Optimization with interval data: new problems, algorithms, and applications.

Mohsen Mohammadi Dehcheshmeh
University of Louisville

OPTIMIZATION WITH INTERVAL DATA: NEW PROBLEMS, ALGORITHMS, AND APPLICATIONS

Mohsen Mohammadi
M.S., Industrial Engineering, Azad University, Iran, 2013
B.S., Industrial Engineering, Azad University, Iran, 2010

A Dissertation Submitted to the Faculty of
the J.B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy in Industrial Engineering

Department of Industrial Engineering
University of Louisville
Louisville, Kentucky

May 2021

OPTIMIZATION WITH INTERVAL DATA: NEW PROBLEMS, ALGORITHMS, AND APPLICATIONS

Mohsen Mohammadi
M.S., Industrial Engineering, Azad University, Iran, 2013
B.S., Industrial Engineering, Azad University, Iran, 2010

A Dissertation submitted on

April 28, 2021

To the following Dissertation Committee

ACKNOWLEDGMENTS

First and foremost, I wish to express my wholehearted gratitude to Dr. Monica Gentili, my committee chair and advisor, who not only patiently guided me through every step of my Ph.D. journey but also stood by me at moments of frustration and disappointment. I feel so grateful that as I will pursue my career in academia, I will always have a perfect example of a mentor, a scientist, and a wonderful person, to look up to.

I would like to thank my committee members: Dr. Lihui Bai, Dr. Csaba Biro, Dr. Gail W. DePuy, and Dr. John S. Usher, for examining my work and providing me with insightful feedback. Special thanks go to Dr. Milan Hladík from Charles University for his generous help throughout the study and for his research contributions. It was my great honor to collaborate with and be taught by such a high-caliber scholar. I also thank Dr. Raffaele Cerulli from the University of Salerno for his contribution to the fourth chapter of my dissertation. For this, I am grateful. Finally, this research would not have been possible without the financial support from the School of Interdisciplinary and Graduate Studies and the Department of Industrial Engineering at the University of Louisville.

Getting a Ph.D. is a long and challenging journey. I have been extremely fortunate to have a loving family to support me along the way. I am grateful for the love and encouragement that my parents, Houshang and Hamdam, have always given me through any endeavor I have undertaken. Finally, I also would like to thank my three sisters for their support in this process.

ABSTRACT

OPTIMIZATION WITH INTERVAL DATA: NEW PROBLEMS, ALGORITHMS, AND APPLICATIONS

Mohsen Mohammadi

April 28, 2021

The parameters of real-world optimization problems are often uncertain due to the failure of exact estimation of data entries. Throughout the years, several approaches have been developed to cope with uncertainty in the input parameters of optimization problems, such as robust optimization, stochastic optimization, fuzzy programming, parametric programming, and interval optimization. Each of these approaches tackles the uncertainty in the input data with different assumptions on the source of uncertainty and imposes different requirements on the returned solutions. In this dissertation, the approach we take is that of interval optimization, and more specifically, interval linear programming. The two main problems we consider in this context are, considering all realizations of the interval data, the problems of finding the range of the optimal values and determining the set of all possible optimal solutions. While the theoretical aspects of these problems are well-studied, the algorithmic aspects and the engineering implications have not been explored. In this dissertation, we partially fill these voids. In the first part of the dissertation, we present and test three heuristics to find bounds on the worst optimal value of the equality-constrained interval linear program, which is known to be an intractable problem. In the second part of the dissertation, motivated by a real-case problem in the healthcare context, we define and analyze a new problem, the outcome range problem, in interval linear programming.

The solution to the problem would help decision-makers quantify unintended/further consequences of optimal decisions made under uncertainty. Basically, the problem finds the range of an extra function of interest (different from the objective function) over all possible optimal solutions of an interval linear program. We analyze the problem from the theoretical and algorithmic perspectives. We evaluate the performance of our algorithms on simulated problem instances and on a real-world healthcare application. In the third part of the dissertation, we extend our analysis of the outcome range problem, exploring different theoretical properties and designing several new solution algorithms. We also test our solution approaches on different datasets, highlighting the strengths and weaknesses of each approach. Finally, in the last part of the dissertation, we discuss an application of interval optimization in the sensor location problem in the traffic management context. Particularly, we propose an optimization approach to handle the measurement errors in the full link flow observability problem. We show the applicability of our approach on several real-world traffic networks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background and contributions

Undoubtedly, linear programming (LP) is known to be one of the most widely used optimization tools to formulate and solve practical problems [88]. A linear programming problem indeed is very simple to construct. In practice, however, it is not always possible to estimate input parameters of a linear program accurately. This may impair results of the problem and consequently inferences made upon them. The need to consider uncertainty in input parameters of a linear program has been a subject of interest since the very first work in this area by Dantzig in mid 1950s [35], where uncertainty in input parameters were modeled by considering different scenarios occurring with different probabilities. Since then, several approaches have been developed to deal with data uncertainty in optimization problems, such as robust optimization (see, e.g., [4, 6, 7, 10]), stochastic optimization (see textbooks [14, 70, 76, 108]), fuzzy optimization (see [86, 71, 109] and references therein), parametric programming [41, 42], and interval linear programming [58, 111, 113]. Each of the approaches bears its own advantages and limitations. In general, techniques developed to address uncertainty in optimization problems differ with respect to (i) the source of uncertainty, and (ii) requirements on the returned solution. In this dissertation, we address the uncertainty in the input parameters by the interval linear programming approach which assumes that parameters can vary independently within given upper and lower bounds without any further assumptions. In this sense, interval linear programming (ILP) investigates overall properties of an interval-valued linear program considering all the possible realizations of the interval data. Two main problems are subject of study in interval linear programming:

- the *optimal value range problem:* Determining the range of all optimal values of the objective function computed over all realizations of the interval data [30, 57, 111, 97];

- the *optimal solution set problem:* Describing the set of all possible optimal solutions resulting from all the scenarios of the interval input parameters [2, 44, 59, 61, 65].

The first problem can become computationally expensive depending on whether constraints are in the equality form or variables are unrestricted. The second problem is in general more computationally demanding as the optimal solution set of an interval linear program can be nonconvex or even disconnected. This dissertation contributes to the analysis of the above mentioned problems, introduces a new problem in this context, and also explores a new application of interval optimization as described below:

- In general, there are very few studies developing approximation methods for intractable cases in the optimal value range problem. We design several heuristic algorithms to approximate one of the intractable cases of the problem, that is, the worst optimal value of the equality-constrained interval linear program;

- Although basic properties related to the optimal solution set problem have been addressed in the literature, different aspects and implications of the problem have not been explored. In this dissertation, we partially fill this gap by defining a novel problem, namely, the outcome range problem, which aims at assessing further impacts of optimal decision making under uncertainty. Specifically, the goal is to find lower and upper bounds on an additional linear function, modeled to describe the further impacts, over the optimal solution set of an interval linear program. We extensively investigate the problem both theoretically and algorithmically, and show its real-world implications through a real case study on healthcare access measurement;

- We explore a new application of interval optimization for the sensor location problem in the context of traffic management. In particular, we address the full link flow observability problem assuming errors in the monitored data (described by real intervals). To the best of our knowledge, our work in this dissertation would be among the first to explicitly address uncertainty in the monitored data in flow observability problems for traffic networks.

1.2 Structure of this dissertation

Chapters 2-5 of this proposal are four academic papers. In particular, the second and third chapters are published papers [93, 94], the fourth chapter is under revision [95], and the last chapter is in preparation for submission to a journal. The chapters are therefore self-contained, that is, each of them has its own abstract, introduction, conclusions, and notations. Next, we give an overview of each chapter.

In Chapter 2, we consider the equality-constrained interval linear program for which finding the worst optimal value is known to be NP-hard. We design three heuristics to compute bounds on the worst optimal value. In particular, we present a constructive greedy algorithm to determine a lower bound and two enclosure-based approaches to find an upper bound. We evaluate the methods on a set of randomly generated instances.

In Chapter 3, we formulate and study a new problem in the interval linear programming literature, namely, the *outcome range problem.* The problem intends to quantify further impacts of optimal decision making under uncertainty, which are modeled by a linear function referred to as outcome function. Particularly, the outcome range problem is the problem of finding the upper and lower bounds of a given outcome function over the set of all possible optimal solutions of an interval linear program. The problem often arises when decisions have differential impacts on different communities and sub-populations such as decisions made by government agencies, public health decision makers, policy makers, etc. In this chapter, we give a general definition of the problem and focus on a specific case of it where interval uncertainty only occurs in the right-hand side vector (we focused on inequality-constrained problems). For this specific case, we assess the computational complexity of the problem and study some of its theoretical foundations. We then design two heuristics for the problem, and we test them on two sets of randomly generated problem instances highlighting the performance of each algorithm. To show the relevance of our problem in practice and the efficiency of our algorithms, we provide a healthcare application of it.

In Chapter 4, we extend our works in Chapter 3 to look deeper into properties of the outcome range problem. We narrow down our study on the case of interval right-hand side vector for equality-constrained problems. We study the computational complexity of the problem, and we also explore theoretical properties related to some characteristics of the problem in the scenario space. Moreover, we formally discuss the relationship between the outcome range problem and some other known similar problems in the literature such as the optimal value range problem, multiobjective optimization, and bilevel optimization. We develop three heuristics to estimate the range of an outcome function. We test them on three different datasets and analyze their quality and efficiency.

Chapter 5 considers the problem of locating the minimum number of sensors on a subset of links of a traffic network to fully observe all link flows, namely, the full link flow observability problem. A common assumption in this problem is that the monitored data are error-free (i.e., no measurement error). In this chapter, we relax this assumption and consider the measurement errors as given intervals. We then present an optimization problem to cope with the variability in the solution caused by measurement errors and design a local search algorithm to solve the problem. Lastly, we test our method on five real traffic networks, showing the validity of our approach.

CHAPTER 2

BOUNDS ON THE WORST OPTIMAL VALUE IN INTERVAL LINEAR PROGRAMMING

2.1 Introduction

In real life, problems are subject to uncertainty due to inaccurate estimations or unexpected changes. One of the basic tools to describe uncertainty in a linear programming problem is interval linear programming (ILP), where we assume that there are a-priori known intervals within which parameters of a linear program can vary. Intervals are appropriate tools to represent uncertainty arising from, for example, measurement errors, missing data, rounding errors, and statistical estimations. They are very easy to establish as we only need to know two endpoints for each interval without any other additional assumption. Interval linear programming has applications in several areas, including portfolio selection problems [79], environmental management [28, 84], interval matrix games [82, 85], and transportation [27, 75, 129].

Different topics have been addressed in this area (see [58] for a recent survey). One of them is determining the optimal value range, that is, the problem of finding the best and the worst optimal values among all the optimal values obtained over all data perturbations. The optimal value range problem has been addressed in some papers in the literature; most of them propose exact algorithms for the problem [30, 57, 73, 97, 111]. Few contributions focus on getting approximated values (i.e., bounds) for some of the intractable cases of the problem [1, 62]. In this paper, we propose three approaches to determine lower and upper bounds to one of the intractable cases of the problem.

The paper is organized as follows. Next section gives a formal definition of the problem and reviews the existing contributions in the literature addressing it. Our proposed approaches are based on some existing theoretical results which are recalled in Section 2.3. Section 2.4 contains the details of our approaches. Our experimental results are shown in Section 2.5. Conclusions and further steps of the research are discussed in Section 2.6.

2.2 Problem description and existing results

An interval linear matrix is defined as

$$\mathbf{A} := [\underline{A}, \overline{A}] = \{A \in \mathbb{R}^{m \times n} : \ \underline{A} \leq A \leq \overline{A}\},$$

where $\underline{A}$ and $\overline{A}$ are given matrices. We define the mid-point (A_c) and the radius (A_Δ) of an interval matrix as $A_c := \frac{1}{2}(\overline{A} + \underline{A})$, and $A_\Delta := \frac{1}{2}(\overline{A} - \underline{A})$. Interval vectors can be defined analogously. Throughout we use bold symbols to show interval vectors and matrices. We focus on the following interval linear programming problem

$$z(\mathbf{A}, \mathbf{b}, \mathbf{c}) := \min \ \mathbf{c}^T x \ \text{ subject to } \ \mathbf{A}x = \mathbf{b}, \ x \geq 0. \tag{2.1}$$

where $x \in \mathbb{R}^n$ is the decision vector, $\mathbf{c}$ is an interval n-dimensional vector, $\mathbf{b}$ is an interval m-dimensional vector, and $\mathbf{A}$ is an interval matrix of the appropriate dimensions. We refer to any triple (A, b, c), $A \in \mathbf{A}$, $b \in \mathbf{b}$, and $c \in \mathbf{c}$, as a *scenario.*

With each scenario (A, b, c), we can associate a linear programming problem, namely LP(A, b, c), whose optimal value is denoted by $z(A, b, c)$:

$$z(A, b, c) = \min\ c^T x \ \text{ subject to }\ Ax = b,\ x \geq 0. \tag{2.2}$$

Hence, an ILP problem is a family of linear programming problems associated with all the possible scenarios. We denote by B an optimal basis of LP(A, b, c). Indeed, B is the set of indices of basic variables, and index set N similarly stands of for nonbasic variables. Subscript B on a matrix (vector) denotes its submatrix (subvector) composed of columns (elements) indexed by B; subscript N is defined analogously.

The optimal value range problem consists of determining the minimum and maximum optimal values of (2.1) obtained over all possible realizations of the interval data, that is, it consists of solving the two following problems

$$\underline{z} := \inf\{z(A, b, c) :\ A \in \mathbf{A},\ b \in \mathbf{b},\ c \in \mathbf{c}\}, \tag{2.3}$$

$$\overline{z} := \sup\{z(A, b, c) :\ A \in \mathbf{A},\ b \in \mathbf{b},\ c \in \mathbf{c}\}, \tag{2.4}$$

where $\underline{z}$ is said to be the *best* optimal value, and $\overline{z}$ is said to be the *worst* optimal value. It is worth noting that the computational complexity of the two problems varies depending on the structure of the underlying interval linear problem [58]. In particular, when the underlying interval linear program has equality constraints and nonnegative constraints on the variables, it is known that computing $\underline{z}$ is an easy task, while determining $\overline{z}$ is NP-hard [111]. Table 2.1 shows an overview of the computational complexity of the optimal value range problem for the three different types of interval linear programming problems studied in the literature. Few contributions focus on computing bounds for the intractable cases. Allahdadi and Golestane [1] applied Monte Carlo simulation to get a lower bound for the worst optimal value when the problem is of Type I. Hladík [62] computed bounds on the best optimal value when the problem is of Type II. In this paper, we propose three approaches to determine lower and upper bounds to the worst optimal value of the problem when the underlying interval linear program is of Type I. Note that the worst optimal value can be either finite, infeasible or unbounded. In the real-world context, we are generally interested in problems admitting a finite optimal value. Thus, in our analysis, we assume the underlying interval linear program always admits a finite worst optimal value.

Table 2.1: Summary of the computational complexity of the optimal value range problem [58].

	Type I $\mathbf{A}x = \mathbf{b},\ x \geq 0$	Type II $\mathbf{A}x \leq \mathbf{b}$	Type III $\mathbf{A}x \leq \mathbf{b},\ x \geq 0$
Best optimal value $\underline{z}$	Polynomial	NP-hard	Polynomial
Worst optimal value $\overline{z}$	NP-hard	Polynomial	Polynomial

2.3 Preliminaries

In this section, we recall some basic theorems and results which are used in our solution approaches. One of the basic topics in interval computation is solving interval systems of equations (inequalities). Consider the following interval system in a general form

$$\mathbf{A}x \overset{(\leq)}{=} \mathbf{b} \quad (x \geq 0),$$

where $\mathbf{b}$ is an interval m-dimensional vector, and $\mathbf{A}$ is an interval matrix of the appropriate dimensions. The solution set of a interval system of equations (inequalities) is the set of all possible solutions resulting from all scenarios of the interval parameters, that is,

$$\{x \in \mathbb{R}^n |\ \exists\, A \in \mathbf{A},\ \ \exists\, b \in \mathbf{b} : Ax \overset{(\leq)}{=} b\ \ (x \geq 0)\}.$$

Theorems 2.1-2.3 characterize the solution set of the three main types of interval system of linear equations/inequalities.

Theorem 2.1. *(see [105]) The solution set of* $\mathbf{A}x = \mathbf{b}$ *is described by*

$$|A_c x - b_c| \le A_\Delta |x| + b_\Delta. \tag{2.5}$$

Theorem 2.2. *(see [112]) The solution set of* $\mathbf{A}x = \mathbf{b}, \;\; x \ge 0$ *is described by*

$$\underline{A}x \le \overline{b}, \;\; -\overline{A}x \le -\underline{b}, \;\; x \ge 0. \tag{2.6}$$

Theorem 2.3. *(see [53]) The solution set of* $\mathbf{A}x \le \mathbf{b}$ *is described by*

$$A_c x - A_\Delta |x| \le \overline{b}. \tag{2.7}$$

Let us consider the i-th constraint of (2.1), i.e., $\mathbf{a}_i x = \mathbf{b}_i$. Let us consider the two *extremal* versions associated with it

$$\sum_j \overline{a}_{ij} x_j = \underline{b}_i, \quad \sum_j \underline{a}_{ij} x_j = \overline{b}_i.$$

We refer to a scenario where each constraint takes either of the above extremal versions as an *extremal scenario*. Extremal scenarios are important for our analysis because of the following theorem.

Theorem 2.4. *(see [30, 112]) The worst optimal value of (2.1) occurs on one of the extremal scenarios of the problem, that is,*

$$\overline{z} = \inf\{z(A_c - diag(p)A_\Delta, b_c + diag(p)b_\Delta, \overline{c}) : \;\; p \in \{\pm 1\}^m\}$$

where $diag(p)$ *is a diagonal matrix with entries* $p_1, \ldots, p_m$.

Note that the number of extremal scenarios is 2^m. Chinnek and Ramadan [30] described an algorithm which explores all the 2^m extremal scenarios to find the worst optimal value.

