that $a_2^T x' / b_2^T x' = l$ $\exists x'' \in conv(\mathcal{X})$ such that $a_2^T x'' / b_2^T x'' = u$ and $a_1^T x'' / b_1^T x'' \leq a_1^T x' / b_1^T x'$. Therefore, the constraint $l \leq a_2^T x / b_2^T x$ in P_1 is superfluous and can be removed without affecting the optimal objective value. □

VI.2. Underestimation of $f(x)$

In this section we suggest several ways to underestimate $f(x)$ over $conv(\mathcal{X})$ or its part intersecting the cone $C(Q)$ defined by an image space rectangle $Q = \{y \in \mathbb{R}^k : l_i \leq y_i \leq u_i, i = 1, \ldots, k\}$. To make the notation less cumbersome, let us consider $f(x)$ in some polytope $P \subset \mathbb{R}^n : 0 \notin P$. We also maintain our assumption that $b_i^T x > 0 \ \forall x \in P$, $i = 1, \ldots, k$, and, in addition, assume that the numerators $a_i^T x$, $i = 1, \ldots, k$ are unisignant on P. The latter precondition is a little restrictive, as the sign of the numerator does not affect polynomial-time solvability of the FCOPs under our consideration, but we can overcome this obstacle at additional computational expense by partitioning P appropriately and computing the bounds separately in each part. Note that the conic constraints that define Q give explicit bounds on the values of the ratios, and hence the numerators (assuming that the denominators are positive). So suppose that

$$I^+ = \{i : a_i^T x \geq 0 \ \forall x \in P\} \text{ and } I^- = \{i : a_i^T x < 0 \ \forall x \in P\}$$

We start with a linear underestimator. Suppose we replace $b_i^T x$ by

$$\beta_i = \begin{cases} \sup_{x \in P} b_i^T x & i \in I^+, \\ \inf_{x \in P} b_i^T x & i \in I^-. \end{cases}$$

Then, obviously,

$$\frac{a_i^T x}{\beta_i} \leq \frac{a_i^T x}{b_i^T x} \quad \forall x \in P, i = 1, \ldots, k.$$

and consequently

$$\phi_1(x) = \sum_{i=1}^{k} \frac{a_i^T x}{\beta_i} = \left(\sum_{i=1}^{k} \frac{a_i}{\beta_i} \right)^T x \leq f(x) \quad \forall x \in P. \tag{6.3}$$

The benefit of this underestimator is that it is computationally cheap, since the vector

$\beta_i \in \mathbb{R}^k$ can be computed via solving k linear problems.

Next, we consider an underestimator based on the sum of at most two linear fractions. Suppose we overestimate the vectors b_i, $i \in I^+$, by a vector $\bar{\beta} = (\bar{\beta}_1, \ldots, \bar{\beta}_n)$, where

$$\bar{\beta}_j = \max_{i \in I^+} \{b_{ij}\} \quad j = 1, \ldots, n,$$

and underestimate the vectors b_i such that $i \in I^-$, by a vector $\underline{\beta} = (\underline{\beta}_1, \ldots, \underline{\beta}_n)$, where

$$\underline{\beta}_j = \min_{i \in I^-} \{b_{ij}\} \quad j = 1, \ldots, n,$$

To unify the notation, when reasonable, let us introduce

$$\underline{\bar{\beta}} = \begin{cases} \bar{\beta} & i \in I^+ \\ \underline{\beta} & i \in I^- \end{cases}.$$

Then again, under the assumptions mentioned above, we have

$$\frac{a_i^T x}{\underline{\bar{\beta}}^T x} \leq \frac{a_i^T x}{b_i^T x} \quad \forall x \in P, i = 1, \ldots, k.$$

The underestimator above is expected to be rather weak, since, provided there is no $j \in I^+$ such that $b_j \geq b_i \ \forall i \in I^+$, or $j \in I^-$ such that $b_j \leq b_i \ \forall i \in I^-$,

$$\forall i : \frac{a_j^T x}{\underline{\bar{\beta}}^T x} < \frac{a_j^T x}{b_j^T x} \ \forall x \in P, \ i = 1, \ldots, k.$$

Moreover, assuming low pairwise correlation between the denominator vectors, the bound should deteriorate as the distance $\|b_i - b_j\|$ increases. The following modification aims to remedy this weakness. Since

$$\frac{a_i^T x}{b_i^T x} = \left(\frac{\bar{\beta}^T x}{b_i^T x} \right) \frac{a_i^T x}{\underline{\bar{\beta}}^T x} \quad i = 1, \ldots, k,$$

we have

$$\frac{a_i^T x}{b_i^T x} \geq \left(\inf_{x \in P} \frac{\bar{\beta}^T x}{b_i^T x} \right) \frac{a_i^T x}{\bar{\beta}^T x} \geq \frac{a_i^T x}{\bar{\beta}^T x} \quad i \in I^+, \tag{6.4}$$

and

$$\frac{a_i^T x}{b_i^T x} \geq \left(\sup_{x \in P} \frac{\underline{\beta}^T x}{b_i^T x} \right) \frac{a_i^T x}{\underline{\beta}^T x} \geq \frac{a_i^T x}{\underline{\beta}^T x} \quad i \in I^-. \tag{6.5}$$

In fact, unless all b_i are equal for $i \in I^+$, the last inequality in (6.4) is strict for at least one $i \in I^+$. The same is true for (6.5) unless all $b_i, i \in I^-$ are equal. Also, such strengthening guarantees that the first inequlities in (6.4) and (6.5) are respectively tight in at least

$$\arg\min_{x \in P} \frac{\bar{\beta}^T x}{b_i^T x}, \ i \in I^+ \quad \text{and} \quad \arg\max_{x \in P} \frac{\underline{\beta}^T x}{b_i^T x}, \ i \in I^-.$$

Finally, we arrive at

$$\begin{aligned} \phi_2(x) &= \phi_2^+(x) + \phi_2^-(x) \\ &= \frac{\left(\sum_{i \in I^+} \inf_{\xi \in P} \frac{\bar{\beta}^T \xi}{b_i^T \xi} a_i \right)^T x}{\bar{\beta}^T x} + \frac{\left(\sum_{i \in I^-} \sup_{\xi \in P} \frac{\underline{\beta}^T \xi}{b_i^T \xi} a_i \right)^T x}{\underline{\beta}^T x} \\ &\leq f(x) \quad \forall x \in P. \end{aligned} \tag{6.6}$$

This underestimation is much more computationally expensive than (6.3). In order to obtain the coefficients we need to solve up to $k+2$ linear FCOPs. Minimizing it is also a sum-of-ratios problem unless either I^- or I^+ is empty, but we can again underestimate (6.6) using (6.3), or simply by computing

$$\underline{\phi}_2 = \inf_{x \in P} \phi_2^+(x) + \inf_{x \in P} \phi_2^-(x).$$

Finally, we suggest another quasiconvex piecewise linear fractional underestimator of

$$f(x) = \sum_{i=1}^{k} \frac{a_i^T x}{b_i^T x}, \quad x \in P.$$

For some $j \in \{1, \ldots, k\}$, multiply every fraction by $b_j^T x / b_j^T x$:

$$f(x) = \frac{a_j^T x}{b_j^T x} + \sum_{\substack{i=1 \\ i \neq j}}^{k} \left(\frac{b_j^T x}{b_i^T x} \frac{a_i^T x}{b_j^T x} \right) \Rightarrow f(x) \geq \frac{a_j^T x}{b_j^T x} + \sum_{\substack{i=1 \\ i \neq j}}^{k} \alpha_{ji} \frac{a_i^T x}{b_j^T x}$$

where

$$\alpha_{ji} = \begin{cases} \inf\limits_{x \in P} \dfrac{b_j^T x}{b_i^T x} & i \in I^+ \\ \sup\limits_{x \in P} \dfrac{b_j^T x}{b_i^T x} & i \in I^- \end{cases}, 1 \leq i \leq k, i \neq j.$$

Performing this for every $j = 1, \ldots, k$, we arrive at

$$\phi_3(x) = \max_{1 \leq j \leq k} \left\{ \frac{\left(\sum\limits_{i=1}^{k} \alpha_{ji} a_i \right)^T x}{b_j^T x} \right\} \leq f(x) \quad \forall x \in P. \tag{6.7}$$