2.4 Finding lower and upper bounds for $\overline{z}$

In this section, we describe our proposed approaches to find an upper bound and a lower bound for $\overline{z}$, that is, finding two values $\overline{z}^U$ and $\overline{z}^L$ such that $\overline{z}^L \leq \overline{z} \leq \overline{z}^U$.

2.4.1 Finding an upper bound

We describe two approaches to determine an upper bound. Our first approach applies a procedure similar to the one used in [62] for an ILP problem with inequality constraints (Type II). It determines an upper bound using a basis B which is feasible for each scenario.

Definition 2.1. A basis B is strongly feasible for (2.1) if x_B is such that $x_B = A_B^{-1}b \geq 0$, for all $A_B \in \mathbf{A}_B,\ b \in \mathbf{b}$.

Initially, we determine a basis B which is feasible for any scenario of a given ILP problem. We can find such a basis by solving to optimality LP(A, b, c) for an arbitrary scenario. We consider then the following interval system of equations

$$\mathbf{A}_B x_B = \mathbf{b}, \quad x_N = 0. \tag{2.8}$$

To proceed further, system $A_B x_B = b$ needs to be solvable for all $A_B \in \mathbf{A_B},\ b \in \mathbf{b}$. That is, matrix $\mathbf{A}_B$ needs to be regular for each $A_B \in \mathbf{A}_B$. If this condition holds, we are interested in the following set

$$H = \{x_B \in \mathbb{R}^{|B|} \mid A_B x_B = b \ \text{ for some } \ A_B \in \mathbf{A}_B,\ b \in \mathbf{b}\}. \tag{2.9}$$

The set (2.9) is described by (2.5) and it is difficult to determine exactly. However, there exist several methods in the literature to find an interval enclosure of it, that is, an interval vector containing it [112]. Let $\mathbf{x}_B$ be such an interval enclosure. If the lower bounds are non negative, that is $(\underline{x}_B)_j \geq 0$ for all $j \in B$ (where $(\underline{x}_B)_j$ denotes the lower endpoint of the j-th component of $\mathbf{x}_B$), then B is strongly feasible. As a result, we can determine an upper bound by using the following proposition.

Proposition 1. *If a basis B is strongly feasible for (2.1) , then the following gives an upper bound on (2.4)*

$$\overline{z}^{U_1} = \max\ \overline{c}_B^T x_B \ \textit{ subject to } \ x_B \in \mathbf{x}_B, \ \ x_N = 0. \tag{2.10}$$

Proof. Let x'_B be a member of H for a given realization. Since B is strongly feasible, we have

$$\overline{c}_B^T x'_B \geq z(A, b, c) \quad \forall A \in \mathbf{A},\ b \in \mathbf{b},\ c \in \mathbf{c},$$

where $x_N = 0$. Thus, the following holds

$$\overline{z} \leq z' = \max\ \overline{c}_B^T x_B \ \text{ subject to } \ x_B \in H, \ \ x_N = 0.$$

We know that $H \subseteq \mathbf{x}_B$. Therefore, we can conclude

$$\overline{z} \leq z' \leq \overline{z}^{U_1}.$$

□

Note that problem (2.10) can be easily solved by applying interval arithmetic, that is,

$$\overline{z}^{U_1} = \overline{c}_B^T \hat{x}_B,$$

where $\hat{x}_B$ is such that $(\hat{x}_B)_j = (\overline{x}_B)_j$ if $(\overline{c}_B^T)_j \geq 0$, and $(\hat{x}_B)_j = (\underline{x}_B)_j$ otherwise. In other words, the j-th component of the vector $\hat{x}_B$ is equal to the upper endpoint of the j-th component of $\mathbf{x}_B$ if the j-th component of $\overline{c}_B$ is non-negative (i.e., $(\overline{c}_B^T)_j \geq 0$), while it is equal to the lower endpoint of the interval otherwise.

We apply this procedure using different initial starting bases and returning the minimum among all the obtained bounds. In our experiments, described in section 2.5, we used the Hansen-Bliek-Rohn method [99] for finding $\mathbf{x}_B$, and we tried 10 different initial bases. Algorithm 2.1 reports the pseudo-code of our approach.

Algorithm 2.1: Upper bound $\overline{z}^{U_1}$ on $\overline{z}$

```
1  Input: $\mathbf{A}, \mathbf{b}, \mathbf{c}$, stopping condition
   Result: an upper bound of $\underline{f}$
2  Set $k \leftarrow 0$.
3  while stopping condition is not met do
4      put $k \leftarrow k + 1$
5      choose a random scenario LP$(A, b, c)$ and determine an optimal basis $B$
6      if matrix $\mathbf{A}_B$ is regular then
7          find an interval enclosure $\mathbf{x}_B$ of the solution set of (2.8)
8      else
9          go to line 5
10     end
11     if $(\underline{x}_B)_j \geq 0$ for all $j \in B$ then
12         compute $\overline{z}_k^{U_1}$ by interval arithmetic
13     end
14 end
```

The second approach determines an upper bound for $\overline{z}$ by solving a linear optimization problem on an enclosure of the set of all optimal solutions of our problem. More in detail, let us consider the dual of (2.1), that is,

$$\max\ \mathbf{b}^T y \ \text{ subject to }\ \mathbf{A}^T y \leq \mathbf{c}. \tag{2.11}$$

By extending the strong duality concept in linear programming, we have the following interval system of linear inequalities

$$\mathbf{A}x = \mathbf{b}, \ \ x \geq 0, \quad \mathbf{A}^T y \leq \mathbf{c}, \ \ \mathbf{c}^T x = \mathbf{b}^T y. \tag{2.12}$$

There are dependencies between the interval coefficients in (2.12) (double occurrences of $\mathbf{A}, \mathbf{b}, \mathbf{c}$), and they are hard to deal with in general. By relaxing dependencies between coefficients, we obtain a super set containing all the optimal solutions of (2.1) and (2.11). By Theorems 2.1- 2.3, the solution set of (2.12) can be described by the following system of inequalities:

$$\underline{A}x \le \overline{b},\ -\overline{A}x \le -\underline{b},\ x \ge 0,\ A_c^T y - A_\Delta^T |y| \le \overline{c},\ |c_c^T x - b_c^T y| \le c_\Delta x + b_\Delta^T |y|. \quad (2.13)$$

By applying the approach in [59], we can determine an interval enclosure of (2.13) which contains then all the optimal solutions of our problem. Briefly, the algorithm in [59] is an iterative refinement algorithm. It first linearizes system (2.13), then it determines an enclosure of the solution set of the linearized system and contracts such an enclosure at each iteration until improvement is insignificant. It runs in polynomial time and it returns a sufficiently tight enclosure. The obtained enclosure, in fact, is a set including all the optimal solutions of (2.1) and also solutions that are not necessarily optimal to any scenario. Let $\mathbf{x}$ be such an enclosure. It is easy to show that the following proposition holds:

Proposition 2. *Let* $\mathbf{x}$ *be an enclosure of all the optimal solutions of (2.1), then an upper bound to (2.4) can be obtained by*

$$\overline{z}^{U_2} = \max\ \overline{c}^T x \ \ \textit{subject to}\ \ x \in \mathbf{x}. \quad (2.14)$$

Problem (2.14) can be solved by interval arithmetic, that is, taking the right endpoints of $\mathbf{c}^T\mathbf{x}$.

2.4.2 Finding a lower bound

We use a greedy approach to determine a lower bound to $\overline{z}$. The greedy approach is based on the fact that the scenario where the worst optimal value is achieved is an extremal scenario (as stated in Theorem 2.4). The main idea of the algorithm is to choose at each step, according to a greedy criterion, one of the extremal versions of one of the equality constraints of the system, and to stop when an extremal scenario is obtained. The bound is then computed by solving the linear program resulting from the extremal versions of constraints. The greedy criterion, at each step, compares the optimal values of two linear programs. Each program is associated with one of the extremal versions of the equality constraint under consideration, and it is defined such that its feasible set is obtained by considering: (i) an interval enclosure $\mathbf{x}$ of the set of all optimal solutions of (2.1), (ii) the solution set F described by (2.6), and (iii) one of the extremal versions associated with the constraint under consideration. Let us assume, without loss of generality, that during the first step, the constraint under consideration is the i-th constraint. We particularly consider the following linear programs

$$z_i^1 = \min\ \overline{c}^T x \ \text{ subject to } \ \sum_j \overline{a}_{ij} x_j = \underline{b}_i,\ x \in \mathbf{x} \cap F, \tag{2.15}$$

$$z_i^2 = \min\ \overline{c}^T x \ \text{ subject to } \ \sum_j \underline{a}_{ij} x_j = \overline{b}_i,\ x \in \mathbf{x} \cap F. \tag{2.16}$$

Problems (2.15) and (2.16) are associated with the two extremal versions of the i-th constraint. The greedy algorithm chooses the extremal version which is associated with the problem with the highest optimal value. Let us assume that $z_i^2 \geq z_i^1$, so that the chosen extremal version is $\sum_j \underline{a}_{ij} x_j = \overline{b}_i$. In the next step, the greedy algorithm considers another constraint of (2.1), say the k-th constraint, and the two following linear programs associated with each of its extremal versions

$$z_k^1 = \min\ \overline{c}^T x \ \text{ subject to } \ \sum_j \underline{a}_{ij} x_j = \overline{b}_i,\ \sum_j \overline{a}_{kj} x_j = \underline{b}_k,\ x \in \mathbf{x} \cap F, \tag{2.17}$$

$$z_k^2 = \min\ \overline{c}^T x \ \text{ subject to } \ \sum_j \underline{a}_{ij} x_j = \overline{b}_i,\ \sum_j \underline{a}_{kj} x_j = \overline{b}_k,\ x \in \mathbf{x} \cap F. \tag{2.18}$$

The algorithm will choose the extremal version of k-th constraint corresponding to the highest value between z_k^1 and z_k^2. The algorithm proceeds in this way, by adding the set of extremal versions of the constraints examined so far to the two linear programming problems to be considered at each step. It stops when an extremal scenario is obtained, that is, when an extremal version for each of the constraints is chosen. The returned lower bound $\overline{z}^L$ is the highest value between the optimal values of the two linear programs obtained by adding the last examined constraint. Note that the algorithm assumes an ordered sequence of the constraints to be defined at the beginning. Of course, with different sequences, the final value would be different. Hence, we run the greedy algorithm with different initial sequences of the constraints and consider the tightest bound among all. In our experiment, described in the next section, we generated 10 initial different sequences. Our proposed approach for finding a lower bound is described in Algorithm 2.2.

Algorithm 2.2: Lower bound $\overline{z}^L$ on $\overline{z}$

```
1  Input: $\mathbf{A}, \mathbf{b}, \mathbf{c}$, stopping condition
   Result: a lower bound of $\underline{f}$
2  Set $k \leftarrow 0$.
3  while stopping condition is not met do
4      put $k \leftarrow k + 1$
5      for each $i \in I$ do
6          solve the following problems:
7          $z_i^1 = \min\ \overline{c}^T x$ subject to $x \in \mathbf{x} \cap F,\ x \in S,\ \sum_j \overline{a}_{ij} x_j = \underline{b}_i$,
8          $z_i^2 = \min\ \overline{c}^T x$ subject to $x \in \mathbf{x} \cap F,\ x \in S,\ \sum_j \underline{a}_{ij} x_j = \overline{b}_i$.
9          add to $S$ the extremal version of the $i$-th constraint corresponding to the
             highest among the values $\{z_i^1, z_i^2\}$
10     end
11     set $\overline{z}_k^L \leftarrow \max\{z_m^1, z_m^2\}$
12 end
```

2.5 Experimental results

In this section, we present our computational experiments and related results. The input data for the experiments were generated as follows. For a given problem size

(m, n) and uncertainty parameter (γ), we randomly generated entries of $\underline{A}$ in $[-10, 10]$ and values of a solution $x^* \in \mathbb{R}^n$ in $[1, 10]$, both using a uniform distribution. Vector $\overline{b}$ was composed as $\overline{b} = \underline{A}x^*$. To ensure boundedness of the feasible set, we kept the coefficients of the last row of matrix $\underline{A}$ positive. Matrix $\overline{A}$ and vector $\underline{b}$ were computed as $\overline{A} = \underline{A} + \gamma J$ and $\underline{b} = \overline{b} - \gamma e$, where $J \in \mathbb{R}^{m \times n}$ is a matrix of ones and $e = (1, \ldots, 1)^T$ is a vector of ones with the proper dimension. Similarly, the entries of vector $\underline{c}$ were randomly generated in $[1, 10]$ using a uniform distribution and $\overline{c} = \underline{c} + \gamma e$.

The experiments were carried out on a workstation with an Intel(R) Xeon(R) CPU E31270 processor at 3.4 GHz with 4.00 GB of RAM. All the methods were coded in MATLAB(R2016b), using IBM ILOG CPLEX 12.6 for solving the linear programs and INTLAB v10.2 for the interval arithmetic [116].

Results are shown in Table 2.2. Each number in the table is an average of the results obtained on 20 instances. The first three columns describe the values of the input parameters. The following two columns are related to the exact value of $\overline{z}$, which we computed by implementing the exhaustive search algorithm described in [30]. The columns show the running time and the number of the linear programs to be solved. The last six columns report results of our three proposed methods. We used the Hansen-Bliek-Rohn method [99] for finding the enclosure $(\mathbf{x}_B)$ and the algorithm in [59] to determine $\mathbf{x}$. For each method, the table shows the average running time and the average gap from the optimum. The gap of the lower bound $\overline{z}^L$ is computed as $|\overline{z} - \overline{z}^L|/|\overline{z}|$, while the gap of an upper bound $\overline{z}^U$ id computed as $|\overline{z}^U - \overline{z}|/|\overline{z}|$. When the gap value is not displayed, it means the method failed to find an upper bound on all the 20 instances because of either the singularity of the underlying matrix $\mathbf{A}_B$ or non-positivity of $\mathbf{x}_B$. We set the stopping condition in Algorithms 2.1 and 2.2 to 10 iterations, that is, the maximum number of times those algorithms start over.

The greedy algorithm returns a tight lower bound (with a maximum gap of 0.0799 with $m = 15$, $n = 25$ and $\gamma = 1$) for all the problem sizes and uncertainty parameters, and in many cases, it returns the exact values. For low uncertainty (i.e., $\gamma \leq 0.1$), $\overline{z}^{U_1}$ is a reasonable bound (with a maximum gap of 0.2223 with $m = 3$, $n = 5$ and $\gamma = 0.1$); however, as either the size or the uncertainty of the instances increases, it becomes very hard to find a basis which is feasible for each scenario. Although calculating $\overline{z}^{U_2}$ is always possible, its gap from the optimal value, even for small sizes and low uncertainty, is significant. Finally, all the proposed methods perform well in terms of running times (with an average of 0.1414 of a second for $\overline{z}^{L}$, 0.0408 of a second for $\overline{z}^{U_1}$, and 0.7261 of a second for $\overline{z}^{U_2}$) and in particular, for $m \geq 15$, the average running times of the methods are considerably shorter than those of the exact approach.

Table 2.2: Average running times (sec.) and average gaps from exact values obtained from the three proposed approaches.

input			$\overline{z}$		$\overline{z}^{L}$		$\overline{z}^{U_1}$		$\overline{z}^{U_2}$	
m	n	γ	time	LPs	gap	time	gap	time	gap	time
3	5	0.01	0.0634	8	0	0.0193	0.0035	0.0514	0.0731	0.0539
3	5	0.1	0.0622	8	0	0.0112	0.2223	0.0353	0.4006	0.0477
3	5	0.5	0.0623	8	0	0.0110	0.3268	0.0329	1.3545	0.0495
3	5	1	0.0626	8	0.0018	0.0108	0.7505	0.0312	1.8633	0.0682
5	8	0.01	0.0751	32	0	0.0192	0.0084	0.0339	0.1575	0.0485
5	8	0.1	0.0751	32	0	0.0186	0.1282	0.0316	0.6440	0.0704
5	8	0.5	0.0745	32	0	0.0187	0.9750	0.0303	2.2585	0.1432
5	8	1	0.0753	32	0.0366	0.0196	-	-	2.4909	0.1758
10	15	0.01	0.4415	1,024	0	0.0986	0.0204	0.0635	0.3248	0.1530
10	15	0.1	0.4431	1,024	0	0.0978	0.2189	0.0658	1.3740	0.2631
10	15	0.5	0.4421	1,024	0.0045	0.0965	-	-	3.0030	0.9261
10	15	1	0.4448	1,024	0.0453	0.0964	-	-	2.9160	0.0296
15	25	0.01	18.1444	32,768	0	0.2080	0.0557	0.0634	0.6161	0.4337
15	25	0.1	18.0605	32,768	0	0.2044	-	-	2.6752	0.7554
15	25	0.5	18.1189	32,768	0.0203	0.2032	-	-	4.9617	0.1125
15	25	1	18.1256	32,768	0.0799	0.2009	-	-	4.3123	0.0541
18	50	0.01	821.5837	262,144	0	0.3805	0.0391	0.0671	1.1778	1.2925
18	50	0.1	822.4073	262,144	0	0.3740	-	-	7.0895	2.0670
18	50	0.5	822.9145	262,144	0.0030	0.3669	-	-	20.4363	7.4967
18	50	1	821.1614	262,144	0.0603	0.3726	-	-	16.9554	0.2810

2.6 Conclusions

In this paper, we proposed three approaches to calculate bounds on the worst optimal value of equality-constrained interval linear programming problems. We presented a greedy algorithm to determine a lower bound and two enclosure-based approaches to determine an upper bound. The methods were tested on a set of randomly generated instances. Our results show that the three approaches require relatively low computational times comparing to the exact approach. The greedy algorithm returns a tight bound for all the problem instances. One of the two approaches to find an upper bound computes a good bound, however, it fails to return a solution when uncertainty is high. On the other hand, our second approach to find an upper bound returns a bound on every instance which, however, is not very tight. A possible direction of this research is investigating theoretical properties of the problem to design an exact algorithm, based on a branch and bound strategy, embedding the proposed methods to determine bounds.