Underestimator (6.7) is also a computationally intensive one. Calculating coefficients only requires solving $k(k-1)$ linear fractional problems. In our case this, however, can be done using efficient combinatorial algorithms. But ϕ_3, being a pointwise maximum of quasiconcave (and also quasiconvex) functions, is no longer necessarily quasiconcave, and thus may not reach its minimum in a vertex of P. Therefore, we cannot rely on combinatorial algorithms to compute $\inf_{x \in P} \phi_3(x)$. We can still do this in polynomial time using other techniques, because ϕ_3 is quasiconvex. One such approach is the generalized Dinkelbach's algorithm [12], but the procedure requires solving a sequence of LPs. Other methods are available and have similar computational complexity. Taking into account formidable size of analytical representation of P even for moderate instances of the considered combinatorial problems, we deem *direct* use of ϕ_3 for calculating the lower bound on $\inf_{x \in P} f(x)$ impractical. We may, however, find it useful for another purpose, as we show in the next section. Also we can derive

a weaker, but easier to compute lower bound based on (6.7):

$$\underline{\phi}_3 = \max_{1\le j\le k}\left\{\inf_{x\in P}\frac{\left(\sum_{i=1}^{k}\alpha_{ji}a_i\right)^T x}{b_j^T x}\right\} \le \inf_{x\in P}\phi_3(x) \le \inf_{x\in P} f(x). \tag{6.8}$$

This bound requires solving k^2 single-ratio FCOPs, hence it still remains significantly more computationally intensive than (6.3) and (6.6).

A legitimate question is: how do the lower bounds $\underline{\phi}_1 = \inf_{x\in P}\phi_1(x)$, $\underline{\phi}_2$ and $\underline{\phi}_3$ compare to the simple bound

$$\underline{\phi}_0 = \sum_{i=1}^{k}\inf_{x\in P}\frac{a_i^T x}{b_i^T x}$$

and to each other? The answer is that, in general, neither of $\underline{\phi}_i$, $i = 0,\ldots,3$ dominates any other. For each bound at least one problem instance was generated on which that particular bound outperforms the others in P defined by (6.2). In particular, $\underline{\phi}_0$ gives the highest value in the setup used for the figures 4 and 5. To benchmark average performance, we generate 2 batches of instances: one for MMRST and one for MMRP problem. Both batches have similar structure; they include instances with 10 different graph sizes, and for each graph size we generate an instance for 2, 3, 5 and 10 ratios, totaling 40 instances per batch. We compare average performance of the bounds $\underline{\phi}_i$, $i = 1,\ldots,3$ against $\underline{\phi}_0$ as

$$\frac{\underline{\phi}_i - \underline{\phi}_0}{|\underline{\phi}_0|}.$$

The same results are presented relative to instance size and number of ratios in the objective. Tables 7 and 8 contain the results for the MMRST problem. All tested MMRST instances are complete graphs with standard uniform edge parameter vectors a_i, b_i. In addition to an obvious empiric conclusion about the quality of the lower bound $\underline{\phi}_2$, there are several other things to mention. For example, the linear lower

Table 7 Average performance of the suggested lower bounds vs. k for MMRST instances

k	$\frac{\underline{\phi}_1-\underline{\phi}_0}{\lvert\underline{\phi}_0\rvert}$	$\frac{\underline{\phi}_2-\underline{\phi}_0}{\lvert\underline{\phi}_0\rvert}$	$\frac{\underline{\phi}_3-\underline{\phi}_0}{\lvert\underline{\phi}_0\rvert}$
2	2.80	4.15	10.2
3	6.54	8.29	0.99
5	12.3	14.15	0.78
10	18.9	20.24	0.31

bound $\underline{\phi}_1$ seems to perform surprisingly well for the MMRST problem. In fact, both $\underline{\phi}_1$ and $\underline{\phi}_2$ significantly outperform $\underline{\phi}_0$ on *all* 40 instances. This does not contradict our remark above about the case when the latter bound dominated the others, since the notorious 6-vertex instance was not included in the benchmark due to its size. The bound $\underline{\phi}_3$, on the other hand, shows poor performance, and dominated $\underline{\phi}_0$ only in 85% cases. Also, the results for the MMRST instances demonstrate an interesting tendency for all bounds to improve their quality compared to $\underline{\phi}_0$ as the graph size increases. This appears to be true for the first two bounds as the number of ratios increases, too, while $\underline{\phi}_3$ shows the opposite. The performance trends are roughly the same, however less pronounced, for the MMRP instances according to the results presented in Tables 9 and 10. The major differences for MMRP instances are that the linear underestimator performs significantly worse (dominates $\underline{\phi}_0$ for 63% instances), while $\underline{\phi}_3$ does significantly better (91%). The linear fractional underestimator is still a clear winner.

VI.3. Localization of optimal solutions to $\min_{x\in X} f(x)$.

In this section we suggest ways to reduce the initial set $conv(\mathcal{X})$ to a smaller set, in which optimal solutions to $\min\{f(x) : x \in X\}$ lie. Our main algorithm works with

Table 8 Average performance of the suggested lower bounds vs. graph size for MMRST instances

$\|V\|$	$\frac{\underline{\phi}_1-\underline{\phi}_0}{\|\underline{\phi}_0\|}$	$\frac{\underline{\phi}_2-\underline{\phi}_0}{\|\underline{\phi}_0\|}$	$\frac{\underline{\phi}_3-\underline{\phi}_0}{\|\underline{\phi}_0\|}$
10	0.50	0.58	-0.0003
20	1.35	1.61	0.17
30	2.24	2.60	0.17
50	3.81	4.46	0.38
70	5.47	6.39	0.57
100	6.82	8.04	0.64
200	13.37	15.43	1.14
300	18.08	20.65	1.31
400	22.71	26.11	1.59
500	27.16	31.22	1.81

conic partition of $\mathcal{X}$, which corresponds to the rectangular partition of the image space set $\mathcal{Y}$. For this reason we choose to look for the reduced search region in the form of smallest possible image space rectangle Q that contains the optimal image points. In [39] Skiskim and Palocsay start with the rectangle Q_0 calculated as in (6.2). We can try to reduce it further using an iterative procedure similar to the optimization algorithm for continuous sum of linear ratios by Falk and Palocsay [15].

By analogy with section VI.1, we adopt here the following notation:

$$\pi(Q) = \sum_{i=1}^{k} U_i(Q) - L_i(Q).$$

We use $\pi(Q)$ as an indicator of the overall size of Q. Also, we assume that a heuristic H is available that, given a description of the problem $\mathcal{P}$ and a rectangle $Q \in \mathbb{R}^k$ provides a solution $H(\mathcal{P}, Q) \in \mathcal{X}$. This is not a restrictive assumption, since a feasible

Table 9 Average performance of the suggested lower bounds vs. k for MMRP instances

k	$\frac{\underline{\phi}_1-\underline{\phi}_0}{\lvert\underline{\phi}_0\rvert}$	$\frac{\underline{\phi}_2-\underline{\phi}_0}{\lvert\underline{\phi}_0\rvert}$	$\frac{\underline{\phi}_3-\underline{\phi}_0}{\lvert\underline{\phi}_0\rvert}$
2	0.14	1.31	0.89
3	0.06	1.78	0.59
5	0.56	3.59	0.60
10	0.32	3.47	0.09

Table 10 Average performance of the suggested lower bounds vs. graph size for MMRP instances

(l, w)	$\frac{\underline{\phi}_1-\underline{\phi}_0}{\lvert\underline{\phi}_0\rvert}$	$\frac{\underline{\phi}_2-\underline{\phi}_0}{\lvert\underline{\phi}_0\rvert}$	$\frac{\underline{\phi}_3-\underline{\phi}_0}{\lvert\underline{\phi}_0\rvert}$
(5,3)	-0.25	1.22	0.60
(7,5)	-0.30	1.72	0.44
(9,7)	0.04	2.54	0.50
(12,10)	-0.12	4.26	0.56
(14,12)	0.05	6.62	1.18
(17,15)	0.38	1.81	0.58
(20,18)	0.44	1.19	0.14
(22,20)	0.39	1.73	0.21
(24,22)	0.79	1.67	0.53
(27,25)	1.28	2.61	0.65

solution is easy to find for the class of problems that we consider in this dissertation. Our implementation does not use H explicitly. Instead, the procedure that is used