CHAPTER 3

THE OUTCOME RANGE PROBLEM IN INTERVAL LINEAR PROGRAMMING

3.1 Introduction

In real life problems, we are sometimes interested in evaluating additional functions of interest over the results of an optimization problem, that is, we are interested in evaluating functions of optimal decisions. Let us consider, for instance, an optimization problem developed to design a new transportation network. A possible function of interest, in addition to a cost function which would be optimized, could be an environmental cost function, useful to evaluate how an optimal transportation network impacts surrounding areas. As another example, decisions regarding the optimal location of clinics in a given region, while they can improve public health in a community, might in turn lead to undesirable consequences on a larger scale, such as disparities in access to healthcare among different communities. We refer to the additional functions of interest as *outcome functions*, which are used to evaluate unintended consequences of optimal decision making.

Outcome functions do not have a direct role in the decision process. They are not, in other words, the main objective function of an optimization problem, whereas they might have a significant role in providing important information for future decisions or actions. This is particularly relevant for government agencies, public health decision makers, policy makers, city managers and other stakeholders who make decisions that have differential impacts on different communities and sub-populations. For example, Nobles et al. [103] and Gentili et al. [48, 49, 50] used outcome functions to evaluate spatial access to pediatric and adult primary care. They developed an optimization model for matching patients and providers, and defined two linear outcome functions to quantify spatial access to healthcare services. In another study, Zheng et al. [136] presented an application in telecommunication networks, where one is interested in designing a network such that enough band-width is allocated between two nodes to minimize the total demand lost. An outcome function of interest, in this context, is the local performance of each node, defined as the volume of unmet requests from the node.

Quantifying the impact of decisions using outcome functions becomes even more relevant when decisions are made in an uncertain environment, which is the focus of this paper. Uncertainty in optimization problems usually derives from uncertainty in input parameters, occurring due to measurement errors, missing data, rounding errors, statistical estimations, etc. Solutions to optimization problems can exhibit considerable sensitivity to perturbations in the input parameters, thus often returning a solution which is highly infeasible and/or suboptimal [10].

Throughout the years, several approaches to treat uncertainty in input data have emerged such as robust optimization, stochastic optimization, parametric programming, fuzzy programming, and interval optimization, depending on the source of uncertainty and the requirements on the returned solution. In this paper, we adopt the approach of interval linear programming (ILP) where we assume that input parameters can vary within a-priori known intervals. Several topics have been subject of research in this area (see [58] for a comprehensive survey on the topics): (i) Oettli and Prager [105] and Rohn [112] addressed the problem of characterizing the set of all possible feasible solutions; (ii) Novotná et al. [104] studied the duality gap problem in interval linear programming; (iii) the problem of describing the set of all possible optimal solutions was studied by Allahdadi and Nehi [2] and later by Garajová and Hladík [44] and also its approximation was discussed by [59, 65, 74]; (iv) the problem of determining a satisficing solution space was subject of study in [127, 137], and also different approaches to get a robust satisficing solution space were discussed in [28, 29]. A problem of particular interest, because of its relevance from an application perspective, is that of finding the range of optimal values of an interval linear program, known in the literature as the *optimal value range problem*. The exact formulation and characterization of the problem was discussed in [30, 57, 97, 111], while [62, 93] developed some approximation algorithms for the intractable cases. A different approach to get a satisficing optimal value range was investigated by [69]. The optimal value range problem has been applied in several application problems, such as transportation problems with interval supply and demand [27, 36, 75], matrix games with interval-valued payoffs [82, 85], and portfolio selection problems with interval approximations of expected returns [78, 79].

In this context, our focus is on studying a problem close to the optimal value range problem where we are interested in determining the range of an outcome function (other than the objective function) associated with an interval linear program. To this aim, we introduce the *Outcome Range Problem* which consists of determining the minimum and the maximum values of a given (additional) linear function over the set of all possible optimal solutions of an interval-valued linear program. We formally define our problem, analyze its relation to and differences with the optimal value range problem, and study a specific case where uncertainty occurs only in the right-hand side of the underlying linear program. We show that solving the outcome range problem to optimality is not an easy task; we then study some theoretical properties of the problem and develop two solution approaches to approximate the optimal values. We evaluate our solution techniques on a set of randomly generated instances, and finally, to outline the relevance of our problem for reliable decision making, we present a case study where we apply our approach to quantify spatial access to healthcare services.

The remainder of the paper is structured as follows. We first present a motivating example to motivate our problem in Section 3.2. We then introduce some basic notations, and formally define the outcome range problem in Section 3.3. We assess the computational complexity of the problem in Section 3.4. In Section 3.5, we explore theoretical properties of our problem. We describe our solution techniques in Section 3.6. In Section 3.7, we discuss results of our experimental study. Section 3.8 presents a healthcare application of our problem. Finally, we summarize our findings in Section 3.9.

3.2 Quantifying an outcome function under uncertainty: a motivating example

Let us consider the classical transportation problem [128] where the main goal is to decide how to transfer goods from a set of m origins to a set of n destinations with minimal cost such that the capacity at each origin is not exceeded, and the demand at each destination is satisfied. The transportation problem can be formulated as follows

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \tag{3.1}$$

$$\text{subject to}$$

$$\sum_{j=1}^{m} x_{ij} \leq s_i, \quad \forall i = 1, ..., n, \tag{3.2}$$

$$\sum_{i=1}^{n} x_{ij} \geq d_j, \quad \forall j = 1, ..., m, \tag{3.3}$$

$$x_{ij} \geq 0, \quad \forall i = 1, ..., n, \ j = 1, ..., m, \tag{3.4}$$

where x_{ij} is a decision variable which determines the size of the shipment from origin i to destination j, c_{ij} is the unit shipping cost from origin i to destination j, s_i is the total supply of origin i, and d_j is the total demand of destination j. The objective function of the model minimizes the total transportation cost. The two sets of constraints ensure that the resulting transportation plan respects the capacity at each origin (Eq. (3.2)), and meets the demand of each destination (Eq. (3.3)). Let us consider a specific instance of the problem where there are three origins and three destinations (see Table 3.1 for shipping costs, supply and demand levels).

Table 3.1: Shipping costs, supply and demand levels.

	to			supply (ton)
from	destination 1	destination 2	destination 3	
origin 1	\$40	\$21	\$23	70
origin 2	\$24	\$43	\$19	75
origin 3	\$31	\$35	\$21	81
demand (ton)	85	64	71	

The optimal shipping cost, considering the input data in Table 3.1, is \$4,945 and the optimal solution to the problem is shown in Figure 3.1, where the label on each arc denotes the total quantity shipped on the arc. We can associate with the transportation problem an outcome function to evaluate, for example, the environmental impact [106, 138] of the optimal transportation plan, as total pounds of CO_2 emissions.

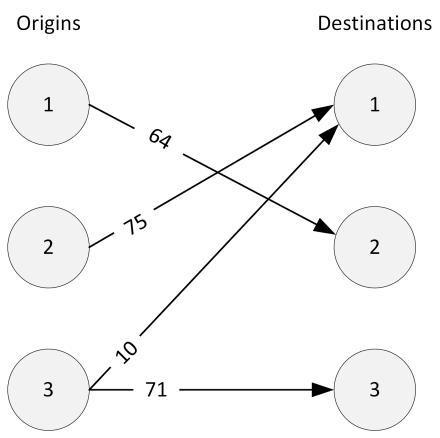


Figure 3.1: The optimal transportation plan for the problem (3.1-3.4) with input data as described in Table 3.1.

The CO_2 emissions depend on the amount of fuel consumed to transport the products to destinations, and consequently vary with the travel distance and with the amount of products. Let r_{ij} denote the total pounds of CO_2 emitted in the atmosphere per unit of the product shipped from origin i to destination j (the specific values of these parameters for our example are reported in Table 3.2), and let $f(x) = \sum_{i,j} r_{ij}x_{ij}$ be an outcome function associated with a given transportation problem. The value of this outcome function on the optimal transportation plan for our example is equal to 3,940 *lb*.

Table 3.2: The CO_2 emission associated with the arcs of the transportation network.

	to		
from	destination 1	destination 2	destination 3
origin 1	30 *lb*	17 *lb*	18 *lb*
origin 2	19 *lb*	32 *lb*	14 *lb*
origin 3	22 *lb*	25 *lb*	17 *lb*

Now let us assume that the demands are not known with certainty, but rather they vary in given intervals. Then the mathematical formulation reads

$$
\begin{aligned}
\min \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} & \\
\text{subject to} & \\
\sum_{j=1}^{m} x_{ij} &\le s_i, && \forall i = 1,...,n,\\
\sum_{i=1}^{n} x_{ij} &\ge [\underline{d}_j, \overline{d}_j], && \forall j = 1,...,m,\\
x_{ij} &\ge 0, && \forall i = 1,...,n,\ j = 1,...,m,
\end{aligned}
$$

where $[\underline{d}_j, \overline{d}_j]$ is the range of values which can be assumed by the demand at destination j, for all j. The question we would like to address is: *how does uncertainty in the parameters affect the environmental cost?* That is, *how does the environmental cost (the total CO_2 emission) change when the parameters change?* If we apply one of the most commonly used approaches to address uncertainty in optimization models such as, for example, robust optimization, we would only be able to evaluate the outcome function on a single robust solution [123] or a number of solutions with some level of protection against uncertainty in the data [10]. However, such an evaluation would not answer our question of quantifying the variation of the outcome function in response to uncertainty in the parameters. A much more useful information would be, for example, the range of variation of the outcome function, that is, the *best* and *worst* values of the outcome function over the set of *all* the optimal solutions corresponding to all realizations of the uncertain data.

In our example, let us assume the demand level intervals are $d_1 \in [85, 87]$, $d_2 \in [64, 66]$, and $d_3 \in [71, 73]$. For the sake of clarity in the exposition, let us also assume that the demand at the destinations can only take integer values in the given intervals. By applying a conservative robust approach, we would look for a shipment plan which is feasible under all the possible data perturbations, and would then evaluate

Table 3.3: All the possible realizations of the uncertain demand with the corresponding optimal solutions and values of $f(x)$.

data realization	demand values	optimal solutions									$f(x)$
		x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{23}	x_{31}	x_{32}	x_{33}	
1	$\{d_1 = 85, d_2 = 64, d_3 = 71\}$	0	64	0	75	0	0	10	0	71	3,940 *lb*
2	$\{d_1 = 85, d_2 = 64, d_3 = 72\}$	0	64	1	75	0	0	10	0	71	3,958 *lb*
3	$\{d_1 = 85, d_2 = 64, d_3 = 73\}$	0	64	2	75	0	0	10	0	71	3,976 *lb*
4	$\{d_1 = 85, d_2 = 65, d_3 = 71\}$	0	65	0	75	0	0	10	0	71	3,957 *lb*
5	$\{d_1 = 85, d_2 = 65, d_3 = 72\}$	0	65	1	75	0	0	10	0	71	3,975*lb*
6	$\{d_1 = 85, d_2 = 65, d_3 = 73\}$	0	65	2	75	0	0	10	0	71	3,993 *lb*
7	$\{d_1 = 85, d_2 = 66, d_3 = 71\}$	0	66	0	75	0	0	10	0	71	3,974 *lb*
8	$\{d_1 = 85, d_2 = 66, d_3 = 72\}$	0	66	1	75	0	0	10	0	71	3,992 *lb*
9	$\{d_1 = 85, d_2 = 66, d_3 = 73\}$	0	66	2	75	0	0	10	0	71	4,010 *lb*
10	$\{d_1 = 86, d_2 = 64, d_3 = 71\}$	0	64	1	75	0	0	11	0	70	3,963 *lb*
11	$\{d_1 = 86, d_2 = 64, d_3 = 72\}$	0	64	2	75	0	0	11	0	70	3,981 *lb*
12	$\{d_1 = 86, d_2 = 64, d_3 = 73\}$	0	64	3	75	0	0	11	0	70	3,999 *lb*
13	$\{d_1 = 86, d_2 = 65, d_3 = 71\}$	0	65	1	75	0	0	11	0	70	3,980 *lb*
14	$\{d_1 = 86, d_2 = 65, d_3 = 72\}$	0	65	2	75	0	0	11	0	70	3,998 *lb*
15	$\{d_1 = 86, d_2 = 65, d_3 = 73\}$	0	65	3	75	0	0	11	0	70	4,016 *lb*
16	$\{d_1 = 86, d_2 = 66, d_3 = 71\}$	0	66	1	75	0	0	11	0	70	3,997 *lb*
17	$\{d_1 = 86, d_2 = 66, d_3 = 72\}$	0	66	2	75	0	0	11	0	70	4,015 *lb*
18	$\{d_1 = 86, d_2 = 66, d_3 = 73\}$	0	66	3	75	0	0	11	0	70	4,033 *lb*
19	$\{d_1 = 87, d_2 = 64, d_3 = 71\}$	0	64	2	75	0	0	12	0	69	3,986 *lb*
20	$\{d_1 = 87, d_2 = 64, d_3 = 72\}$	0	64	3	75	0	0	12	0	69	4,004 *lb*
21	$\{d_1 = 87, d_2 = 64, d_3 = 73\}$	0	64	4	75	0	0	12	0	69	4,022 *lb*
22	$\{d_1 = 87, d_2 = 65, d_3 = 71\}$	0	65	2	75	0	0	12	0	69	4,003 *lb*
23	$\{d_1 = 87, d_2 = 65, d_3 = 72\}$	0	65	3	75	0	0	12	0	69	4,021 *lb*
24	$\{d_1 = 87, d_2 = 65, d_3 = 73\}$	0	65	4	75	0	0	12	0	69	4,039 *lb*
25	$\{d_1 = 87, d_2 = 66, d_3 = 71\}$	0	66	2	75	0	0	12	0	69	4,020 *lb*
26	$\{d_1 = 87, d_2 = 66, d_3 = 72\}$	0	66	3	75	0	0	12	0	69	4,038 *lb*
27	$\{d_1 = 87, d_2 = 66, d_3 = 73\}$	0	66	4	75	0	0	12	0	69	4,056 *lb*

the outcome function on the returned robust solution. In this case for example, by applying the worst case robust approach [40], we would choose to ship 87 units to destination 1, 66 units to destination 2, and 73 units to destination 3 for a total cost of \$ 5,099, and with an environmental impact equal to 4,056 *lb*.

Let us list, for this simple example, all the realizations of the uncertain data. They are shown in the first two columns of Table 3.3. For each realization of the data, we solved the corresponding transportation problem, and evaluated the outcome function on the corresponding optimal solution. Columns 3-11 in the table report the optimal solutions, and the last column in the table reports the corresponding value of the outcome function. In this simple example, the best value of the outcome function is equal to 3,940 *lb* (corresponding to scenario 1) and the worst value is equal to 4,056 *lb* (corresponding to scenario 27). Hence, in this case, we can say that given all the possible realizations of the interval data, the total CO_2 emission of the transportation plan would range between 3,940 *lb* and 4,056 *lb*. As can be seen from the results, the optimal solutions are very sensitive to the demand perturbations. This makes the problem of finding the best and the worst values of $f(x)$ a nontrivial one.

In this simple example, given a linear program with interval parameters and a linear outcome function, we determined the best and the worst values of the latter over all the possible optimal solutions obtained from all the realizations of the interval data. We refer to this problem as the outcome range problem. Its formal definition is given in the next section.

3.3 The outcome range problem

Let us introduce some needed notation which is commonly used in the interval linear programming literature [58, 113]. Given two matrices $\underline{A}, \overline{A} \in \mathbb{R}^{m\times n}$, we define an interval matrix as the set

$$\mathbf{A} = [\underline{A}, \overline{A}] := \{A \in \mathbb{R}^{m\times n} : \ \underline{A} \leq A \leq \overline{A}\},$$

where matrices $\underline{A}, \overline{A}$ are called the lower and the upper bounds of $\mathbf{A}$, respectively, and comparing matrices is understood componentwise. The set of all m-by-n real interval matrices is denoted by $\mathbb{IR}^{m\times n}$. We define an interval vector analogously. For the sake of simplicity, we write $\mathbb{IR}^{m}$ instead of $\mathbb{IR}^{m\times 1}$ to denote the set of all real interval vectors of order m. Throughout this paper, we use bold symbols for interval vectors and matrices. Let us consider the following interval linear programming (ILP) problem in the form of

$$\min\ \mathbf{c}^T x \ \text{ subject to } \ x \in \mathcal{M}(\mathbf{A}, \mathbf{b}), \tag{3.5}$$

where we are given $\mathbf{c} \in \mathbb{IR}^n$, $\mathbf{b} \in \mathbb{IR}^m$, and $\mathbf{A} \in \mathbb{IR}^{m \times n}$. The set $\mathcal{M}(\mathbf{A}, \mathbf{b})$ denotes the feasible set described by linear constraints with the interval coefficient matrix $\mathbf{A}$ and the interval right-hand side vector $\mathbf{b}$. Interval linear programming has been extensively studied with three main types of $\mathcal{M}(\mathbf{A}, \mathbf{b})$, which are shown in Table 3.4. The type of constraints and restriction on variables in an interval linear program can considerably impact its properties. Thus, each type of interval linear programs is usually treated separately in the literature.[1]

Table 3.4: Different types of interval linear constraints [58]

type	interval linear system
(I)	$\mathcal{M}(\mathbf{A}, \mathbf{b}) = \{x \in \mathbb{R}^n;\ \ \mathbf{A}x = \mathbf{b},\ \ x \geq 0\}$
(II)	$\mathcal{M}(\mathbf{A}, \mathbf{b}) = \{x \in \mathbb{R}^n;\ \ \mathbf{A}x \leq \mathbf{b}\}$
(III)	$\mathcal{M}(\mathbf{A}, \mathbf{b}) = \{x \in \mathbb{R}^n;\ \ \mathbf{A}x \leq \mathbf{b},\ \ x \geq 0\}$

We refer to any triple (A, b, c), where $A \in \mathbf{A}$, $b \in \mathbf{b}$, and $c \in \mathbf{c}$, as a *scenario*. With each scenario (A, b, c), we can associate a linear program, namely LP(A, b, c), whose feasible set and optimal value are denoted by $\mathcal{M}(A, b)$ and $z(A, b, c)$, respectively, i.e.,

$$z(A, b, c) := \min\ c^T x \ \text{ subject to } \ x \in \mathcal{M}(A, b).$$

Hence, an interval linear program is a family of linear programs associated with all $A \in \mathbf{A}$, $b \in \mathbf{b}$ and $c \in \mathbf{c}$. For a particular scenario (A, b, c), the corresponding LP(A, b, c) can be infeasible, unbounded or admit a finite optimal value. We denote by $S(A, b, c)$ an optimal solution (or the set of all optimal solutions) of a linear program LP(A, b, c), if any, admitting a finite optimal value. We denote by Ω the set

[1]References [30, 60] address the general form.

of all the optimal solutions of an interval linear program, referred to as the *optimal set*, that is,

$$\Omega := \bigcup_{A\in\mathbf{A}, b\in\mathbf{b}, c\in\mathbf{c}} S(A,b,c).$$

We are now ready to formally define our problem. Given the ILP (3.5) and an additional linear function $f : \mathbb{R}^n \to \mathbb{R}$, where $f(x) = r^T x$ with $r \in \mathbb{R}^n$, the *outcome range problem* consists in solving the two following optimization problems

$$\underline{f} := \min \ f(x) \ \text{ subject to } \ x \in \Omega,$$
$$\overline{f} := \max \ f(x) \ \text{ subject to } \ x \in \Omega.$$

We define the pair of optimal values $\{\underline{f}, \overline{f}\}$ to be the optimal solution of the outcome range problem.