Algorithm 3

Require: $\mathcal{P} = (\mathcal{X}, a_1, \ldots, a_k, b_1, \ldots, b_k)$, $0 < \epsilon < 1$.
Ensure: $Q \in \mathbb{R}^k$, $\bar{x}$

Compute Q_0 as in (6.2);
$\bar{x} \leftarrow H(\mathcal{P}, Q_0)$;
$j \leftarrow 0$;
repeat
    $j \leftarrow j + 1$;
    $l_i \leftarrow \inf\{y_i : y \in \mathcal{Y} \cap Q_j\} \quad i = 1, \ldots, k$;
    $u_i = f(\bar{x}) - \sum_{\substack{r=1 \\ r \neq i}}^{k} l_r \quad i = 1, \ldots, k$;
    $Q_j \leftarrow \{y \in \mathbb{R}^k : l \leq y \leq u\}$;
    $\tilde{x} \leftarrow H(\mathcal{P}, Q_j)$;
    if $f(\tilde{x}) < f(\bar{x})$ **then**
        $\bar{x} \leftarrow \tilde{x}$;
    end if
until $\frac{\pi(Q_{j-1}) - \pi(Q_j)}{\pi(Q_j)} < \epsilon$;
return $Q_j, \bar{x}$;

for solving a problem

$$\begin{aligned} \min \quad & \frac{c^T x}{d^T x}; \\ \text{s.t.:} \quad & Ax \leq b; \\ & x \in conv(\mathcal{X}); \end{aligned} \tag{6.9}$$

yields a set of feasible solutions as a side result. However, in principle, other (and distinct) methods may be used for solving (6.9) and finding $x \in \mathcal{X}$; therefore, when reasonable, we prefer to emphasize this in the pseudocodes of our algorithms. Algorithm 3 terminates after a finite number of steps. This is obvious, since $\pi(Q_j)$ reduces either through the reduction of $U(Q_j)$, or the increase in $L(Q_j)$. Both $U(Q_j)$ and $L(Q_j)$ are bounded; the latter from above by $\inf\{f(x) : x \in conv(\mathcal{X}))\}$, and the former by $\inf\{f(x) : x \in \mathcal{X})\}$, and so a finite improvement by some $\eta > 0$ can only happen for a finite number of times. Note that we do not want to make an impression that,

given $\epsilon = 0$, $\arg\min\{f(x) : x \in conv(\mathcal{X})\}$ belongs to the boundary of $Q_j, j \to \infty$. It *may* happen so, but in general the algorithm stalls as soon as for each $i = 1, \ldots, k$ $\exists \arg\min\{y_i : y \in \mathcal{Y} \cap Q_j\} \in Q_{j+1}$.

Let $(Q, \bar{x})$ be the result of Algorithm 3, $P = conv(\mathcal{X}) \cap C(Q)$ and $f(\bar{x}) = \bar{u}$. It may be possible to reduce Q further. It is easy to see that the upper bound $U_r(Q)$ for the ratio r, as suggested by Falk and Palocsay in [15] and calculated in line 7 of Algorithm 3 is nothing but the result of summation of the inequalities

$$\begin{aligned} \frac{a_r^T x}{b_r^T x} + \sum_{\substack{i=1 \\ i \neq r}}^{k} \frac{a_i^T x}{b_i^T x} &\leq \bar{u} \\ -\frac{a_i^T x}{b_i^T x} &\leq -\inf_{x \in P} \frac{a_i^T x}{b_i^T x} \quad i = 1, \ldots, k, i \neq r. \end{aligned} \tag{6.10}$$

Returning to the results of the previous section, let us denote by $\underline{\phi}_i^r(P)$ the bound $\underline{\phi}_i$ that is calculated over P *as if* the ratio r has been excluded from the objective, but all constraints of $C(Q)$ regarding this ratio are still in place. Then the sum of right-hand sides of the inequalities in the second line of (6.10) above gives exactly $\underline{\phi}_0^r(Q)$. The computational results in the prevous section of this chapter suggest that, if $k > 2$, we are likely to substantially improve the upper bounds on each ratio in P by setting

$$U_r(Q) = \bar{u} - \max_{0 \leq i \leq 3} \underline{\phi}_i^r(P) \quad r = 1, \ldots, k. \tag{6.11}$$

For this, however, we have to assume that the ratios $1, \ldots, k$ are all unisignant in P. If that is not the case, than we have to partition P appropriately first, and then apply this approach. Hence, the unisignance assumption does not affect generality. Also note that improvement of the bounds via the above approach calls for an iterative procedure. Indeed, once $U(Q)$ is reduced according to (6.11), $L(Q)$ may be improved again by minimizing individual ratios in the reduced P, possibly leading to another improvement of $U(Q)$. Then the bounds may be recalculated, and (6.11) applied

again, and so until no substantial improvement can be achieved.

It is relevant now to mention a remark about the rational choice of the ratio index r to use in Algorithm 2 from section VI.1. After application of (6.11) to Q it may no longer be cubic, and therefore the choice $r = \arg\min_{1\leq i\leq k} \pi_i(Q)$ may indeed reduce the computational effort to approximate the boundary of $\mathcal{Y} \cap Q$.

Further reduction of the initial search region is still possible. Let us apply our underestimators of $f(x)$ in yet another way. Since $\phi_1, \ldots, \phi_3$ achieve their minima over P in potentially distinct points, it may be that all of them, or at least some of them, do not lie in the intersection of the lower level sets

$$D_i(\bar{u}) = \{x \in P : \phi_i(x) \leq \bar{u}\} \quad i = 1, 2, 3.$$

In such case adding the inequalities that describe these level sets explicitly to P may allow us to improve the bounds on the optimal values of the ratios further. This can be done by the following linear constraints. Below we keep the notation adopted in section VI.2. For ϕ_1 we have simply

$$\left(\sum_{i=1}^{k} \frac{a_i}{\beta_i}\right)^T x \leq \bar{u}. \tag{6.12}$$

For ϕ_2, unless I^+ or I^- is empty, the lower level sets are not necessarily convex, and therefore cannot be described by linear inequalities. However, $D_2(\bar{u})$ is contained in the polyheadral set defined by the following two constraints and the inequalities of P:

$$\begin{aligned} (\bar{\gamma} - (\bar{u} - \inf_{\xi\in P} \frac{\underline{\gamma}^T\xi}{\bar{\beta}^T\xi})\bar{\beta})^T x \leq 0; \\ (\underline{\gamma} - (\bar{u} - \inf_{\xi\in P} \frac{\bar{\gamma}^T\xi}{\underline{\beta}^T\xi})\underline{\beta})^T x \leq 0; \end{aligned} \tag{6.13}$$

where $\bar{\gamma}, \underline{\gamma} \in \mathbb{R}^n$ are

$$\bar{\gamma} = \sum_{i \in I^+} \inf_{\xi \in P} \frac{\bar{\beta}^T \xi}{b_i^T \xi} a_i \quad \text{and} \quad \underline{\gamma} = \sum_{i \in I^-} \sup_{\xi \in P} \frac{\underline{\beta}^T \xi}{b_i^T \xi} a_i.$$

If either I^- or I^+ is empty, then $D_2(\bar{u})$ can be represented exactly with the help of

$$(\bar{\gamma} - (\bar{u}\bar{\beta})^T)x \leq 0; \quad \text{or} \quad (\underline{\gamma} - (\bar{u}\underline{\beta})^T)x \leq 0, \tag{6.14}$$

respectively. To describe $D_3(\bar{u})$ we need to add the following k inequalities to P:

$$(\sum_{i=1}^{k} \alpha_{ji} a_i - \bar{u} b_j)^T x \leq 0 \quad j = 1, \ldots, k \tag{6.15}$$

where α_{ji} are as defined in the previous section of this chapter.

Taking into account the nature of coefficients α_{ji}, β_i, $\bar{u}$ and vectors $\bar{\beta}$, $\underline{\beta}$, $\bar{\gamma}$ and $\underline{\gamma}$, and the fact that for a positive polytope $P \subset \mathbb{R}^n$ and n-vectors a_1, a_2

$$a_1 \geq a_2 \Rightarrow \{x \in P : a_1^T x \leq 0\} \subseteq \{x \in P : a_2^T x \leq 0\},$$

it is easy to see that that the constraints (6.12-6.15) become stronger as P reduces. This can be performed iteratively, which is what our final procedure for tightening the initial image space rectangle is based upon. One final remark concerns (6.12): this inequality is different from (6.13)-(6.15) and the constraints that define $C(Q)$, because it is not conic. It may prove useful in conjunction with the conic constraints when tightening linear relaxation (6.9), because it is likely to cut off a large portion of $conv(\mathcal{X})$. We can derive a Chvatal-Gomory cut from (6.12), too, but it is hardly useful for large instances with highly fractional coefficients.

In the listing of Algorithm 4 we refer to the polytope described by (a subset of) constraints (6.12)-(6.15) as $B(Q) \subset \mathbb{R}^n$ to emphasize that the constraints depend on Q. They certainly depend on $conv(\mathcal{X})$ as well, but it is assumed to be a part

Algorithm 4

Require: $\mathcal{P} = (\mathcal{X}, a_1, \ldots, a_k, b_1, \ldots, b_k)$, $0 < \epsilon < 1$.
Ensure: $S = \{Q_1, \ldots, Q_p \subset \mathbb{R}^k\}$

Obtain $R \subset \mathbb{R}^k, \bar{x} \in \mathbb{R}^n$ from Algorithm 3;
Partition R into $R_1, \ldots, R_m$ so that all components of $y \in R^i$, $i = 1, \ldots, m$ are unisignant;
$\bar{u} \leftarrow f(\bar{x})$;
$S \leftarrow \emptyset$;
for all $R \in \{R^1, \ldots, R^m\}$ **do**
    $Q_0 \leftarrow R$;
    $j \leftarrow 0$;
    repeat
        Calculate $B(Q_j)$;
        $u_r \leftarrow \bar{u} - \max\limits_{0 \le i \le 3} \phi_i^r(conv(\mathcal{X}) \cap C(Q_j) \cap B(Q_j)) \quad r = 1, \ldots, k$;
        $Q'_j \leftarrow \{y \in Q_j : y \le u\}$;
        $l_r \leftarrow \inf\{a_r^T x / b_r^T x : x \in conv(\mathcal{X}) \cap C(Q'_j) \cap B(Q_j)\} \quad r = 1, \ldots, k$;
        $Q_{j+1} \leftarrow \{y \in Q'_j : l \le y\}$;
        $j \leftarrow j + 1$;
    until $Q_j = \emptyset$ or $\frac{\pi(Q_{j-1}) - \pi(Q_j)}{\pi(Q_j)} < \epsilon$;
    if $e^T L(Q_j) < \bar{u}$ **then**
        $S \leftarrow S \cup \{Q_j\}$;
    end if
end for
return S;

of the problem description. Also, in Algorithm 4 for the most part we refer to the original problem space $\mathbb{R}^n$, since the constraints (6.12)-(6.15) cannot be described in the image space easily.

To evaluate efficacy of the suggested range reduction technique we performed computational experiments on the same batches of instances as in the previous section. We measure performance of Algorithms 3 and 4 based on how they reduce Q_0 computed by (6.2). The reported results include the average relative perimeter

Table 11 Optimal ratio range reduction results vs. k for MMRST instances

k	Algorithm 3		Algorithm 4			gap, %
	$\frac{\pi(Q)}{\pi(Q_0)}$,%	$\frac{vol(Q)}{vol(Q_0)}$,%	$\frac{\pi(Q)}{\pi(Q_0)}$,%	$\frac{vol(Q)}{vol(Q_0)}$,%	$\frac{\min \pi_r(Q)}{\pi_r(Q_0)}$, %	
2	0.17	<0.01	0.16	<0.01	0.08	5.65
3	82.38	57.13	30.13	5.21	19.76	33.39
5	93.60	73.72	34.81	2.20	27.61	39.34
10	95.61	64.69	36.57	0.04	32.85	44.12

reduction $\pi(Q)/\pi(Q_0)$, average relative volume reduction

$$\frac{vol(Q)}{vol(Q_0)}, \quad vol(Q) = \prod_{i=1}^{k}(U_i(Q) - L_i(Q));$$

and, for Algorithm 4, relative reduction of $\min_{1\le i\le k} \pi_i(Q)$. We also provide the average IP gap remaining after the range reduction. The results are provided in the same format as in the previos section, i.e., against the number of ratios and against the graph size. Tables 11 and 12 present the results for the MMRST instances.

The results suggest that Algorithm 3 performs well on average for the 2-ratio instances. It is thus not surprising that the improvement of Algorithm 4 on the results of Algorithm 3 for $k = 2$ is negligible. The reason is that the reduction (6.11) is not applicable in this case, so we can only expect improvement through application of the level set constraints (6.12)-(6.15). Even then it is hard to reduce Q further, since it is already small. However, performance of the first method deteriorates sharply when $k > 2$. For the second method the decrease of performance with k is much less noticeable in terms of all measures.

The same benchmark data, when looked upon against the size of the graph, suggests that Algorithm 4 does yield stable and significant reduction of the region containing optimal image points. The gap column here is worth of a separate remark:

Table 12 Optimal ratio range reduction results vs. graph size for MMRST instances

$\|V\|$	Algorithm 3		Algorithm 4			gap, %
	$\frac{\pi(Q)}{\pi(Q_0)}$,%	$\frac{vol(Q)}{vol(Q_0)}$,%	$\frac{\pi(Q)}{\pi(Q_0)}$,%	$\frac{vol(Q)}{vol(Q_0)}$,%	$\frac{\min \pi_r(Q)}{\pi_r(Q_0)}$, %	
10	73.18	64.15	47.85	10.77	37.11	27.94
20	72.02	59.66	34.21	3.65	26.36	30.14
30	68.01	48.38	29.27	1.59	22.98	29.82
50	68.03	52.53	26.42	0.93	20.88	30.77
70	70.52	58.26	24.32	0.71	19.18	30.36
100	66.04	43.23	21.93	0.41	17.46	31.42
200	65.62	44.48	19.74	0.23	15.82	31.51
300	64.49	35.96	17.26	0.12	13.96	31.16
400	66.22	42.27	17.01	0.09	13.81	31.61
500	65.23	39.91	16.14	0.08	13.13	31.47

stability of the average gap does not mean, as shown in Table 11, that we should expect a value of about 30% after applying Algorithm 4. It does suggest that, on average, in terms of this measure performance of the algorithm changes with k similarly for the tested graph sizes.

Tables 13 and 14 present the same results for the MMRP instances. The average performance trends are seen to be very similar. One additional thing worth mentioning is that for 2 ratios the (5,3) and (7,5) instance were solved to optimality via Algorithm 3. This is yet another empirical evidence of how close the boundaries of $conv(Y)$ and $\mathcal{Y}$ may be near the optimal image point for small k.