Example 3.1. Consider the following two-dimensional linear program with interval right-hand sides

$$\min \ (2,-5)^T x \ \text{ subject to } \begin{pmatrix} 1 & -1 \\ -1 & -1 \\ 0 & 1 \end{pmatrix} x \leq \begin{pmatrix} [4,7] \\ [-6,8] \\ [4,9] \end{pmatrix}, \quad x \geq 0,$$

and consider the following outcome function

$$f(x) = 8x_1 + 9x_2.$$

Let us consider Figure 3.2 where the optimal solution $\{\underline{f}, \overline{f}\}$ of the problem is shown. In the figure, the intersection and the union of all the feasible sets of the linear programs associated with all the scenarios are shown in dark and light gray, respectively. Specifically, the intersection of all the feasible sets is obtained by setting the right-hand sides at their lower bound, while the union of all the feasible sets is obtained by setting all the right-hand sides at their upper bound. The black area represents the set Ω, that is, the set of all optimal solutions obtained from all the realizations of the interval data. Both the minimum and the maximum values of $f(x)$ occur at the endpoints of the bold line and are shown in the figure. Their values are $\underline{f} = 36$ and $\overline{f} = 81$, respectively. In particular, $\overline{f}$ is obtained on the point $x^{1*} = (0, 9)$ which is the optimal point of several linear programs one of which is associated, for example, with scenario $b^T = (4, -6, 9)$, while $\underline{f}$ is obtained on the point $x^{2*} = (0, 4)$ which is the optimal point of a linear program associated, for example, with scenario $b^T = (7, 8, 4)$.

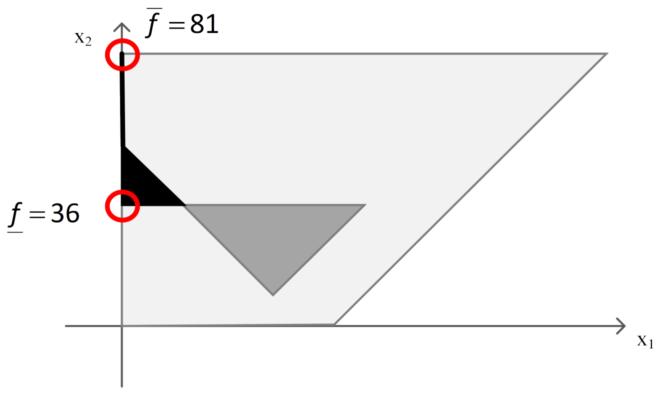


Figure 3.2: (Example 3.1) Intersection of all feasible sets in dark gray; union in light gray; set of all optimal solutions in black.

3.3.1 Our focus

As can be observed from Example 3.1, the difficulty in solving the outcome range problem relies on the fact that its feasible set, that is, the set Ω, is not explicitly known; nor a convenient implicit description of it (e.g., polyhedral description) is available in general [44, 59] (with some exceptions as outlined in Section 3.4). This is true even if we consider a simplified version of the underlying ILP where we only deal with interval right-hand sides. As we mentioned earlier, the three types of interval linear programs are analyzed separately in the literature because the feasible region and the optimal set might change when applying standard linear transformations. In the discussion to follow, we will focus on solving the outcome range problem when the underlying interval linear program is of Type III, that is, it contains inequality and non-negativity constraints, and uncertainty occurs only in the right-hand side of the program. Formally, we assume the following interval linear program:

$$[\mathrm{ILP}_b] \qquad \min\ c^T x \ \text{ subject to } \ Ax \leq \mathbf{b}, \ \ x \geq 0, \tag{3.6}$$

where $c \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{IR}^m$, and $A \in \mathbb{R}^{m \times n}$ are given. The linear program and an optimal solution (or the set of all optimal solutions), if one exists, associated with a given scenario $b \in \mathbf{b}$ are denoted by $\mathrm{LP}(b)$ and $s(b)$, respectively. We also denote by $z(b)$ the optimal value corresponding to $\mathrm{LP}(b)$ (infinity and infeasiblity are also allowed). We focus on solving the two following optimization problems

$$\underline{f} = \min\ f(x) \ \text{ subject to } \ x \in \Omega_b, \tag{3.7}$$

$$\overline{f} = \max\ f(x) \ \text{ subject to } \ x \in \Omega_b, \tag{3.8}$$

where Ω_b is the optimal set of $\mathrm{ILP}_\mathbf{b}$. In the rest of the paper, we will refer to this special case of the outcome range problem as $\mathrm{ORP}_\mathbf{b}$.

Remark 3.1. From an application perspective, solving $\mathrm{ORP}_\mathbf{b}$ is meaningful when the set Ω_b is not empty and bounded (see [43, 44] for conditions for emptiness and boundedness of Ω_b). In what follows, we will assume this is the case.

3.4 Computational complexity of the outcome range problem ($\mathbf{ORP_b}$)

We here address the computational complexity of $\mathrm{ORP}_\mathbf{b}$. Some additional notation is needed at this point. Let us recall the linear program associated with a given $b \in \mathbf{b}$

$$\min\ c^T x \ \text{ subject to } \ Ax \leq b, \ x \geq 0.$$

The standard form reads

$$\min\ c^T x + 0^T d \ \text{ subject to } \ Ax + Id = b,\ x \geq 0,\ d \geq 0, \tag{3.9}$$

where $d \in \mathbb{R}^m$ is the vector of slack variables and $I \in \mathbb{R}^{m\times m}$ is the identity matrix. Let us define $\tilde{A} := [A|I]$, $\tilde{c}^T := [c^T|0^T]$, and $\tilde{x}^T := [x^T|d^T]$. We rewrite (3.9) as

$$\min\ \tilde{c}^T\tilde{x} \ \text{ subject to } \ \tilde{A}\tilde{x} = b,\ \tilde{x} \geq 0. \tag{3.10}$$

We similarly define $\tilde{r}^T := [r^T|0^T]$.

Definition 3.1. By a basis B we mean an index set $B \subseteq \{1, \ldots, n+m\}$ such that $\tilde{A}_B$ is nonsingular, where a subscript B on a matrix (row vector) denotes the submatrix (subvector) composed of columns indexed by B. That is, set B is the set of indices associated with basic variables. Analogously, an index set $N := \{1, \ldots, n+m\} \setminus B$ indicates indices for nonbasic variables and as a subscript it represents restriction to nonbasic indices.

A basis B is an optimal basis of LP (3.10) if and only if it satisfies the following conditions

$$\tilde{A}_B^{-1} b \geq 0, \tag{3.11a}$$

$$\tilde{c}_N^T - \tilde{c}_B^T \tilde{A}_B^{-1} \tilde{A}_N \geq 0^T. \tag{3.11b}$$

Now let us recall the assumptions under which the optimal set $\Omega_{\mathbf{b}}$ can be explicitly defined.

Definition 3.2. An ILP$_{\mathbf{b}}$ problem is said to be *B-stable*, if B is an optimal basis of LP(b) for all $b \in \mathbf{b}$. Furthermore, it is called *unique B-stable* if it is B-stable and the optimal basis of LP(b) is unique for all $b \in \mathbf{b}$.

B-stability is a very important property in interval linear programming because it can simplify the description of the optimal set. In the case of unique B-stability of $\text{ILP}_{\mathbf{b}}$, the optimal set (Ω_b) can be described by a polyhedral set.

Lemma 3.1. *[5] If (3.6) is unique B-stable with the optimal basis B, the optimal set (Ω_b) is described by the following linear system* [2]

$$\tilde{A}_B\tilde{x}_B \leq \overline{b}, \ \ -\tilde{A}_B\tilde{x}_B \leq -\underline{b}, \ \ \tilde{x}_B \geq 0, \ \ \tilde{x}_N = 0.$$

Another relevant topic in interval linear programming is determining the optimal value range, that is, the problem of finding the best and the worst optimal values among all the optimal values obtained over all data perturbations. We define the optimal value range of $\text{ILP}_{\mathbf{b}}$ (3.6) as

$$\underline{z} := \inf \{z(b) : \ b \in \mathbf{b}\}, \tag{3.12}$$

$$\overline{z} := \sup \{z(b) : \ b \in \mathbf{b}\}. \tag{3.13}$$

Note that (3.12) and (3.13) can assume any value, including infinity and infeasibility. The interval $[\underline{z}, \overline{z}]$ then gives the optimal value range. By [30, 125], we know that for $\text{ILP}_{\mathbf{b}}$ (3.6) we have:

$$\underline{z} = \min \ c^T x \ \text{ subject to } \ Ax \leq \overline{b}, \ x \geq 0, \tag{3.14}$$

$$\overline{z} = \min \ c^T x \ \text{ subject to } \ Ax \leq \underline{b}, \ x \geq 0. \tag{3.15}$$

Now we analyze the computational complexity of the outcome range problem. Specifically, Theorem 3.1 assesses the computational complexity of $\text{ORP}_{\mathbf{b}}$. Proposition 3 considers a special case of $\text{ORP}_{\mathbf{b}}$ which is polynomially solvable. Finally, Proposition 4 and Corollary 3.1 investigate another polynomially solvable case by exploiting a

[2]We adopt Lemma 3.1 from the results discussed in [61] (see [61] for more details).

relation between ORP$_{\mathbf{b}}$ and the optimal value range problem.

Theorem 3.1. *Problem ORP$_{\mathbf{b}}$ is NP-hard.*

Proof. We proceed by a different interval-related problem which is known to be NP-hard. Let us consider an ILP problem of Type I with a fixed coefficient matrix and a fixed objective vector (i.e., fixed A and c), i.e.,

$$\min\ c^T x \ \text{ subject to } \ Ax = \mathbf{b}, \ \ x \geq 0. \tag{3.16}$$

Let Ξ be the optimal set of (3.16). By Theorem 7 in [44] (cf. p. 282), we know that computing the exact interval hull of Ξ is NP-hard. Now let us reformulate problem (3.16) as follows

$$\min\ c^T x \ \text{ subject to } \ Ax \leq \mathbf{b}, \ \ -Ax \leq -\mathbf{b}, \ \ x \geq 0. \tag{3.17}$$

We know by Theorem 2 in [47] (cf. p. 606) that the optimal set of (3.17) is equal to the optimal set of (3.16). For the sake of simplicity, let us introduce the following notation

$$A' := \left[\begin{array}{c} A \\ \hline -A \end{array}\right], \ \ \mathbf{b}' := \left[\begin{array}{c} \mathbf{b} \\ \hline -\mathbf{b} \end{array}\right].$$

We then can rewrite the problem (3.17) as an ILP$_{\mathbf{b}}$, that is,

$$\min\ c^T x \ \text{ subject to } \ A'x \leq \mathbf{b}', \ \ x \geq 0.$$

Therefore, we can conclude that $\Xi = \Omega_b$. As a result, we can say that computing the exact interval hull of Ω_b is also NP-hard. Now if we consider $f(x) = x_i$, for any $i \in \{1, \ldots, n\}$, we can conclude that ORP$_{\mathbf{b}}$ is NP-hard. $\square$

Proposition 3. *If ILP$_{\mathbf{b}}$ is unique B-stable, then ORP$_{\mathbf{b}}$ is polynomially solvable.*

Proof. Let the basis B be the unique optimal basis for all the data realizations, then based on Lemma 3.1, ORP$_{\mathbf{b}}$ is equivalent to solving the two following linear programs

$$\underline{f} = \min\ \tilde{r}_B^T \tilde{x}_B \ \text{ subject to }\ \tilde{A}_B\tilde{x}_B \le \overline{b},\ \ -\tilde{A}_B\tilde{x}_B \le -\underline{b},\ \ \tilde{x}_B \ge 0,\ \ \tilde{x}_N = 0, \tag{3.18}$$

$$\overline{f} = \max\ \tilde{r}_B^T \tilde{x}_B \ \text{ subject to }\ \tilde{A}_B\tilde{x}_B \le \overline{b},\ \ -\tilde{A}_B\tilde{x}_B \le -\underline{b},\ \ \tilde{x}_B \ge 0,\ \ \tilde{x}_N = 0. \tag{3.19}$$

□

Let $\mathscr{B}$ be the set of all optimal bases of ILP$_{\mathbf{b}}$ (if any). Let us consider a given $B \in \mathscr{B}$. We can associate with it, by (3.11b), a cone containing all the cost vectors that are optimal for B, that is,

$$H^B := \{\psi \in \mathbb{R}^{m+n} :\ \psi_N^T - \psi_B^T \tilde{A}_B^{-1}\tilde{A}_N \ge 0^T\}.$$

We define $\mathscr{C}$ as the intersection of all the cones containing all the cost vectors for which basis B remains optimal, that is,

$$\mathscr{C} = \bigcap_{B \in \mathscr{B}} H^B.$$

The following proposition states another polynomially solvable case of ORP$_{\mathbf{b}}$ by leveraging a relation with the optimal value range problem.

Proposition 4. *Suppose that $\underline{z}$ and $\overline{z}$ are finite values. If r is such that $r \in \mathcal{C}$, then ORP$_{\mathbf{b}}$ is polynomially solvable.*

Proof. Let us recall that the optimal value range of ILP$_{\mathbf{b}}$ is polynomially solvable, that is,

$$[\mathrm{P}_1] : \underline{z} = \min\ c^T x \ \text{ subject to } \ Ax \leq \overline{b},\ x \geq 0,$$

$$[\mathrm{P}_2] : \overline{z} = \min\ c^T x \ \text{ subject to } \ Ax \leq \underline{b},\ x \geq 0,$$

and that $\mathrm{ORP}_{\mathbf{b}}$ consists in solving the following two optimization problems

$$[\mathrm{P}_3] :\ \underline{f} = \min\ r^T x \ \text{ subject to } \ x \in \Omega_b, \quad [\mathrm{P}_4] :\ \overline{f} = \max\ r^T x \ \text{ subject to } \ x \in \Omega_b.$$

From the hypothesis, we know that $\underline{z}$ is a finite value. Let x^* be an optimal solution of P_1, i.e., $\underline{z} = c^T x^*$. By definition, we know that $x^* \in \Omega_b$. Since $r \in \mathcal{C}$, we can write

$$r^T x^* = \min\ \ r^T x \ \text{ subject to } \ Ax \leq \overline{b},\ x \geq 0.$$

Let us now consider a generic $\hat{x} \in \Omega_b$ in P_3, which is an optimal solution of the linear program associated with a scenario $\hat{b} \in \mathbf{b}$. Again, given $r \in \mathcal{C}$, we have

$$r^T \hat{x} = \min\ \ r^T x \ \text{ subject to } \ Ax \leq \hat{b},\ x \geq 0.$$

Since $\hat{b} \leq \overline{b}$ and $A\hat{x} \leq \overline{b}$, we can say $r^T \hat{x} \geq r^T x^*$. This is true for any vector $\hat{x} \in \Omega_b$, and thus x^* is also an optimal solution to P_3. Therefore, we can compute $\underline{f}$ by

$$\underline{f} = \min\ r^T x \ \text{ subject to } \ Ax \leq \overline{b},\ x \geq 0,$$

which is polynomially solvable. We can use a similar argument for P_2 and P_4. □

Now it is easy to see that the following special case of Proposition 4 holds.

Corollary 3.1. *Suppose that $f(x) = c^T x$. If $\underline{z}$ and $\overline{z}$ are finite values, then we have $[\underline{f}, \overline{f}] = [\underline{z}, \overline{z}]$.*

Note that even if we assume that $f(x) = c^T x$, the outcome range problem is not equivalent to the optimal value range problem in general. Below, we illustrate this by an example.

Example 3.2. Consider the following ILP$_\mathbf{b}$ problem

$$\min\ -4x_2 \ \text{ subject to } \ x_1 + x_2 \leq [-1, 5],\ x_1, x_2 \geq 0,$$

and let $f(x) = -4x_2$ also be an outcome function. By (3.15), it is easy to see that $\overline{z}$ is infeasible, and by applying (3.14) we get $\underline{z} = -20$. Therefore, $[-20, \infty]$ gives the optimal value range.[3] However, from Figure 3.3, it is not hard to observe that

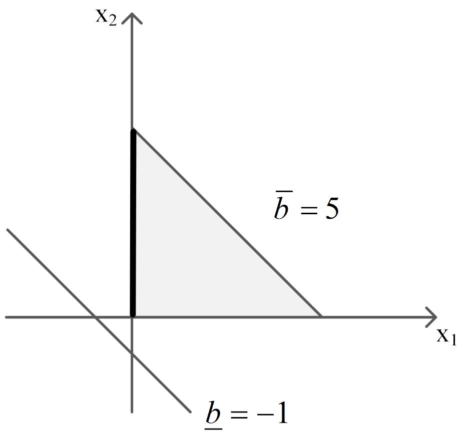


Figure 3.3: (Example 3.2) Union of all feasible sets in light gray; set of all optimal solutions in bold.

$\underline{f} = -20$ (scenario $b = 5$) and $\overline{f} = 0$ (scenario $b = 0$). Hence, the outcome function $f(x)$ ranges in the interval $[-20, 0]$, which is different from the optimal value range.

Corollary 3.1 and Example 3.2 indeed imply that the outcome range problem can be seen as a generalized form of a special case of the optimal value range problem [64].

[3]We denote infeasibility by the convention $\min\ \emptyset = \infty$.

3.5 Properties of the outcome range problem ($\mathrm{ORP}_{\mathbf{b}}$)

In this section, we study some theoretical properties of $\mathrm{ORP}_{\mathbf{b}}$ aimed at characterizing the scenarios corresponding to the optimal values of (3.7) and (3.8). Throughout this section we only report results related to the computation of $\underline{f}$. All the results are applicable to the computation of $\overline{f}$ as well. Let us introduce some definitions first.

Definition 3.3. A given scenario $b \in \mathbf{b}$ is referred to as

(i) *a middle scenario if*

$$\underline{b}_i < b_i < \overline{b}_i,\ \forall i \in \{1, \ldots, m\}.$$

(ii) *a weakly extremal scenario if*

$$b_i = \underline{b}_i\ \vee\ b_i = \overline{b}_i,\ \text{for some } i \in \{1, \ldots, m\}.$$

(iii) *a strongly extremal scenario if*

$$b_i = \underline{b}_i\ \vee\ b_i = \overline{b}_i,\ \forall i \in \{1, \ldots, m\}.$$

Note that, according to the above definition, a strongly extremal scenario is also a weakly extremal scenario, but the opposite does not hold true. From the geometrical standpoint, given a hypercube $\mathbf{b}$, a middle scenario is in the interior of the hypercube, a weakly extremal scenario is on the boundary of the hypercube, and a strongly extremal scenario is a vertex of the hypercube.

Definition 3.4. $b^* \in \mathbf{b}$ is an *optimal scenario* of (3.7) if $\underline{f} = f(x^*)$, where $x^* \in s(b^*)$.

Definition 3.5. Given an optimal scenario b^* for (3.7), an optimal basis B^* of the linear program $\mathrm{LP}(b^*)$ is a *global optimal basis* of (3.7).

Remark 3.2. Note that, given a global optimal basis B^* of (3.7), the optimal value $\underline{f}$ and the optimal scenario b^* are the optimal value and an optimal solution, respectively, of the following linear program

$$\min\ \tilde{r}_{B^*}^T \tilde{A}_{B^*}^{-1} b \ \text{ subject to } \ \tilde{A}_{B^*}^{-1} b \geq 0,\ b \in \mathbf{b},$$

in variables b.

The results to follow identify conditions to characterize the optimal scenario b^* either as a middle or a weakly (strongly) extremal scenario.

Proposition 5. *If* $(0, \ldots, 0)^T \notin \mathbf{b}$*, then there exists a weakly extremal scenario* $\hat{b}$ *such that* $b^* = \hat{b}$*.*

Proof. Let the basis B^* be a global optimal basis of (3.7). By Remark 3.2, the optimal scenario b^* is an optimal solution of the following linear program

$$\min\ \tilde{r}_{B^*}^T \tilde{A}_{B^*}^{-1} b \ \text{ subject to } \ \tilde{A}_{B^*}^{-1} b \geq 0,\ b \in \mathbf{b}. \tag{3.20}$$

Let us refer to the feasible set of (3.20) as P_{B^*}. It is known that there exists an optimal solution of (3.20) which is an extreme point of P_{B^*}.

We know that each vertex of P_{B^*} is a vector satisfying all the constraints such that at least m of the constraints are binding and are linearly independent. We also know that matrix $\tilde{A}_{B^*}^{-1}$ is a full rank square matrix of order m. Note that since $(0, \ldots, 0)^T \notin \mathbf{b}$, then $\tilde{A}_{B^*}^{-1} b \neq 0$ for each $b \in \mathbf{b}$. Therefore, any extreme point of P_{B^*} corresponds to a vector for which at least one of the constraints in $b \in \mathbf{b}$ is binding. We can then conclude that there exists $i \in \{1, \ldots, m\}$ such that $b_i^* = \underline{b}_i$ or $b_i^* = \overline{b}_i$. This completes the proof. $\square$

Note that b^* can still be a weakly extremal scenario even in the case of $(0, \ldots, 0)^T \in \mathbf{b}$, but this requires the vector $(0,\ldots,0)^T$ not to be a middle scenario.

Corollary 3.2. *Suppose that vector $(0,\ldots,0)^T$ is such that it is a weakly extremal scenario of of the interval vector $\mathbf{b}$. Then there exists a weakly extremal scenario $\hat{b}$ such that $b^* = \hat{b}$.*

Proposition 5 also reveals an interesting observation related to middle scenarios.

Corollary 3.3. *If the optimal scenario b^* is a unique middle scenario, then we have $b^* = (0,\ldots,0)^T$.*

Proof. Similar to the proof of Proposition 5, let the basis B^* be a global optimal basis of (3.7). Consequently, the optimal scenario b^* is an optimal solution of (3.20). Suppose for the sake of contradiction that b^* is a unique middle scenario, i.e., $\underline{b}_i < b_i^* < \overline{b}_i$ for all $i \in \{1,\ldots,m\}$, but $b^* \neq (0,\ldots,0)^T$. Since b^* is the unique optimal solution of (3.20), then m linearly independent constraints needs to be binding on b^* to form an extreme point. $\tilde{A}_{B^*}^{-1}$ is a full rank square matrix of order m; thus, we need to have $\tilde{A}_{B^*}^{-1}b = 0$. This system possesses one unique solution which is $(0,\ldots,0)^T$. Therefore, $b^* = (0,\ldots,0)^T$. We derive a contradiction here, and this completes the proof. $\square$

Note that the opposite of Corollary 3.3 is not valid. That is, if the optimal scenario is $b^* = (0,\ldots,0)^T$, then this does not necessarily imply neither that b^* is unique, nor that it is a middle scenario. The following provides a counterexample.

Example 3.3. Consider the following interval linear program

$$\min\ 5x_1 + 6x_2 \ \text{ subject to } \ 4x_1 + 5x_2 \leq [0,5],\ x_1, x_2 \geq 0,$$

and let $f(x) = 10x_1 + 3x_2$ be an outcome function. From Figure 3.4, we observe that $x^* = (0,0)$ is the unique optimal solution for all the linear programs LP(b), for all $b \in \mathbf{b}$, that is, $\Omega_b = \{(0,0)\}$. Hence, we have $\underline{f} = 0$. It is not hard to see that any

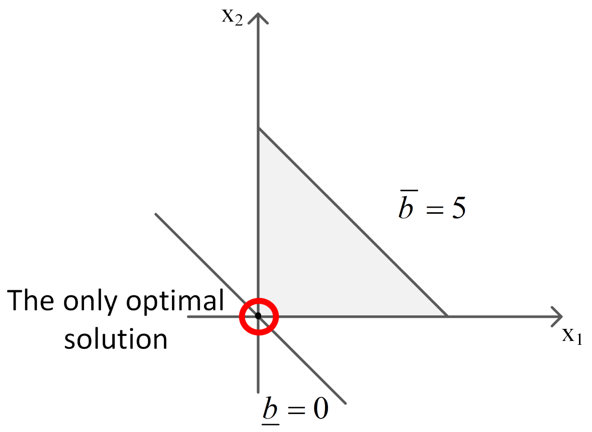


Figure 3.4: (Example 3.3) Union of all feasible sets in light gray; the only optimal solution is red circled.

scenario in the interval $[0, 5]$ is an optimal scenario for $\text{ORP}_{\mathbf{b}}$. Therefore, $b^* = 0$ is an optimal scenario, but it is neither unique nor a middle scenario.

We now present a condition under which b^* is a strongly extremal scenario.

Proposition 6. *If B^* is non-degenerate for LP(b^*), then b^* is a strongly extremal scenario.*

Proof. Let us recall that, given a global optimal basis B^*, the following linear program returns $\underline{f}$ and b^*.

$$\min\ \tilde{r}_{B^*}^T \tilde{A}_{B^*}^{-1} b \ \text{ subject to } \ \tilde{A}_{B^*}^{-1} b \geq 0,\ b \in \mathbf{b}$$

We know that B^* is a non-degenerate optimal basis of LP(b^*), and thus b^* is such that $\tilde{A}_{B^*}^{-1} b^* > 0$. Therefore, to have an extreme point, m linearly independent constraints in $b \in \mathbf{b}$ need to binding on b^*, that is, $b_i^* = \underline{b}_i$ or $b_i^* = \overline{b}_i$ for all $i \in \{1, \ldots, m\}$. The proof is now concluded. □

Finally, the following observation states another case under which an optimal scenario b^* is strongly extremal. It follows directly from Proposition 4 in Section 3.4.

Observation 3.1. *Assume that $\underline{z}$ is a finite value. If r is such that $r \in \mathcal{C}$, then we have $b^* = \overline{b}$, which is a strongly extremal scenario.*

3.6 Solution methods

In Section 3.4, we show that $\text{ORP}_{\mathbf{b}}$ is an NP-hard problem in general; however, when the underlying $\text{ILP}_{\mathbf{b}}$ is unique B-stable, we can solve $\text{ORP}_{\mathbf{b}}$ to optimality in polynomial time. B-stability is unlikely to occur when we are dealing with wide intervals and large problems. Therefore, unless P=NP, there is no hope for any polynomial-time solvable characterization of the problem in general. As such, we here describe two different approaches to approximate the optimal solution of $\text{ORP}_{\mathbf{b}}$. Specifically, we present a super-set based method and a local search algorithm.

3.6.1 Super-set based method

As stated in the previous sections, an explicit description of the optimal set Ω_b is not always available. However, if we are able to find a super-set $E(\Omega_b)$ containing it, i.e., such that $\Omega \subseteq E(\Omega_b)$, we could then approximate the optimal values $\underline{f}$ and $\overline{f}$ by solving the two following optimization problems

$$\underline{f}^L = \min\ r^T x \ \text{ subject to } \ x \in E(\Omega_b), \tag{3.21}$$

$$\overline{f}^U = \max\ r^T x \ \text{ subject to } \ x \in E(\Omega_b), \tag{3.22}$$

where $\underline{f}^L$ and $\overline{f}^U$ denote a lower bound of $\underline{f}$ and an upper bound of $\overline{f}$, respectively.

To define a super-set $E(\Omega_b)$, we can apply some duality properties in linear programming. More specifically, let us recall the dual of $\text{ILP}_{\mathbf{b}}$ for a particular $b \in \mathbf{b}$,

$$\max\ b^T y \ \text{ subject to } \ A^T y \le c,\ y \le 0,$$

where $y \in \mathbb{R}^m$ is the vector of decision variables. By the strong duality condition in linear programming, we can describe the optimal solution set of LP(b) by means of the following linear system

$$Ax \leq b, \quad x \geq 0, \quad A^T y \leq c, \quad y \leq 0, \quad c^T x = b^T y.$$

Let us assume, without loss of generality, that b is a vector of decision variables varying within the interval vector $\mathbf{b}$. We can then characterize the optimal set Ω_b as

$$Ax \leq b, \quad x \geq 0, \quad A^T y \leq c, \quad y \leq 0, \quad c^T x = b^T y, \quad b \in \mathbf{b}, \tag{3.23}$$

in variables x, y, b. This leads to a nonlinear programming problem, due to the nonlinear term $b^T y$, which is very difficult to solve. Therefore, we linearize it by using McCormick envelope techniques [90]. Let $[\underline{y}, \overline{y}]$ be an interval enclosure for y. We then apply overestimator and underestimator constraints to linearize the nonlinear constraint $c^T x = b^T y$. The resulting system reads

$$\begin{aligned}
Ax \leq b, \quad x \geq 0, \quad A^T y \leq c, \quad y \leq 0, \quad b \in \mathbf{b}, && \text{(3.24a)}\\
c^T x \leq \underline{y}^T b + \overline{b}^T y - \overline{b}^T \underline{y}, && \text{(3.24b)}\\
c^T x \leq \overline{y}^T b + \underline{b}^T y - \underline{b}^T \overline{y}, && \text{(3.24c)}\\
c^T x \geq \overline{y}^T b + \overline{b}^T y - \overline{b}^T \overline{y}, && \text{(3.24d)}\\
c^T x \geq \underline{y}^T b + \underline{b}^T y - \underline{b}^T \underline{y}, && \text{(3.24e)}
\end{aligned}$$

where (3.24b)-(3.24c) are called overestimators, while (3.24d)-(3.24e) are called underestimators.

System (3.24) is a super-set containing Ω_b. Therefore, we can use it to solve problems (3.21) and (3.22). To compute an interval enclosure $[\underline{y}, \overline{y}]$ for y, we can apply the contractor algorithm in [59]. Briefly, the contractor algorithm is an iterative refinement algorithm. It starts with an enclosure of an optimal set and contracts such an enclosure at each iteration until improvement is insignificant. It runs in polynomial time, and it returns a sufficiently tight interval enclosure for y. We use this algorithm in our experiment in Section 3.7 to get the interval enclosure $[\underline{y}, \overline{y}]$.

3.6.2 Local search algorithm

In this section, we describe a local search algorithm to approximate $\underline{f}$ and $\overline{f}$. Local search is a heuristic method which, given a current feasible solution, tries to improve it by exploring feasible solutions in its neighborhood [120]. Since the returned solution will be a member of Ω_b, the local search algorithm gives a lower bound for $\overline{f}$ (denoted as $\overline{f}^L$) and an upper bound for $\underline{f}$ (denoted as $\underline{f}^U$). Our algorithm starts with an initial solution associated with a given scenario $b \in \mathbf{b}$, then it explores two neighborhoods of the solution, obtained by perturbing b, to find a new solution. If the new solution is better than the current one, then it stores the solution and starts a new iteration. The algorithm proceeds in this way until a stopping condition is met. We discuss our neighborhood structure and details of our algorithm next.

3.6.2.1 Neighborhood structure

We define our neighborhood structure in the scenario space, that is, given an optimal solution of a linear program associated with a particular scenario $b \in \mathbf{b}$, we define two neighborhood structures, namely plus and minus neighborhoods, obtained by perturbing b. Specifically, a plus neighbor (minus neighbor) of a scenario $b \in \mathbf{b}$ is obtained by increasing (decreasing) some components of b by a given quantity. The number of components of b to be perturbed and the amount of perturbation (increment or decrement) are adjustable values. We formally define our neighborhood structures as

$$N^+_{k,h}(b) := \{\tilde{b} \in \mathbf{b} : \tilde{b}_i = b_i + k\phi^+_i,\ \tilde{b}_j = b_j,\ i \in P,\ j \neq i,\ P \in P(h)\}, \tag{3.25}$$

$$N^-_{k,h}(b) := \{\tilde{b} \in \mathbf{b} : \tilde{b}_i = b_i - k\phi^-_i,\ \tilde{b}_j = b_j,\ i \in P,\ j \neq i,\ P \in P(h)\}, \tag{3.26}$$

where ϕ^+_i and ϕ^-_i represent the maximum allowable perturbation of b_i, and they are computed, respectively, as $\phi^+_i = \overline{b}_i - b_i$ and $\phi^-_i = b_i - \underline{b}_i$. Both the plus and the minus neighborhoods of a given scenario are defined depending on two parameters: parameter $k \in (0, 1]$ which is a fraction of $\phi^{\pm}_i$ by which we perturb b_i, and parameter $h \in (0, 1]$ which is a fraction of the total number of components in vector b which we perturb simultaneously. Let us consider the set $\{1, \ldots, m\}$ as the index set of components in vector b. For each value of the parameter h, we denote by $P(h)$ the collection of all the possible subsets of $\{1, \ldots, m\}$ of cardinality $\lfloor h \times m \rfloor$ [4], that is, $P(h) := \{P \subseteq \{1, \ldots, m\} : |P| = \lfloor h \times m \rfloor\}$. Basically, each subset P in $P(h)$ represents a choice of $\lfloor h \times m \rfloor$ components of a current scenario $b \in \mathbf{b}$, which are simultaneously perturbed. Given a scenario $b \in \mathbf{b}$ and a value of h, the number of neighbors in either $N^+_{k,h}(b)$ or $N^-_{k,h}(b)$ is equal to $\binom{m}{\lfloor h \times m \rfloor}$. Finally, for a particular $b \in \mathbf{b}$, a fixed value of k, a fixed value of h, and a set $P \in P(h)$, we determine a neighbor $\tilde{b}$ in either $N^+_{k,h}(b)$ or $N^-_{k,h}(b)$, and denote by $f^+_{b,k,h,P}$ (f^+ for short when no confusion arises) or $f^-_{b,k,h,P}$ (f^- for short when no confusion arises) the value of the outcome function computed on an optimal solution of the linear program associated with $\tilde{b}$.

[4] $\lfloor . \rfloor$ denotes the floor function.