Note that although the output of Algorithm 3 serves as the input to Algorithm 4, this is done in order to (1) reduce the size of partition in the line 2 of the latter algorithm in the case of non-unisignant ratios and (2) to decrease run time of the

Table 13 Optimal ratio range reduction results vs. k for MMRP instances

k	Algorithm 3		Algorithm 4			gap, %
	$\frac{\pi(Q)}{\pi(Q_0)}$,%	$\frac{vol(Q)}{vol(Q_0)}$,%	$\frac{\pi(Q)}{\pi(Q_0)}$,%	$\frac{vol(Q)}{vol(Q_0)}$,%	$\frac{\min \pi_r(Q)}{\pi_r(Q_0)}$, %	
2	0.02	<0.01	0.01	<0.01	0.01	0.17
3	71.05	51.87	32.44	7.56	20.78	27.76
5	92.24	76.12	33.86	0.68	26.46	36.34
10	98.07	85.41	38.79	0.03	34.69	41.77

Table 14 Optimal ratio range reduction results vs. graph size for MMRP instances

(l, w)	Algorithm 3		Algorithm 4			gap, %
	$\frac{\pi(Q)}{\pi(Q_0)}$,%	$\frac{vol(Q)}{vol(Q_0)}$,%	$\frac{\pi(Q)}{\pi(Q_0)}$,%	$\frac{vol(Q)}{vol(Q_0)}$,%	$\frac{\min \pi_r(Q)}{\pi_r(Q_0)}$, %	
(5,3)	68.86	52.00	42.97	10.18	31.93	26.05
(7,5)	70.81	63.94	32.50	2.06	25.28	24.69
(9,7)	65.60	49.14	33.23	4.44	25.51	28.85
(12,10)	69.31	48.30	23.19	0.29	18.61	24.73
(14,12)	64.54	54.91	29.10	1.34	22.67	28.81
(17,15)	65.25	55.65	29.76	1.69	23.34	31.08
(20,18)	57.38	45.94	17.76	0.12	14.31	25.29
(22,20)	54.15	48.78	16.00	0.05	13.21	22.03
(24,22)	75.00	75.00	19.17	0.21	15.15	26.47
(27,25)	62.56	39.83	19.05	0.31	14.84	27.12

overall range reduction procedure, since, although Algorithm 4 can obviously achieve the same reduction on its own, it would have to perform possibly more iterations, each of which may, in fact, be more expensive that the first algorithm as a whole. Also, we have to admit that the constraints (6.15), as well as the underestimator ϕ_3 they

are based on, do not seem to perform well enough to justify their high computational cost.

VI.4. A global optimization algorithm

Finally, we suggest a global optimization algorithm for the considered class of FCOPs based on our findings in this chapter. The algorithm is essentially a branch-and-bound procedure applied to a partition S obtained from Algorithm 2 after applying it to an initial image rectangle Q.

During the initial stage we compute the starting partition S using Algorithm 4. Then we apply Algorithm 2 to each $Q \in S$. At this point, its purpose is not obtaining the approximation of $\mathcal{Y}$, but obtaining a good incumbent solution and the upper bound on the optimal value. Therefore, we use a different stopping criterion for Algorithm 2 during the initial stage: we stop building the approximation once the upper bound does not improve for a specified number of steps.

We perform a branch-and-bound in the main stage of the algorithm. With each node N we associate the lower bounf $\underline{f}(N)$, an image space partition $S(N)$ and two sets that contain the indices of the variables that are fixed at node N: $E_0 = \{i : x_i = 0\}$ and $E_1 = \{i : x_i = 1\}$. The branching strategy is different from the one adopted in Chapter V for solving CMRST problems, because, since we perform a substantial amount of calculations at each node, the branching suggested by Aggarwal et al. [1] is extremely time consuming even for relatively small instances. Instead, we rely on the observation that, for combinatorial problems, the good solutions often have some structural similarity, i.e., they are comprised of the similar sets of *elements* (e.g., edges for spanning trees, arcs for directed paths). Therefore, as the algorithm proceeds, we collect the information about the encountered good solutions in the

vector $\omega \in \mathbb{R}^n$ such that $\omega = \sum_{x \in W} x$, where W is a set of incumbent solutions that yield good upper bounds. Then the branch-and-bound tree nodes N' and N'' can be obtained by determining $j \notin E(N)$ such that ω_j is maximum and setting

$$E_1(N') = E_1(N) \cup \{j\}, \quad E_0(N') = E_0(N);$$
$$E_1(N'') = E_1(N), \qquad E_0(N'') = E_0(N) \cup \{j\}.$$

The partitions $S(N')$ and $S(N'')$ can be obtained from $S(N)$ by applying Algorithm 4 to each $Q \in S(N)$ after fixing the variables appropriately. Once this is done, the lower bounds $\underline{f}(N')$ and $\underline{f}(N'')$ are computed by applying a fixed number of streps of a modified Algorithm 2 to $S(N')$ and $S(N'')$. This modification only involves using the lower bound $\max\{\underline{\phi}_0, \underline{\phi}_1, \underline{\phi}_2\}$ instead of $\underline{\phi}_0$, and therefore does not change the structure of Algorithm 2 in any significant way. The rest of the details concerning the suggested branch-and-bound approach is clarified in the pseudocode of the algorithm.

To conclude this section, we would like to make another short remark regarding the variable fixing process, since the pseudocode of Algorithm 5 does not capture these details. Unless solving the constrained minimum ratio problems is done via linear programming techniques, which is impractical due to large sizes of such LPs, fixing the variables of a characteristic vector $x \in \mathbb{R}^n$ to 0 or 1 requires forcing the corresponding elements of the sought combinatorial objects out of, or, respectively, into the solution. For example, when we are looking for a spanning tree, fixing $x_i = 0$ means excluding the edge e_i from the graph. Forcing $x_i = 1$ could be achieved via either contracting the endpoints of e_i, or modifying the weight of an edge so that it is guaranteed to be included in any feasible solution. When we are looking for an optimal path in a directed acyclic graph $G = (V, A)$, fixing a variable to 1 is convenient to do in the following manner. Suppose the arc (v, u) is to be forced into

Algorithm 5

Require: $\mathcal{P} = (\mathcal{X}, a_1, \ldots, a_k, b_1, \ldots, b_k)$, $0 < \epsilon < 1$, m_1, m_2.

Ensure: $\bar{x}$ - the best incumbent solution found, $\underline{z}$ - the lower bound for the optimal objective value

Obtain S from Algorithm 4;
Run Algorithm 2 starting with $S^0 = S$ until the incumbent solution $\bar{x}$ does not improve for m_1 steps;
$S(N) \leftarrow S$, $E_0(N) \leftarrow \emptyset$, $E_1(N) \leftarrow \emptyset$, $\underline{f}(N) \leftarrow \underline{z}$;
$\omega \leftarrow \bar{x}$;
$\mathcal{N} \leftarrow \{N\}$;
while $\mathcal{N} \neq \emptyset$ **do**
    Choose $N \in \mathcal{N}$ with the smallest $\underline{f}(N)$;
    $\underline{z} \leftarrow \underline{f}(N)$;
    $\mathcal{N} \leftarrow \mathcal{N} \setminus \{N\}$;
    Choose j such that ω_j is maximum, $1 \leq j \leq n$, $j \notin E_0(N) \cup E_1(N)$;
    $E_1(N') \leftarrow E_1(N) \cup \{j\}$, $E_0(N') \leftarrow E_0(N)$;
    $E_1(N'') \leftarrow E_1(N)$, $E_0(N'') \leftarrow E_0(N) \cup \{j\}$;
    for $M \in \{N', N''\}$ **do**
        $x_i \leftarrow \xi_i \; \forall i \in E_\xi(M)$, $\xi = 0, 1$;
        $S(M) \leftarrow S(N)$;
        Obtain $S'(M)$ by applying Algorithm 4 to each $Q \in S(M)$;
        Obtain a lower bound $\underline{f}(M)$ and an incumbent $\bar{x}^M$ via running Algorithm 2 started with $S^0 = S'(M)$ for m_2 steps;
        if $|f(\bar{x}^M) - f(\bar{x})| \leq \epsilon |f(\bar{x})|$ **then**
            $\omega \leftarrow \omega + \bar{x}^M$;
            if $f(\bar{x}^M) < f(\bar{x})$ **then**
                $\bar{x} \leftarrow \bar{x}^M$;
            end if
        end if
        if $|f(\bar{x}) - \underline{f}(M)| > \epsilon |f(\bar{x})|$ **then**
            $\mathcal{N} \leftarrow \mathcal{N} \cup \{M\}$;
        end if
    end for
end while
return $\bar{x}, \underline{z}$;

the solution. Define

$$R = \{w \in V(G) : w \text{ is reachable from } u\}.$$

Then forcing (v, u) into the solution means deleting all arcs (p, q) such that $p \notin R$ and $q \in R$.

VI.5. Computational results

This section concludes the chapter with the computational results that assess the performance of the algorithms suggested in this chapter on MMRST and MMRP instances. In order to be consistent, we test the global optimization algorithm on the same batch of the instances that was used in numerical experiments in Chapters IV and V. Algorithm 5 is tested in two settings: with no approximation steps performed at a branch-and-bound tree node (denoted as Algorithm 5-0), and with 8 approximation steps performed at a branch-and-bound tree node (denoted as Algorithm 5-8). For both of these settings the target gap value is set to 1%, and computation time is limited to 1 hour. Along with the branch-and-bound scheme we provide the performance results of Algorithm 2 used as a heuristic in conjunction with the search space reduction technique described in Algorithm 4. It yields both the best feasible solution found, and the lower bound achieved from the approximation (hence the gap value). The targes IP gap value for Algorithm 2 is also set to 1%, but the runtime is limited to 10 minutes or 30000 iterations, whichever is reached sooner. In all cases the performance estimates are obtained by averaging the results for over 3 instances.

All algorithms are implemented in C++ using Microsoft Visual Studio 2003 environment[1]. We rely on the Boost Graph Library [38] implementation of adjacency

[1]The source code is available upon request

matrix to represent graphs. The experiments were performed on a computer with Intel® Core™ 2 Duo 3.16 GHz CPU and 3.23 GB of RAM.

Tables 15 and 16 summarize the results for the complete and sparse MMRST instances, respectively. It is evident that the performance of the branch-and-bound scheme suggested in this chapter, although comparable to the performance of the considered MIP models, is inferior to the algorithm from the previous chapter. The fact that it outperforms all other used methods (except for the heuristic) on the 2-ratio instances should not be attributed to the branch-and-bound scheme, but to the Algorithm 2-based heuristic instead, since it runs before the branch-and bound procedure starts, and almost always reaches the target gap - in fact, after the reduction step usually. This is clear since no branch-and-bound tree nodes were processed for almost all MMRST instances with $k = 2$. On the other hand, the performance of Algorithm 2, when used as a heuristic, is impressive on MMRST instances for all tested values of k. This, however, raises a legitimate question: Is it always true that $\mathcal{Y}$ approximates $conv(Y)$ accurately in the vicinity of an optimal solution?

Table 17 summarizes the results for the MMRP instances. Although the performance of the heuristic is still very good for this problem, it is clear that MMRP image polytopes of the considered instances may not be approximated by $calY$ as well as for the MMRST problem. This conclusion is suggested by the fact that for some smaller instances the heuristic reaches the iteration limit, but is unable to bridge the gap. Such difference in performance on MMRST and MMRP instances may be possibly explained by the structural difference of the solutions to these problems: a spanning tree always contains $|V - 1|$ edges, while an (s, t)-path may contain as few as 1 arc, or as many as l arcs for the considered instances.

On the other hand, the branch-and-bound scheme for the MMRP problem shows much stronger performance than for the MMRST, and, in particular, it by far outper-

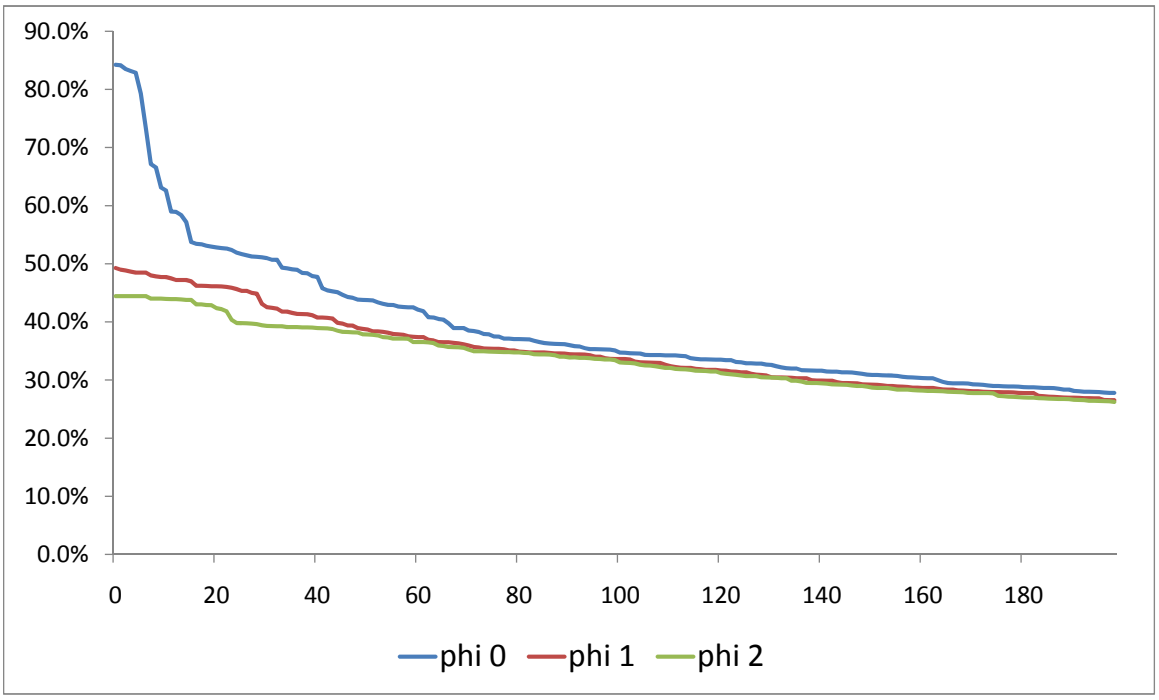


Fig. 6 IP gap progress in Algorithm 2 with different lower bounds used.

forms the considered MIP model. This is most likely due to the effectiveness of the branching rule suggested in the end of the previous section for the MMRP problem.

Another remark should be made regarding the approximation algorithm. Although the bounds $\underline{\phi}_1$, and $\underline{\phi}_2$ are typically much stronger than $\underline{\phi}_0$, they start to converge fast as the image space partition S derived by Algorithm 2 becomes more refined. This phenomenon is captured in Figure 6 for a complete MMRST instance with 30 vertices and 5 ratios. The same tendency is observed for the considered MMRP instances. Therefore, although these bounds are very helpful when reduction of the initial image rectangle is performed, using them in Algorithm 2 is not justified because of a much higher computational cost.

Table 15 Performance of Algorithm 2 and Algorithm 5 on complete MMRST instances

k	n	Algorithm 2			Algorithm 5-0			Algorithm 5-8		
		steps $\times 10^3$	gap (%)	run time (sec.)	nodes $\times 10^3$	gap (%)	run time (sec.)	nodes $\times 10^3$	gap (%)	run time (sec.)
2	10	0.0	0.9	0.0	0.0	0.9	0.0	0.0	0.9	0.0
	15	0.0	0.9	0.0	0.0	0.9	0.0	0.0	0.9	0.0
	20	0.0	0.8	0.7	0.0	0.8	0.7	0.0	0.8	1.0
	30	0.0	1.0	2.0	0.0	1.0	2.0	0.0	1.0	2.3
	40	0.0	0.9	3.3	0.0	0.9	3.3	0.0	0.9	4.0
	50	0.0	0.8	4.7	0.0	0.8	4.7	0.0	0.7	10.7
	80	0.0	0.7	33.0	0.0	0.7	33.0	0.0	0.8	55.7
	100	0.0	1.0	70.3	0.0	1.0	70.3	0.3	0.6	42.0
3	10	0.2	1.0	0.0	0.6	0.0	3.7	0.6	0.2	21.0
	15	0.2	1.0	1.0	70.0	0.0	644.0	17.2	1.4	1780.0
	20	0.4	1.0	2.7	148.3	22.4	3600.0	24.9	5.5	3600.0
	30	0.6	1.0	10.3	79.4	18.3	3600.0	13.0	14.5	3600.0
	40	0.5	1.0	13.0	55.2	18.7	3600.0	6.4	19.9	3600.0
	50	0.5	1.0	18.7	35.4	22.9	3600.0	3.9	21.9	3600.0
4	10	3.1	1.0	16.0	14.5	0.0	99.0	22.7	8.3	2491.7
	15	6.0	1.0	49.3	197.5	32.6	3600.0	24.3	10.5	2798.6
	20	11.9	1.0	148.0	160.5	16.0	3600.0	19.6	16.5	3600.0
	30	13.7	1.0	335.7	88.2	21.7	3600.0	9.7	23.5	3600.0
5	10	40.7	1.0	304.0	28.4	0.00	1945.7	22.5	8.6	2563.6
	15	49.0	1.1	600.0	23.3	16.5	3600.0	23.7	11.47	2982.0
	20	34.6	1.8	600.0	12.8	21.4	3600.0	16.6	20.3	3600.0