Algorithm 3.1: Local search algorithm to compute $\underline{f}^U$

```
Input: $A$, $\mathbf{b}$, $c$, $r$, $Q$, $V$, max-shakes, threshold
Result: an upper bound on $\underline{f}$
Compute $\underline{f}^U_{int}$.
Set $\underline{f}^U \leftarrow \underline{f}^U_{int}$ and $b \leftarrow \hat{b}_{int}$.
Put $q \leftarrow 1$ and $v \leftarrow 1$.
Set $k \leftarrow Q^{(q)}$ and $h \leftarrow V^{(v)}$.
Randomly select a set $P$ in $P(h)$.
Put $u \leftarrow 0$ and $o \leftarrow 1$.
while $u \leq max\text{-}shakes$ do
    Compute $f^+$ and $f^-$.
    Set $\hat{f} \leftarrow \min\{f^+, f^-\}$ and let $\hat{b}$ be the corresponding right-hand side.
    Determine $improvement \leftarrow \underline{f}^U - \hat{f}$.
    if $improvement \geq threshold$ then
        Set $\underline{f}^U \leftarrow \hat{f}$ and $b \leftarrow \hat{b}$.
        Set $k \leftarrow Q^{(1)}$.
    else if $q < |Q|$ then
        Set $q \leftarrow q + 1$.
        Put $k \leftarrow Q^{(q)}$.
    else if $v < |V|$ then
        if $o \leq \lfloor \frac{1}{h} \rfloor$ then
            Set $o \leftarrow o + 1$.
            Let $\Gamma$ be the set of all indices chosen so far for the current $h$.
            Randomly generate a set $P$ in $P(h)$ such that $P \cap \Gamma = \emptyset$.
            Set $k \leftarrow Q^{(1)}$.
        else
            Update $v \leftarrow v + 1$ and $o \leftarrow 1$.
            Put $h \leftarrow V^{(v)}$.
            Randomly generate a set $P$ in $P(h)$.
            Set $k \leftarrow Q^{(1)}$.
        end
    else
        Update $u \leftarrow u + 1$.
        Set $k$, $h$ to their initial values and set $o$,$q$, and $v$ to 1.
        Randomly generate a scenario $b \in \mathbf{b}$.
        Randomly generate a set $P$ in $P(h)$.
    end
end
```

3.6.2.2 The algorithm

The pseudo-code Algorithm 3.1 shows details of our algorithm to compute $\underline{f}^U$; we can apply a similar scheme to compute $\overline{f}^L$. Line 1 contains input of the algorithm: A, $\mathbf{b}$, c are parameters of the ILP$_{\mathbf{b}}$, r is the coefficient vector of an outcome function, Q is an ordered set of all the selected values k, V is an ordered set of all the selected values h, max-shakes indicates the stopping condition, and threshold represents the minimum acceptable improvement during execution of the algorithm. We denote by $Q^{(q)}$ and $V^{(v)}$ the q-th and v-th elements in the two ordered sets, respectively.

Line 2 computes an initial solution $\underline{f}^U_{int}$, and stores the associated scenario $\hat{b}_{int}$. An initial solution can be computed by solving ORP$_{\mathbf{b}}$ for a randomly generated scenario. The algorithm repeatedly refines an initial solution by using the neighborhood structures defined earlier. Lines 4-5 set the initial values of parameters k and h. Line 6 generates a set P in the collection $P(h)$. Line 7 initiates counter variables. Line 8 checks whether the stopping condition is met. At each iteration, using the neighborhood structures (3.25) and (3.26), lines 9-11 determine a potential incumbent solution, and compute the improvement. If the improvement is acceptable (line 12), lines 13-14 update $\underline{f}^U$ and b , and reset k to its initial value. Otherwise, the algorithm tries the next value of k in Q (lines 15-17). After trying all $k \in Q$, the algorithm chooses a different set P for the current value of h, or it tries different values of h (lines 18-29). Specifically, it first randomly selects a new set P in $P(h)$. Note that lines 19-23 generate different sets P so that they are mutually exclusive. After trying a maximum number $\lfloor \frac{1}{h} \rfloor$ of different sets P in $P(h)$ for a given h, if still no improvement is achieved, then lines 24-29 choose a new value of h in V. The algorithm continues in this way until all h in V are selected. Lines 30-35 apply a shaking step. The aim of this step is to move the search to a different area of the search space. After trying all values of h and k without getting any improvement, a shaking phase starts. In this phase, the input parameters k and h are set to their initial values, counters are re-initialized, a random scenario in $\mathbf{b}$ is generated, and a set P in $P(h)$ is randomly generated. The algorithm proceeds in this way until the stopping condition is met. Finally, it returns the tightest approximation among all.

3.7 Experimentation

Here, we present our computational experiments and related results to evaluate the performance of our approaches. Since there exists no algorithm in the literature to compare our approaches with, then, in addition to our super-set based method and our local search (LS) algorithm, we also use FMINCON, a nonlinear programming solver in MATLAB, to solve the nonlinear formulation of the $\mathrm{ORP}_{\mathbf{b}}$, that is, minimizing (maximizing) $f(x)$ subject to system (3.23). We compare all the methods on two sets of randomly generated instances. The first set, referred to as class 1, is a collection of unique B-stable instances so that the output of our approaches can be compared to the optimal values of the problem (see Proposition 3). The second set of instances, referred to as class 2, is a series of general instances for which the unique B-stability property is not guarantied. Thus, for this set of instances, the optimal values are not known.

3.7.1 Description of problem instances

We generated class 1 instances using the following procedure. First, for a given problem size (m, n) and uncertainty parameter (i.e., interval width) (δ), entries of matrix $A \in \mathbb{Z}^{m\times n}$ were randomly generated in $[-10, 10]$ using uniformly distributed pseudorandom integers. Similarly, vectors $c \in \mathbb{Z}^n, \underline{b} \in \mathbb{Z}^m, r \in \mathbb{Z}^n$ were randomly taken in $[-20, -1], [10, 20]$, and $[-20, 20]$, respectively. Vector $\overline{b}$ was constructed as $\overline{b} = \underline{b} + \delta e$, where $e = (1, ..., 1)^T$ is a vector of ones with the convenient dimension.

To ensure boundedness of the optimal set, we kept entries of the last row of matrix A positive. To have a unique B-stable instance, we found an optimal basis by solving the linear program associated with a randomly chosen scenario, and checked whether the optimal basis is unique and common to all scenarios, i.e., we checked the following conditions[5]

$$\tilde{c}_N^T - \tilde{c}_B^T \tilde{A}_B^{-1} \tilde{A}_N > 0^T,$$
$$\tilde{A}_B^{-1} b_c - |\tilde{A}_B^{-1}| b_\Delta \geq 0,$$

where $b_c := \frac{1}{2}(\bar{b}+\underline{b})$ and $b_\Delta := \frac{1}{2}(\bar{b}-\underline{b})$ denote the center and the radius of the interval $\mathbf{b}$. If both conditions held true, we saved the instance. Otherwise, we started over the process to generate a new instance. In our experimental study, for class 1 instances, we considered the following problem sizes and values for the uncertainty parameter: $m = \{10, 30, 50, 80, 100\}$, $n = \{15, 45, 75, 120, 150\}$ and $\delta = \{0.1, 0.25, 0.5, 0.75, 1\}$. We studied 25 different combinations of m, n, δ, and generated 30 instances for each combination, for a total of 750 instances.

We used a similar procedure to generate class 2 instances, except that the unique B-stability was not required for these instances. For class 2 instances, we considered the following problem sizes and values for the uncertainty parameter: $m = \{10, 30, 50, 80, 100, 200, 300, 400, 500\}$, $n = \{15, 45, 75, 120, 150, 300, 400, 500, 600\}$ and $\delta = \{0.1, 0.25, 0.5, 0.75, 1\}$. We examined 45 different combinations of m, n, δ, and again we generated 30 instances for each combination, for a total of 1,350 instances.

[5]Here, we adopt the unique B-stability conditions for our problem.

3.7.2 Implementation of the algorithms

The input parameters for the local search algorithm were chosen as follows. The two ordered sets Q and V were such that $Q = \{0.1, 0.25, 0.5, 0.75, 1\}$ and $V = \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.5, 1\}$. The max-shake parameter was set equal to one, and the threshold parameter was set equal to 0.001. FMINCON has five stopping criteria namely maximum iterations, maximum function evaluations, step tolerance, function tolerance, and constraint tolerance. We set the maximum iterations and the maximum function evaluations to 300,000, step tolerance to $(1.000E-10)$, and function and constraints tolerances to $(1.000E-6)$. For each problem, we first solved a linear program associated with a randomly generated scenario, and we then took an optimal solution of the linear program as the starting point for the FMINCON. We imposed a time limit of 30 minutes on the solver for each instance such that if the solver cannot normally converge to a solution within 30 minutes, it is terminated and its current solution is returned (if it lies within the feasibility tolerance). For the cases the solver reached one of its internal stopping criteria before reaching the time limit, it started over from a different starting point and continued in this way until either it converged to a solution or it reached the time limit. For the cases for which the solver did not normally converge to a solution within the time limit even after trying multiple starting points, we report the best feasible solution found among all (if any).

Lastly, the experiments were carried out on a workstation with an Intel(R) Core (TM) i7-4790 CPU processor at 3.60 GHz with 32.00 GB of RAM. All the methods were coded in MATLAB(R2019b), using IBM ILOG CPLEX 12.9 for solving linear programs.

Table 3.5: Results related to the computation of $\underline{f}$ on class 1 instances (average gap and average running time)

input			average gap			average time (sec)		
m	n	δ	LS $\underline{f}^{U_1}$	FMINCON $\underline{f}^{U_2}$	super-set $\underline{f}^{L}$	LS $\underline{f}^{U_1}$	FMINCON $\underline{f}^{U_2}$	super-set $\underline{f}^{L}$
10	15	0.1	0.0001	0.0035	0.0122	0.3214	0.4921	0.0016
10	15	0.25	0.0001	0.0016	0.1530	0.3212	0.2684	0.0007
10	15	0.5	0.0001	0.0186	0.1618	0.3411	0.3133	0.0007
10	15	0.75	0.0001	0.0052	0.1411	0.3385	0.3034	0.0007
10	15	1	0.0003	0.0190	0.9195	0.3437	0.3516	0.0007
30	45	0.1	0.0004	0.0057	0.1594	0.7657	1.6774	0.0020
30	45	0.25	0.0029	0.0063	1.0221	0.8666	1.6139	0.0020
30	45	0.5	0.0003	0.0094	0.5711	0.9188	1.5557	0.0020
30	45	0.75	0.0004	0.0085	1.5103	0.9019	1.5185	0.0019
30	45	1	0.0043	0.0175	1.4701	0.9346	1.6153	0.0020
50	75	0.1	0.0013	0.0045	0.3009	1.4564	3.2491	0.0042
50	75	0.25	0.0018	0.0148	1.0008	1.5302	3.8956	0.0041
50	75	0.5	0.0017	0.0084	0.9686	1.6549	4.3446	0.0040
50	75	0.75	0.0066	0.0688	2.4971	1.6501	3.3910	0.0040
50	75	1	0.0010	0.0640	4.3327	1.6596	3.6361	0.0040
80	120	0.1	0.0020	0.0381	1.0832	2.9608	12.1033	0.0106
80	120	0.25	0.0142	0.0618	2.3559	3.1187	12.3475	0.0102
80	120	0.5	0.0030	0.0227	1.4559	3.2310	12.5632	0.0102
80	120	0.75	0.0018	0.0514	2.2257	3.2231	13.2636	0.0100
80	120	1	0.0039	0.0510	2.3866	3.3486	11.3097	0.0099
100	150	0.1	0.0030	0.0646	1.5206	4.2595	26.5560	0.0163
100	150	0.25	0.0034	0.0222	1.6112	4.5020	22.6072	0.0158
100	150	0.5	0.0117	0.0504	2.6134	4.7064	23.4783	0.0154
100	150	0.75	0.0153	0.0811	2.4725	4.7758	22.0584	0.0148
100	150	1	0.0063	0.0563	3.6750	4.5380	23.5457	0.0151

3.7.3 Analysis of the results

In this section, we only discuss the results related to $\underline{f}$. The analysis of the results for $\overline{f}$ led to similar conclusions, so we do not report them in the paper. Table 3.5 shows the results related to the computation of $\underline{f}$ on class 1 instances, for which the optimal value can be computed by solving a linear program (see Proposition 3). Each number in the table is an average of the results obtained on 30 instances. In the table, the first three columns show the input parameters, and the following six columns report the results of the solution approaches. We recall that the super-set based method returns a lower bound for $\underline{f}$ (columns $\underline{f}^L$), while the local search algorithm and FMINCON return an upper bound (columns $\underline{f}^{U_1}$ and $\underline{f}^{U_2}$, respectively). The gap of an approximate value $\hat{\underline{f}}$ from the optimal value is computed by $|\frac{\hat{\underline{f}}-\underline{f}}{\underline{f}}|$. Hence, lower values correspond to better performance of the approach. For each method, the table reports the average gap and the average running time (in seconds).

The local search algorithm converges fast to a very tight upper bound (with a maximum average gap of 1.53% and a maximum average running time of 4.78 seconds) for all the problem sizes and all the values of the uncertainty parameter δ. FMINCON returns a reasonable upper bound (with a maximum average gap of 8.11%), but it takes significantly longer time than the local search algorithm to converge (with the average running time ranges between 0.27 of a second and 26.56 seconds). Although calculating $\underline{f}^L$ is fast, its gap from the the optimal value, with the exception of small size instances and low uncertainty, is significant.

For class 2 instances, given the poor performance of the super-set based method, we only focus on the results obtained from the local search and the solver. As noted earlier, the unique B-stability property is not guarantied in class 2 instances, and as a result we are not able to solve $\mathrm{ORP}_{\mathbf{b}}$ to optimality using existing methods. We here compare the local search and FMINCON against each other. For instances where the local search outperforms the solver, i.e. $\underline{f}^{U_1} < \underline{f}^{U_2}$, we calculate the gap as $|\frac{\underline{f}^{U_1}-\underline{f}^{U_2}}{\underline{f}^{U_2}}|$, while for instances where the solver returns a better solution than the local search, namely $\underline{f}^{U_2} < \underline{f}^{U_1}$, the gap is determined by $|\frac{\underline{f}^{U_2}-\underline{f}^{U_1}}{\underline{f}^{U_1}}|$. Additionally, the following measure gives a weighted average gap (WAG) for each method. Specifically, it applies both the average gap and the number of times an algorithm outperforms the other (denoted by NI), and reads

$$\text{WAG} = \frac{(\text{NI}) * (\text{average gap})}{\text{total number of instances}}.$$

Thus, the higher the WAG value, the better the performance. Table 3.6 reports the results corresponding to the computation of $\underline{f}$ for class 2 instances. The first three columns show the input data. Columns 4 and 5 give the frequency of times the local search outperforms FMINCON and the weighted average gap, respectively. Similarly,

Table 3.6: Results related to the computation of $\underline{f}$ on class 2 instances.

input			$\underline{f}^{U_1} < \underline{f}^{U_2}$		$\underline{f}^{U_2} < \underline{f}^{U_1}$		average time (sec)	
m	n	δ	freq.	WAG	freq.	WAG	LS	FMINCON
10	15	0.1	22	0.0024	8	0.0000	0.3130	0.4605
10	15	0.25	21	0.0074	9	0.0000	0.3190	0.4511
10	15	0.5	21	0.0045	9	0.0084	0.3301	0.5361
10	15	0.75	19	0.0138	11	0.0002	0.3356	0.3225
10	15	1	21	0.0526	9	0.0081	0.3536	0.3164
30	45	0.1	19	0.0107	11	0.0003	0.8241	1.3156
30	45	0.25	19	0.0085	11	0.0010	0.8808	1.3838
30	45	0.5	20	0.0397	10	0.0028	0.8978	1.4990
30	45	0.75	17	0.0514	13	0.0084	0.9450	2.4027
30	45	1	19	0.0390	11	0.0059	0.9791	1.8116
50	75	0.1	17	0.0031	13	0.0003	1.4768	4.8394
50	75	0.25	15	0.0061	15	0.0089	1.5200	4.4078
50	75	0.5	16	0.3897	14	0.0028	1.7290	4.0354
50	75	0.75	16	0.0360	14	0.0067	1.7371	4.2858
50	75	1	14	0.0585	16	0.1042	1.8498	4.9498
80	120	0.1	23	0.0141	7	0.0013	3.0263	13.7443
80	120	0.25	17	0.0112	13	0.0364	3.4131	14.9026
80	120	0.5	19	0.1273	11	0.0750	3.6254	13.2121
80	120	0.75	16	0.0658	14	0.1080	3.5492	15.0886
80	120	1	15	0.0841	15	0.0504	3.6229	19.0471
100	150	0.1	25	0.0143	5	0.0022	4.6070	30.4261
100	150	0.25	23	0.1741	7	0.0109	4.9523	27.6289
100	150	0.5	18	0.0413	12	0.0147	5.3517	31.3020
100	150	0.75	18	0.0816	12	0.0337	5.6269	44.5924
100	150	1	10	0.0904	20	0.0583	5.2683	38.2789
200	300	0.1	26	0.1033	4	0.0078	18.8387	226.2555
200	300	0.25	15	0.2028	15	0.0120	20.0275	242.7648
200	300	0.5	14	0.0415	16	0.1506	21.6897	298.3727
200	300	0.75	9	0.0318	21	0.5077	20.1736	231.7975
200	300	1	11	0.0254	19	0.2233	20.3656	203.9699
300	400	0.1	18	0.0655	12	0.0150	43.7986	1,400.9399
300	400	0.25	18	0.1182	12	0.0424	46.3638	1,601.2650
300	400	0.5	4	0.0539	26	0.2600	48.3374	1,536.8595
300	400	0.75	6	0.0462	24	0.1886	39.8791	1,413.8520
300	400	1	5	0.0434	25	1.0230	40.4988	1,351.4880
400	500	0.1	22	0.1410	8	0.0196	76.7331	1,740.2052
400	500	0.25	17	3.3220	13	0.0188	76.2029	1,800
400	500	0.5	9	0.0247	21	0.4430	76.8172	1,760.1362
400	500	0.75	3	0.0063	27	0.8576	74.1061	1,784.4656
400	500	1	2	0.0027	28	1.3027	70.8703	1,722.2246
500	600	0.1	22	0.0889	8	0.0274	124.1661	1,768.2018
500	600	0.25	12	0.1292	18	0.0410	123.9820	1,800
500	600	0.5	4	0.0224	26	0.3193	110.2257	1,770.3525
500	600	0.75	4	0.0211	26	1.6025	112.0062	1,800
500	600	1	2	0.0048	28	0.9238	108.6081	1,800

the following two columns represent the same attributes for when FMINCON outperforms the local search. The last two columns indicate the average running times for each method.