Table 16 Performance of Algorithm 2 and Algorithm 5 on sparse MMRST instances

k	n	Algorithm 2			Algorithm 5-0			Algorithm 5-8		
		steps $\times 10^3$	gap (%)	run time (sec.)	nodes $\times 10^3$	gap (%)	run time (sec.)	nodes $\times 10^3$	gap (%)	run time (sec.)
2	20	0.0	0.8	0.0	0.0	0.8	0.0	0.0	0.8	0.0
	40	0.0	0.9	0.0	0.0	0.9	0.0	0.0	0.9	0.0
	60	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0
	80	0.0	1.0	0.7	0.0	1.0	0.7	0.0	1.0	0.7
	100	0.0	1.0	5.0	0.0	1.0	5.0	0.0	1.0	5.0
	120	0.0	1.0	7.0	0.0	1.0	7.0	0.0	1.0	7.0
	140	0.0	1.0	18.0	0.0	1.0	18.0	0.0	1.0	18.0
	160	0.0	1.0	13.3	0.0	1.0	13.3	0.0	1.0	13.3
3	20	0.0	1.0	0.0	0.2	0.1	1.7	0.1	0.7	3.3
	40	0.0	1.0	0.0	109.5	1.2	1381.7	8.8	0.6	1113.0
	60	0.1	1.0	2.0	103.6	13.1	3600.7	11.5	7.5	3599.7
	80	0.2	1.0	4.0	62.2	19.1	3600.0	6.8	10.9	3600.0
	100	0.3	1.0	10.3	41.2	24.1	3600.0	4.1	14.8	3600.0
4	20	0.1	1.0	0.3	0.5	0.3	4.0	0.2	0.6	16.3
	40	0.6	1.0	5.7	81.6	1.0	1219.0	8.7	2.0	1283.0
	60	1.0	1.0	21.7	96.8	13.3	3600.0	9.1	13.0	3600.0
	80	2.8	1.0	103.7	58.6	22.2	3600.0	5.3	19.4	3600.0
5	20	1.3	1.0	7.7	0.9	0.5	9.3	0.8	0.4	49.0
	40	2.6	1.0	35.3	85.8	1.6	2107.3	22.8	2.3	3600.0
	60	9.5	1.0	284.0	73.1	14.8	3600.0	8.7	15.4	3600.0

Table 17 Performance of Algorithm 2 and Algorithm 5 on MMRP instances

k	(l, w)	Algorithm 2			Algorithm 5-0			Algorithm 5-8		
		steps $\times 10^3$	gap (%)	run time (sec.)	nodes $\times 10^3$	gap (%)	run time (sec.)	nodes $\times 10^3$	gap (%)	run time (sec.)
2	(5,3)	20.0	3.4	177.7	0.0	0.0	0.0	0.0	0.7	0.3
	(7,5)	10.0	0.8	117.3	0.0	0.1	0.7	0.0	1.0	2.3
	(9,7)	0.0	0.7	0.0	0.0	0.7	0.0	0.0	1.0	8.7
	(12,10)	4.3	1.4	202.7	0.0	0.5	12.3	0.0	1.0	25.3
	(14,12)	0.0	0.8	4.0	0.0	0.8	3.0	0.0	1.0	112.0
3	(5,3)	0.1	0.8	0.3	0.0	0.2	0.0	34.8	1.0	2287.3
	(7,5)	10.1	3.3	149.0	0.1	0.0	2.3	10.1	20.5	3600.0
	(9,7)	12.6	2.1	401.3	0.0	0.3	8.7	3.2	25.1	3600.0
	(12,10)	0.2	1.0	23.0	0.2	0.0	64.7	0.8	27.6	3600.0
	(14,12)	0.4	1.0	96.7	0.2	0.3	187.3	0.4	28.6	3600.0
4	(5,3)	20.7	2.1	222.0	0.0	0.3	1.0	31.2	11.3	3599.7
	(7,5)	1.2	1.0	21.7	0.1	0.4	8.0	7.3	25.4	3600.0
	(9,7)	3.3	1.0	151.0	0.9	0.0	70.0	2.1	29.5	3600.0
	(12,10)	4.0	2.3	600.0	2.4	0.6	641.7	0.5	31.9	3600.0
	(14,12)	1.8	3.1	595.7	5.9	2.2	3312.7	0.2	32.7	3600.0
5	(5,3)	21.1	1.8	262.0	0.1	0.0	1.3	32.9	13.8	3600.0
	(7,5)	12.8	1.0	293.7	0.3	0.1	14.0	6.2	28.6	3600.0
	(9,7)	7.5	2.6	449.3	1.4	0.0	129.7	2.0	33.3	3600.0
	(12,10)	2.6	7.1	600.0	9.3	2.7	2684.7	0.4	34.7	3600.0
	(14,12)	1.2	11.1	600.0	5.0	6.4	3600.0	0.2	35.7	3600.0

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

This dissertations considers a subclass of sum-of-ratios FCOPs whose linear versions admit polynomial-time exact algorithms. Since the objective function involved in such FCOPs is, in general, multiextremal, it is expected that these discrete problems are hard to solve. We formally establish this result in Chapter III for the MMRP and MMRC problems.

Since in this work we are primarily interested in developing solution techniques, it is logical to try the most generic and traditional MIP approach first. Unfortunally, the combinatorial problems do not typically lend themselves for compact linear MIP formulations. Furthermore, the available techniques for linearizing the sum-of-ratios objective function further substantially enlarges the models, both in terms of constraints and variables. It is not surprizing therefore that the computational experiments confirm that even a state-of-the-art MIP software such as CPLEX is only able to solve very small instances in reasonable time.

When generic solution methodologies do not give satisfactory results, it is natural to use methods that exploit properties of a specific problem. One such global optimization approach for the MMRST is developed in Chapter V. It is based on the existing algorithm by Sciskim and Palocsay [39] for this problem that addresses a special case of $k = 2$ ratios, and can be extended to other FCOPs relatively easily. The developed global optimization procedure shows consistently better performance on denser and larger instances than the linear mixed 0–1 formulations. However, in order to guarantee a near-optimal solution in reasonable time for large scale instances and large number of ratios in the objective, this approach needs substantial

improvement.

Relying on an empiric observation that an image $\mathcal{Y} \subset \mathbb{R}^k$ of a convex hull $\mathcal{X} \subset \mathbb{R}^n$ of the feasible solutions gives a good approximation of a convex hull $conv(Y) \subset \mathbb{R}^k$ of their images, we develop a generic algorithm that attempts to exploit good lower bounds obtained from such approximation in order to solve FCOPs via branch-and-bound. This approach is used in conjunction with the underestimators ϕ_{1-3} of the sum-of-ratios objective function that we develop in Chapter VI. The lower bounds $\underline{\phi}_{1-3}$ given by these underestimators are more expensive to compute than the simplistic lower bound $\underline{\phi}_0$ used by other authors, but typically improve upon it significantly. Based on these findings we develop a technique for reduction of the ranges of values that the individual ratios in the objective take at an optimal solution. Computational experiments show that for $k > 0$ this technique reduces the search space significantly compared to a simple techniques used in [15, 16] and [39]. However, the overall performance of the global optimization method described in Algorithm 5, which uses both new bounds and the suggested reduction technique, is discouraging: its results on the MMRST instances show that it is clearly inferior to Algorithm 1 developed in Chapter V.

On the other hand, a good approximation of $conv(Y)$ given by $\mathcal{Y}$ can be used in a different way. Since in the process of computing the approximation via Algorithm 2 we obtain a collection of feasible solutions, the approximation algorithm itself may be used as a heuristic approach for solving FCOPs. Such approach certainly may not *guarantee* that the problem will be solved to optimality, but since it yield the lower bound on the objective values as well, the quality of a solution, once it is obtained, is known immediately. For *all* generated instances this heuristic, in conjunction with the search reduction technique, by far outperforms the developed global optimization algorithms and the MIP models.