Our results in Table 3.6 suggest that the local search outperformed FMINCON on 683 instances out of 1,350 instances. Moreover, the weighted average gap of the local search is larger than that of FMINCON for 28 combinations of m, n, δ out of the total of 45 combinations. FMINCON tends to return a better weighted average gap for instances with larger sizes and uncertainty parameters, but it also takes a much longer time to converge to a solution, see, for example, the instances with $m = 300$, $n = 400$, and $\delta = 1$. For this particular case, FMINCON took 1,351.49 seconds on average to converge, while our local search converged, on average, in 40.5 seconds. From the table, we can observe that the WAG measure ranges between 0.0024 and 3.3220 for the local search and between 0.0000 and 1.6025 for FMINCON. We also see that the computation time of FMINCON grows faster compared to that of the local search, with a maximum average running time of 1,800 seconds against that of 124.17 seconds for the local search.

3.8 Case study: Healthcare access measurement

Here, we show an application of $\text{ORP}_{\mathbf{b}}$ when an outcome function is used to measure spatial access to healthcare services. We first introduce a linear program which has been recently proposed in the literature to derive a matching between patients and providers. We then use our approach to evaluate how uncertainty in input data influences spatial access to healthcare services, and discuss how the results of our approach can be used for more reliable decision making.

Model 1. Modeling access to primary care.

$\min \sum_{i\in T, j\in W} g d_{ij} x_{ij}$ subject to			$\rightarrow$ Total distance is minimized.
Coverage constraints:			
$\sum_{j\in W} x_{ij} \le e_i$	$\forall i \in T,$	(C1)	$\rightarrow$ The assignment does not exceed population in need in census tract i.
$\sum_{i\in T, j\in W} x_{ij} \ge \alpha E,$		(C2)	$\rightarrow$ The assignment covers as much population as possible within the national access policy.
Accessibility constraints:			
$\sum_{j\in W: d_{ij}\ge d_{max}} x_{ij} = 0$	$\forall i \in T,$	(C3)	$\rightarrow$ Patients are not assigned to too far providers.
$\sum_{j\in W: d_{ij}\ge d_{max}^{mob}} x_{ij} \le m_i e_i$	$\forall i \in T,$	(C4)	$\rightarrow$ Patients that own a vehicle can travel further than patients without a vehicle.
Availability constraints:			
$\sum_{i\in T} g x_{ij} \le c_j^{max}$	$\forall j \in W,$	(C5)	$\rightarrow$ Providers' maximum caseload is not exceeded.
$\sum_{i\in T} g x_{ij} \ge c_j^{min}$	$\forall j \in W,$	(C6)	$\rightarrow$ Providers are assigned a minimum caseload to stay in practice.
Non-negativity constraints:			
$x_{ij} \ge 0$	$\forall i \in T,\ j \in W.$		

3.8.1 Optimization model and outcome function

Optimization models used to quantify potential spatial access to healthcare mimic the interactions between two sets of actors in the system: the population in need of service within each geographical area or community (e.g., census tract level), namely e_i with $i \in T$, and the network of provider locations $j \in W$. Model 1 is a simplified version of the mathematical formulation proposed in the literature [50, 103] to determine a matching between the population in need of healthcare services and providers providing them. The matching is determined to minimize the total distance traveled at the system level under a set of constraints: (i) *coverage constraints* match as many people in need as possible; (ii) *accessibility constraints* ensure the matching takes into account modes of transportation and Health Resources Services Administration recommendations on the maximum allowed distance for matching; (iii) *capacity constraints* account for the maximum and minimum providers' caseload to stay in practice.

The decision variables x_{ij} in the model determine the number of patients in a census tract $i \in T$ assigned to a specific provider location $j \in W$. Parameters of the model include:

- g: number of yearly visits required by a patient,
- e_i: population size in census tract i in need of healthcare services,
- d_{ij}: travel distance between the centroid of census tract i and provider location j,
- E: total population in the system in need of healthcare services,
- α: percentage of the population which should be assigned to a provider,
- d_{max}: maximum allowed distance between a patient and the assigned provider according to the Health Resources Services Administration recommendations,
- d^{mob}_{max}: maximum distance we assume that people without a vehicle are willing to travel to reach the assigned provider,
- m_i: percentage of population in census tract i that owns a vehicle,
- c^{max}_j (c^{min}_j): maximum (minimum) provider's caseload in location j.

For our analysis, we consider an interval version of Model 1 obtained by allowing parameters c^{max}_j to vary within a given interval. Specifically, we assume that the availability constraints (C5) in the model are of the form

$$\sum_{i \in T} g x_{ij} \leq [\lambda c^{max}_j, \beta c^{max}_j] \quad \forall j \in W,$$

where λ and β are the maximum and the minimum perturbations from the nominal values c^{max}_j for $j \in W$, respectively. Note that the resulting intervals vary independently. Such uncertainty in the capacity of a provider can be due to increasing and/or decreasing personnel, overtime or days-off of providers, and inaccurate estimations of the capacity, among others.

Access measures are formulated as linear functions of an optimal assignment derived from an optimization model [103]. For this illustrative example, we consider the access measure $f_i(x)$ defined as the average distance traveled by patients in a given census tract i to reach the assigned provider, which is formally defined as

$$f_i(x) = d_{max} + \frac{1}{e_i} \sum_{j \in W} (d_{ij} - d_{max}) x_{ij} \quad \forall i \in T.$$

The above measure returns the weighted average of the distance traveled by patients in each census tract. We assume that for those patients who are not assigned to a provider, $f_i(x)$ is equal to d_{max}. Thus, the access measure ranges from 0 to d_{max}.

The resulting estimates can be used by policy makers to identify where the communities with the greatest need for improvement are, so that they can be targeted with additional resources, including new providers or facilities, transportation services improvement, tele-health service development, etc.

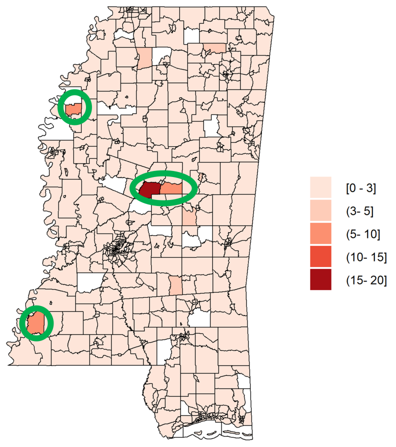
3.8.2 Case study

We illustrate our analysis to quantify access to the primary care service for children in the State of Mississippi in the United States, for a total of 637 census tracts and 897 provider locations. Providers' practice location addresses are obtained from the 2013 National Plan and Provider Enumeration System (NPPES). The patient population is aggregated at the census tract level. We used the 2010 SF2 100% census data and the 2012 American Community Survey data to compute the number of children in each census tract along with information on ownership of cars, to estimate = access to private transportation means. We set $d_{max} = 25$ miles, $d_{max}^{mob} = 10$ miles, $\alpha = 0.85$, and $g = 2$ (see [49] for further details on the input parameters). The resulting model contains 63,573 variables and 3,706 constraints. To account for uncertainty in c_j^{max}, we set $\lambda = 0.8$ and $\beta = 1.2$.

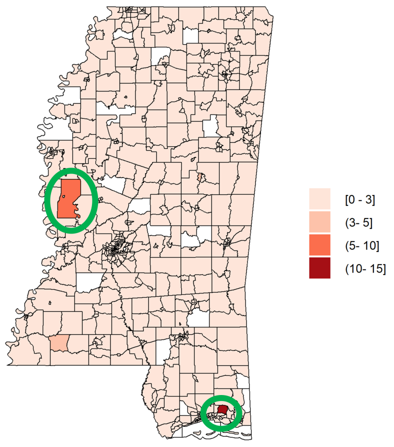
3.8.3 Importance of quantifying sensitivity to data perturbations

Failing to consider uncertainty in the input parameters may significantly affect the choice of which census tracts to target for possible interventions. To elaborate further, we compared the results of Model 1 on two different realizations of interval data, referred to as realizations 1 & 2. Figure 3.5a shows the difference in the access measures obtained in the two optimization runs (corresponding to realizations 1 & 2) . Darker regions represent higher differences, that is, census tracts where the estimate of the access measure is more unstable. The circled census tracts are those for which the resulting access measure changes more than 5 miles between the two runs, implying that some census tracts may be considered having high or low level access depending on which realization of the data is considered. Consider now Figure 3.5b where the difference in the access measures, obtained for two different additional realizations (referred to as realizations 3 & 4) of the parameter c_j^{max}, is shown. The comparison between Figures 3.5a and 3.5b tells two different stories, showing completely different sets of census tracts for which the access measure seems more unstable.

In this sense, quantifying sensitivity of the access measure to data perturbations would be crucial for reliable decision making. Such an analysis would indeed reveal: (i) census tracts that are *certainly* in need of a targeted intervention (e.g., those census tracts for which the access measure is high and not sensitive to data perturbations), and (ii) census tracts that are *certainly not* in need of any intervention (e.g., those census tracts for which the access measure is low and not sensitive to data perturbations). It would also help to determine census tracts that may fall, due uncertainty in the data, in either one of the two categories, and for which, therefore, a deeper investigation might be needed. By solving ORP$_\mathbf{b}$ in this context, we can assess such a quantification. Additionally, we are able to answer questions relevant for policy making, including:



(a) Realizations 1 & 2



(b) Realizations 3 & 4

Figure 3.5: Difference in the access measures considering four random realizations of the input parameters.

- Q1: Given the current primary care resources, what are the minimum and maximum access levels for each census tract?
- Q2: What are the census tracts with the highest (lowest) variability in the access measures?
- Q3: What is the percentage of the census tracts where the access level is higher (lower) than a given threshold for all the possible realizations of the data?

We applied our local search algorithm to solve $\mathrm{ORP}_{\mathbf{b}}$ in this context, and addressed the above questions.

3.8.4 Implementation of algorithms

The outcome function (i.e., the access measure) is associated with each census tract. Hence, we applied our local search algorithm once for each outcome function (total of 637 functions). Zheng et al. [136] used the Monte Carlo approach to evaluate sensitivity of the access measure to uncertainty in the input data. Therefore, we compare the results of our approach with those returned by the Monte Carlo approach.

For the local search algorithm, we defined $Q = \{0.25, 0.5, 0.75, 1\}$. Due to the large size of the problem and the structural dependencies among the decision variables [136], defining an ordered set V and randomly choosing constraints, whose right-hand sides are perturbed simultaneously, would not be very efficient. Thus, we defined a set $V(i)$ for each given census tract i as $V(i) = \{H_1(i), H_2(i), H_3(i)\}$ for all $i \in T$, where $H_l(i)$, $l = 1, 2, 3$, are predefined sets of constraints associated with census tract i. Note that for this specific application, each constraint to be perturbed corresponds to a provider j whose max capacity parameter (c_j^{max}) is perturbed from its nominal value. The first set of constraints to be explored corresponds to providers who are not too far from the census tract under study, that is, $H_1(i) := \{j \in W : d_{ij} \leq 50\}$. The second set of constraints to be explored are those constraints corresponding to providers who do not correspond to constraints in $H_1(i)$ and who are not too far from census tracts which are neighbors of the census tract under study. Specifically, we defined two census tracts to be neighbors if the distance between their centroids is less than 50 miles. Given a census tract i, let us denote the set of neighboring census tracts as the set $A(i) = \{a \in T : d_{ia} \leq 50\}$. The second set of constraints is then defined as $H_2(i) := \{j \in W : d_{aj} \leq 50, \forall a \in A(i)\} \backslash H_1(i)$. Finally, the last set $H_3(i)$ consists of the remaining providers, that is, $H_3(i) := W \backslash \{H_1(i) \cup H_2(i)\}$.

We set the maximum number of shakes to 1 and the minimum acceptable improvement to 0.1. The number of iterations for the Monte Carlo approach was set equal to 100, which is the maximum number of linear programs solved by the local search algorithm among all the runs.

3.8.5 Analysis of the results

Monte Carlo simulation is a simple approach to compute the maximum and the minimum values of the access measure for each census tract; however, in this context, it might lead to a severe underestimation of the overall quantification. To show this, we computed the range of the resulting access measure for each census tract using both the local search algorithm and the Monte Carlo approach. The range is computed as the difference between the maximum and the minimum of the access measure for each census tract. The difference in the results is shown in Figures 3.6 and 3.7. Specifically, Figure 3.6 shows the number of census tracts for which the range of the access measures is within 2 and 20 miles for the two approaches. Results obtained from the Monte Carlo approach show that the access range of 28 census tracts out of 635 census tracts varies between 4 and 20 miles, where that of 12 census tracts varies between 8 and 20 miles. However, the local search algorithm reveals that the access range of 89 census tracts varies between 4 and 20 miles, where that of 47 census tracts varies between 8 and 20 miles. It is noteworthy that Figure 3.6 does not represent census tracts with the access range of less than 2 miles.

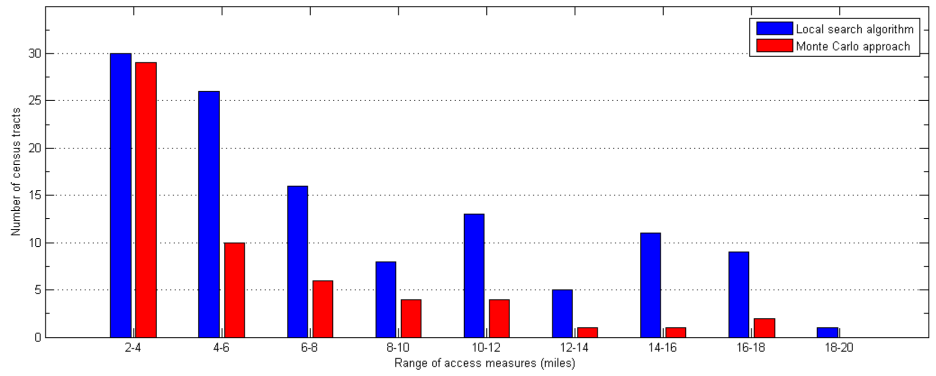


Figure 3.6: Distribution of the census tracts for which the access range varies between 2 and 20 miles for the two approaches (i.e., Monte Carlo approach and the local search algorithm).

Figure 3.7 depicts the map of the difference in the ranges obtained comparing the two approaches. Darker census tracts are those for which the Monte Carlo approach severely underestimates sensitivity of the access measures, that is, those census tracts for which the difference between the range estimated by the Monte Carlo approach and the range estimated by the local search algorithm is greater than 15 miles. From Figures 3.6 and 3.7, it is evident that the Monte Carlo approach is not a right tool

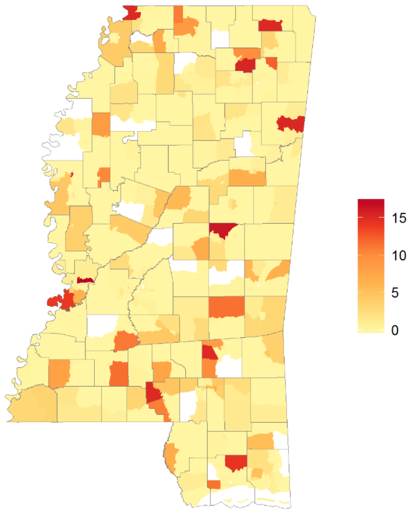
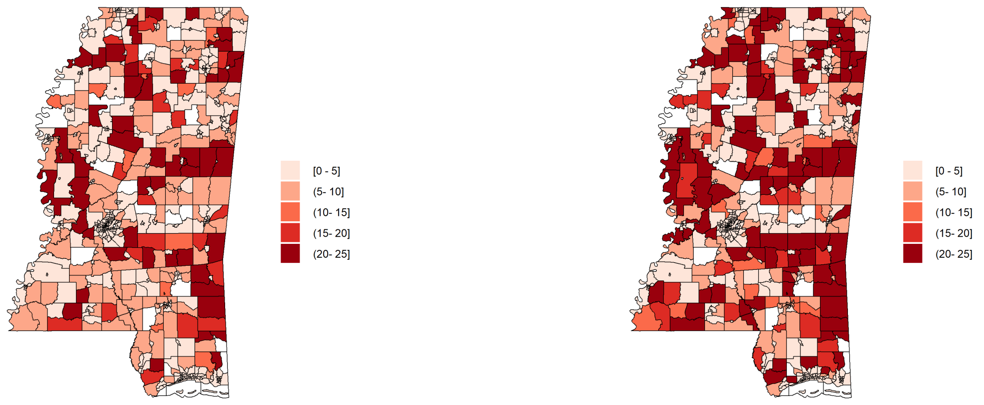


Figure 3.7: Difference between the access ranges estimated by the Monte Carlo approach and those estimated by the local search algorithm.

to quantify sensitivity of the access measure to uncertainty in the data. Its use to answer the questions Q1-Q3 would lead to a severe underestimation. Hence, in what follows, we only focus on the results obtained from our local search algorithm.

Figures 3.8a and 3.8b show the lower and the upper limits of the access measure for each census tract (Q1), and Figure 3.9 shows the range of the access measure for each census tract (Q2). Darker areas in Figure 3.9 are those census tracts where the range of the access measure is greater than 10 miles, which corresponds to 39 census tracts out of 635 (i.e., 6% of the total). Table 3.7 and Figure 3.10 can be used to address question Q3. The table shows the distribution of the minimum and maximum of access within the state among all the census tracts. Figure 3.10a divides the census tracts in two groups according to the value of their minimum access level: dark (light) tracts have a minimum value which is greater (less than or equal to) 10 miles. Figure 3.10b divides the census tracts in two groups according to the value of their maximum access value: dark (light) tracts have a maximum access value which is greater than (less than or equal to) 5 miles. According to Table 3.7, 13% of the census tracts have a minimum value of access which is greater than 10 miles. In other words, the population in these census tracts always travel on average at least 10 miles to reach the assigned provider. These census tracts are the dark regions in Figure 3.10a. On the other hand, 64% of the census tracts (the maximum access level column in Table 3.7) are such that the corresponding population never travel more than 5 miles to reach the assigned provider. These census tracts are the light regions in Figure 3.10b. These findings are important for decision makers to prioritize interventions. Indeed, for example, the dark color census tracts in Figure 3.10a depict those census tracts which are surely in need for targeted actions to improve their access to healthcare services because they were identified by accounting for all the possible realizations of the uncertain data, while the light color census tracts in Figure 3.10b have a good access to healthcare services over all the possible realizations of the uncertain data; hence, they are unlikely to be the object of targeted interventions.