Several research directions are evident from the results of the dissertation. It is interesting to rigorously determine *how* well $conv(Y)$ is represented via $\mathcal{Y}$, and attempt assess the quality of such representation in different cases. A related direction is to study the case $k = 2$ more closely in order to establish the properties of the problem that make it so much easier to solve in the case $k = 2$. Also numerous improvements can be made to Algorithm 5. For example, the fact that it uses the bounds that are typically better than the bounds used in the branching process by Algorithm 1 suggests that the branching strategy used by the latter algorithm is superior. It is logical to adapt it for Algorithm 5.

REFERENCES

1. Aggarwal, V., Aneja, Y., Nair, K.: Minimal spanning tree subject to a side constraint. Computers and Operations Research **9**, 287–296 (1982)

2. Ahuja, R., Magnanti, T., Orlin, J.: Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Englewood Cliffs, N.J. (1993)

3. Arkin, E., Chiang, Y.J., Held, M., Michael, J., Sacristan, S., Skiena, S., Yang, T.C.: On minimum area hulls. Algorithmica **21**, 119–136 (1998)

4. Bitran, G., Magnanti, T.: Duality and sensitivity analysis for fractional programs. Operations Research **24**, 675–699 (1976)

5. Boros, E., Hammer, P.: Pseudo-Boolean optimization. Discrete Applied Mathematics **123**, 155–225 (2002)

6. Busygin, S., Prokopyev, O., Pardalos, P.: Feature selection for consistent biclustering via fractional 0–1 programming. Journal of Combinatorial Optimization **10**, 7–21 (2005)

7. Chandrasekaran, R.: Ratio spanning trees. Networks **7**, 335–342 (1977)

8. Chandrasekaran, R., Aneja, Y., Nair, K.: Minimal cost reliability ratio spanning tree. Ann. Discrete Math. **11**, 53–60 (1981)

9. Chandrasekaran, R., Tamir, A.: Polynomial testing of the query "is $a^b \geq c^d$?" with application to finding a minimal cost reliability ratio spanning tree. Discrete Applied Mathemtaics **9**, 117–123 (1984)

10. Chen, D., Daescu, O., Dai, Y., Katoh, N., Wu, X., Xu, J.: Optimizing the sum of linear fractional functions and applications. In: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 707–716. ACM, New York (2000)

11. Desrochers, M., Laporte, G.: Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. Operations Research Letters **10**, 27–36 (1991)

12. Dinkelbach, W.: On nonlinear fractional programming. Management Science **13**, 492–498 (1967)

13. Dur, M., Horst, R., Thoai, N.V.: Solving sum-of-ratios fractional programs using efficient points. Optimization **49**, 447–466 (2001)

14. Elhedhli, S.: Exact solution of a class of nonlinear knapsack problems. Operations Research Letters **33**, 615–624 (2005)

15. Falk, J., Palocsay, S.: Optimizing the sum of linear fractional functions. In: C. Floudas, P. Pardalos (eds.) Recent Advances in Global Optimization, pp. 221–258. Princeton University Press, Princeton, NJ (1992)

16. Falk, J., Palocsay, S.: Image space analysis of generalized fractional programs. Journal of Global Optimization **4**, 63–88 (1994)

17. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, San Francisco (1979)

18. Gol'stein, E.: Dual problems of convex and fractionally-convex programming in functional spaces. Soviet Math. Dokl. **8**, 212–216 (1967)

19. Handler, G., Zhang, I.: A dual algorithm for the constrained shortest path problem. Networks **10**, 293–310 (1980)

20. Hansen, P., de Aragão, M.P., Ribeiro, C.: Hyperbolic 0–1 programming and query optimization in information retrieval. Mathematical Programming **52**, 256–263 (1991)

21. Jungnickel, D.: Graphs, Networks and Algorithms. Springer, Berlin; Heidelberg; New York (2007)

22. Konno, H., Inori, M.: Bond portfolio optimization by bilinear fractional programming. Journal of the Operations Research Soceity of Japan **32**, 143–158 (1989)

23. Magnanti, T., Wolsey, L.: Optimal trees. In: M.B. et al. (ed.) Network Models, Handbooks in Operations Research and Management Science, vol. 7, pp. 503–615. Elsevier, Amsterdam; New York (1995)

24. Majhi, J., Janardan, R., Schwerdt, J., Smid, M., Gupta, P.: Minimizing support structures and trapped areas in two-dimensional layered manufacturing. Computational Geometry **12**, 241–267 (1999)

25. Majhi, J., Janardan, R., Smid, M., Gupta, P.: On some geometric optimization problems in layered manufacturing. Computational Geometry **12**, 219–239 (1999)

26. Matsumoto, M., Nishimura, T.: Mersenne Twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation **8**, 3–30 (1998)

27. Maurer, J.: The boost random number library . http://www.boost.org/doc/libs/1_38_0/libs/random/index.html. Accessed 29 April 2009

28. Megiddo, N.: Combinatorial optimization with rational objective functions. Mathematics of Operations Research **4**, 414–424 (1979)

29. Miller, L., Tucker, A., Zemlin, R.: Integer programming formulations of traveling salesman problems. Journal of the ACM **7**, 326–329 (1960)

30. Nesterov, Y.: Introductory Lectures on Convex Optimization: A Basic Course. Kluwer, Norwell, MA (2004)

31. Prokopyev, O.: Fractional zero-one programming. In: C. Floudas, P. Pardalos (eds.) Encyclopedia of Optimization, pp. 1091–1094. Springer, New York (2009)

32. Prokopyev, O., Huang, H.Z., Pardalos, P.: On complexity of unconstrained hyperbolic 0–1 programming problems. Operations Research Letters **33**, 312–318 (2005)

33. Prokopyev, O., Meneses, C., Oliveira, C., Pardalos, P.: On multiple-ratio hyperbolic 0–1 programming problems. Pacific Journal of Optimization **1**, 327–345 (2005)

34. Radzik, T.: Parametric flows, weighted means of cuts, and fractional combinatorial optimization. In: P. Pardalos (ed.) Complexity in Numerical Optimization, pp. 351–386. World Scientific, Singapore (1993)

35. Radzik, T.: Fractional combinatorial optimization. In: C. Floudas, P. Pardalos (eds.) Encyclopedia of Optimization, vol. 2, pp. 159–161. Kluwer Academic Publishers (2001)

36. Schaible, S.: Duality in fractional programming: A unified approach. Operations Research **24**, 452–461 (1976)

37. Schaible, S., Shi, J.: Fractional programming: The sum-of-ratios case. Optimization Methods and Software **18**, 219–229 (2003)

38. Siek, J., Lee, L.Q.: The boost graph library . http://www.boost.org/doc/libs/1_38_0/libs/graph/doc/index.html. Accessed 29 April 2009

39. Skiscim, C., Palocsay, S.: Minimum spanning trees with sums of ratios. Journal of Global Optimization **19**, 103–120 (2001)

40. Skiscim, C., Palocsay, S.: The complexity of minimum ratio spanning tree problems. Journal of Global Optimization **30**, 335–346 (2004)

41. Tawarmalani, M., Ahmed, S., Sahinidis, N.: Global optimization of 0–1 hyperbolic programs. Journal of Global Optimization **24**, 385–416 (2002)

42. Wu, T.H.: A note on a global approach for general 0–1 fractional programming. European Journal of Operations Research **101**, 220–223 (1997)

VITA

Oleksii Ursulenko was born in Kyiv, Ukraine in 1979. He received his Bachelor of Science and Master of Science degrees in computer engineering from National Technical University of Ukraine (former Kyiv Polytechnical Institute) in 2000 and 2002, respectively. He started the doctoral program in industrial and systems engineering at Texas A&M University in August 2004 and received his Doctor of Philosophy degree in December 2009. His research interests are in computational optimization, with emphasis on combinatorial problems, especially those with datamining applications in financial, social, and communication networks. He joined the Assistance Platform team of Business Division at Microsoft Corporation in October 2009.

Mr. Ursulenko may be reached at his work address:

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

This document was typeset in LaTeX by Oleksii Ursulenko.