(a) Minimum access level

(b) Maximum access level

Figure 3.8: Minimum and maximum value of the access measures for each census tract.

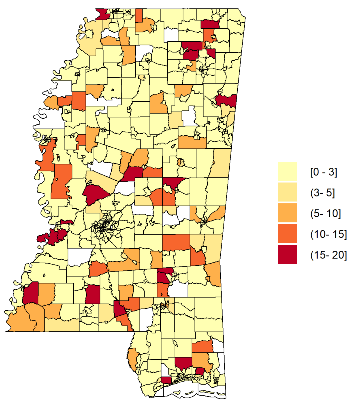
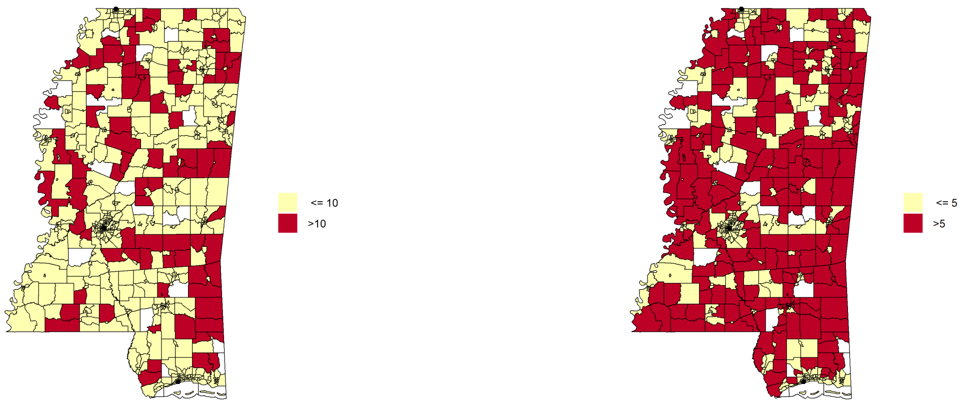


Figure 3.9: Range of the access measure for each census tract.

Table 3.7: Distribution of census tracts according to the minimum and maximum access levels (for different access ranges).

access (mile)	minimum access level	maximum access level
0-5	69%	64%
5-10	18%	14%
10-15	2%	2%
15-20	3%	4%
20-25	8%	15%



(a) Minimum access level (b) Maximum access level

Figure 3.10: Classification of the census tracts according to their minimum and maximum access levels.

3.9 Conclusions

We formulated and studied the *outcome range problem* in the context of interval linear programming. Our problem aims at quantifying unintended consequences of optimal decisions made in an uncertain environment. The problem is particularly relevant for government agencies, public health decision makers, policy makers, city managers and other stakeholders who make decisions that have differential impacts on different communities and sub-populations, and we showed this on a real case study related to healthcare access measurement. In this paper, we gave a very general definition of the outcome range problem, and addressed a specific version of it for which we assessed the computational complexity and studied some theoretical properties. We then offered two approximation methods. Our proposed local search algorithm seems promising in computing a cheap but tight approximation of the problem. In contrast, the proposed super-set based method does not return a tight approximation; thus, there is room for improvement. We tested our methods on two sets of randomly generated instances and on a real case instance. We plan to further explore theoretical properties and solution methods for a more general version of the problem.

CHAPTER 4

HOW TO QUANTIFY OUTCOME FUNCTIONS OF INTERVAL-VALUED LINEAR PROGRAMS

4.1 Introduction

Throughout the years, linear programming has been broadly used to formulate and solve real-world problems. In traditional linear programming, one always assumes that parameters are known exactly. However, in practice, it is quite common to confront imprecise input data which might impair the obtained results and consequently the decisions made upon them. So far several different approaches have been developed to incorporate uncertainty of input parameters into mathematical models. In this paper, we adopt the approach of interval linear programming (ILP), where all (or some) parameters are assumed to vary within given real-valued intervals.

Interval linear programming has been applied in several application areas, such as transportation problems where the supply and demand can vary within a-priori known intervals [27, 36, 75, 129], matrix games with interval-valued payoffs [82, 85], portfolio selection problems with interval approximations of expected returns [78, 79], project management with interval task durations [38], and environmental [28, 84] and waste [124] management problems with interval input parameters.

The main interest of interval linear programming is studying overall properties of an interval problem, considering all possible scenarios of the interval data [58, 111, 113]. Different topics have been studied in interval linear programming (see [58] for a thorough survey on the topics). The two main problems discussed in the ILP literature are (i) finding the range of optimal values of the objective function among all optimal values obtained over all data perturbations [30, 46, 57, 97, 111] and its tight approximation [62, 93] and (ii) describing the set of all optimal solutions resulting from all the possible realizations of the interval data [2, 44, 47] and its inner [65] and outer [59] approximations. Recently, Mohammadi and Gentili [94] formulated the *outcome range problem* where the goal is to find the range of an additional function of interest (namely, an outcome function) over the set of all possible optimal solutions of an interval linear program.

Outcome functions are extra functions of interest associated with the set of optimal solutions of an optimization problem. So far, there has been several studies quantifying outcome functions over an optimal solution of a linear program. For example, Nobles et al. [103] and Gentili et al. [48, 49, 50] formulated some linear programs to match patients and healthcare providers, and then they quantified some outcome functions over the results of the optimization models to evaluate spatial access to healthcare services. As another example, in a transportation problem, an outcome function could be an environmental cost function assessing the environmental impact of an optimal transportation plan on the surrounding areas [94]. Similarly, the notion of the outcome function can be of great interest in the telecommunications network design and evacuation planning [136]. Quantifying outcome functions was also mentioned in [81] as a motivation of addressing uncertainty in linear programming. Outcome functions are also relevant in multiobjective optimization problems where it is common to optimize an extra function over the set of all efficient solutions of a multiobjective program to get a preferred solution, see, for example, [16, 121, 131] and references therein.

Generally speaking, linear optimization problems under right-hand side uncertainty often arise in practical problems and have been subject of numerous studies, see [3, 9, 40, 89, 107, 114] and references therein. Hence, in this paper, we address a special class of the outcome range problem where we consider a linear real-valued outcome function associated with an equality-constrained linear program with interval right-hand sides. We first review some preliminaries and formally define the problem (Section 4.2). We then discuss the computational complexity of the problem (Section 4.3). We also study some theoretical aspects, related to property and geometry of scenarios, of this special class of the problem (Section 4.4). There are some overlaps between the outcome range problem and other known problems such as the optimal value range problem, bilevel optimization, and multiobjective optimization. Here, we clarify the relation between the problems, and discuss different reformulations of the outcome range problem (Section 4.5). Furthermore, given the hardness of computation of the outcome range problem, we offer some heuristics to solve the problem (Section 4.6). In particular, we present an outer approximation for our problem using the reformulation-linearization technique (Section 4.6.1). We also design two inner approximation algorithms to approximate the range of an outcome function: our first algorithm is based on the gradient-restoration algorithm [91] where the goal is to improve a current incumbent solution by moving alternately between feasible and infeasible solutions at each iteration (Section 4.6.2), and for the second algorithm, we develop a guided optimal bases search (Section 4.6.3). We test our methods on three test beds (Section 4.7), and finally in the last section, we summarize our findings and discuss some future directions.

4.2 Problem definition

Let us start by introducing some notation. Consider a linear program

$$z(A,b,c) := \min\ c^T x\ \text{ subject to }\ x \in \mathcal{M}(A,b), \tag{4.1}$$

where $c \in \mathbb{R}^n$ is the objective vector, and $\mathcal{M}(A,b)$ denotes the feasible region described by some linear constraints with matrix coefficient $A \in \mathbb{R}^{m\times n}$ and right-hand side vector $b \in \mathbb{R}^m$. Linear program (4.1) essentially can be in one of the following forms

$$z(A,b,c) = \min\ c^T x\ \text{ subject to }\ Ax = b,\ x \geq 0, \tag{I}$$

$$z(A,b,c) = \min\ c^T x\ \text{ subject to }\ Ax \leq b, \tag{II}$$

$$z(A,b,c) = \min\ c^T x\ \text{ subject to }\ Ax \leq b,\ x \geq 0. \tag{III}$$

An interval matrix is a set of matrices

$$\mathbf{A} = [\underline{A}, \overline{A}] := \{A \in \mathbb{R}^{m\times n} :\ \ \underline{A} \leq A \leq \overline{A}\},$$

where $\underline{A}, \overline{A} \in \mathbb{R}^{m\times n}$ are given, and comparing matrices is understood componentwise. We denote by $\mathbb{IR}$ the set of all real intervals. We use similar notation for interval vectors, i.e., we consider them as one column interval matrices. Throughout the paper, bold symbols stand for interval matrices and vectors.

Given $\mathbf{A} \in \mathbb{IR}^{m\times n}$, $\mathbf{b} \in \mathbb{IR}^m$, and $\mathbf{c} \in \mathbb{IR}^n$, we define an interval linear program as a family of linear programs with $A \in \mathbf{A}$, $b \in \mathbf{b}$, and $c \in \mathbf{c}$, i.e.,

$$\{\min\ c^T x\ \text{ subject to }\ x \in \mathcal{M}(A,b) :\ A \in \mathbf{A},\ b \in \mathbf{b},\ c \in \mathbf{c}\}.$$

In short, we can write the preceding as

$$\min\ \mathbf{c}^T x\ \text{ subject to }\ x \in \mathcal{M}(\mathbf{A}, \mathbf{b}).$$

We refer to any triplet (A, b, c) with $A \in \mathbf{A}$, $b \in \mathbf{b}$, and $c \in \mathbf{c}$ as a scenario, and thus LP (4.1) is associated with the scenario (A, b, c). We denote by $S(A, b, c)$ the set of optimal solutions of (4.1), if any, admitting a finite $z(A, b, c)$. The set of all optimal solutions to an ILP problem, namely the optimal set, is

$$\Omega := \bigcup_{A \in \mathbf{A},\, b \in \mathbf{b},\, c \in \mathbf{c}} S(A, b, c).$$

Now consider an extra function $f\colon \mathbb{R}^n \to \mathbb{R}$ where $f(x) = r^T x$ with $r \in \mathbb{R}^n$.[1] The outcome range problem is the problem of finding the maximum and the minimum vaules of $f(x)$ over the optimal set (Ω) [94], that is,

$$\begin{aligned} \underline{f} &:= \min\ f(x)\ \text{ subject to }\ x \in \Omega, \\ \overline{f} &:= \max\ f(x)\ \text{ subject to }\ x \in \Omega. \end{aligned}$$

In this sense, the range $[\underline{f}, \overline{f}]$ is called the outcome function range.

It is known that transformations such as splitting equalities into inequalities or imposing non-negativity on unrestricted variables, used in classic linear programming, are not applicable in ILP problems in general, due to the so-called dependency problem [47, 63]. Hence, the three main forms of ILP problems, that is, ILP problems described in the forms of (I)-(III), are usually addressed separately in the literature [58]. We here narrow down our study on a special class of ILP problems with interval right-hand sides. This special case contains a large class of problems in practice where often A and c are fixed and uncertainty only happens in the right-hand side vector

[1]In general, outcome functions can be in any form, i.e., they need not be necessarily linear.

b. This is particularly true for network optimization problems such as transportation problems, maximum flow problems, minimum-cost flow problems, multicommodity network flow problems, etc [8, 98]. We formally consider

$$\min\ c^T x \ \text{ subject to } \ Ax = \mathbf{b},\ x \geq 0, \tag{4.2}$$

where $A \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{IR}^m$, and $c \in \mathbb{R}^n$. By LP(b) we mean the linear program associated with ILP (4.2) for a given $b \in \mathbf{b}$. Similar to the general form of the ILP problems, we denote by $S(b)$ and $z(b)$ the set of optimal solutions (if any) of LP(b), and the optimal value of LP(b) for a particular $b \in \mathbf{b}$, respectively. We refer to the optimal set of (4.2) as Ω_b. Hereinafter, we only focus on solving the following problems

$$\underline{f} = \min\ f(x) \ \text{ subject to } \ x \in \Omega_b, \tag{4.3}$$

$$\overline{f} = \max\ f(x) \ \text{ subject to } \ x \in \Omega_b. \tag{4.4}$$

We refer to this special class of the outcome range problem as ORP$_\mathbf{b}$.

Observation 4.1. *By [43, 44], we know that Ω_b is path connected. It is bounded if $S(b)$ is non-empty and bounded for some $b \in \mathbf{b}$. Therefore, in the case of ORP$_\mathbf{b}$, we can say that $f(x)$ is continuous in $[\underline{f}, \overline{f}]$ if Ω_b is bounded.*

Note that, in practice, we are usually interested in finite $\underline{f}$ and $\overline{f}$. In what follows, we assume that this is the case. The following formally states our assumption.

Assumption 4.1. *We assume that there exists $b \in \mathbf{b}$ such that LP(b) admits a finite optimal value.*

Our assumption implies that LP(b) is either infeasible or admits a finite optimal value for any $b \in \mathbf{b}$. Now we discuss an example of the ORP$_\mathbf{b}$ below.

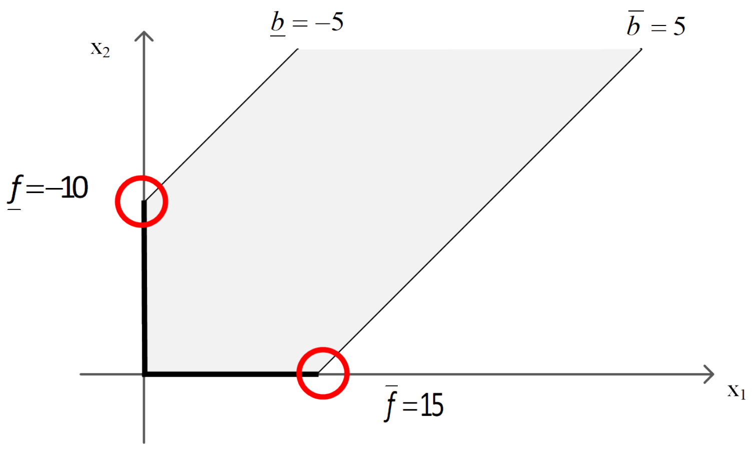


Figure 4.1: (Example 4.1) Union of all the feasible solutions in grey; optimal set in bold.

Example 4.1. Consider the following interval linear program [2]

$$\min\ 5x_1 \ \text{ subject to }\ x_1 - x_2 = [-5, 5],\ x_1, x_2 \geq 0,$$

and an outcome function $f(x) = 3x_1 - 2x_2$. Figure 4.1 depicts the above program. In the figure, union of all the feasible solutions arising from all the realizations of the interval data is shown by the grey area, and the optimal set (Ω_b) is represented by the bold lines. The goal is to minimize (maximize) $f(x)$ over the bold lines. As can be observed from the figure, the outcome function range is $[-10, 15]$. Particularly, $\underline{f}$ is achieved on $x^* = (0, 5)$ which is the optimal solution of the linear program with $b = -5$, while $\overline{f}$ is obtained on $x^* = (5, 0)$ which is the optimal solution of the linear program with $b = 5$.

As already noted, the outcome range problem was initially motivated and studied in [94]. In this paper, we extend our previous results to address the outcome range problem for a more general type of ILP problems (i.e., Type I). We look deeper into properties of the problem and study how it is situated in a broader optimization context. Our solution approaches are more problem-specific here, and we test them

[2]This is a modification of Example 3 in [44].

on a wider range of problem instances.

4.3 Computational complexity

From Figure 4.1, we can see that even for a simple example with one real interval, the optimal set is nonconvex. In fact, a convenient description of the optimal set is not available in general [59]. This builds complexity of the outcome range problem. In this section, we address complexity of the ORP$_{\mathbf{b}}$. We need some definitions at this point. We define the optimal value range problem as the problem of finding the best and the worst optimal values among all the optimal values obtained over all data perturbations. The optimal value range of (4.2) reads as

$$\underline{z} := \inf\{z(b) : \quad b \in \mathbf{b}\}, \tag{4.5}$$

$$\overline{z} := \sup\{z(b) : \quad b \in \mathbf{b}\}, \tag{4.6}$$

where $\underline{z}$ and $\overline{z}$ can take on finite values, infinity or infeasiblity. The range $[\underline{z}, \overline{z}]$ then gives the optimal value range.

Theorem 4.1. *Problem of finding (4.3) and (4.4) is NP-hard.*

Proof. By Theorem 7 in [44] (p. 282), we know that computing the exact interval hull of the optimal set of ILP (4.2) is NP-hard. Now if we put $f(x) = x_i$, for any $i \in \{1, \dots, n\}$, we can conclude that ORP$_{\mathbf{b}}$ also is NP-hard. □

Now let us proceed with an observation related to $\overline{z}$.

Proposition 7. *Checking whether $\overline{z}$ is attained for a given $b \in \mathbf{b}$ is a co-NP-hard problem. This is true even for a class of problems with finite $\overline{z}$.*

Proof. By [45], checking strong solvability (i.e., solvability of each realization) of an interval system $Ax = \mathbf{b},\ x \geq 0$ is co-NP-hard, where A and $\mathbf{b}$ have the form

$$A = \begin{pmatrix} -M & M & 0 \\ \mathrm{e}^T & \mathrm{e}^T & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} [-e, e] \\ 1 \end{pmatrix},$$

where M is a non-negative positive definite matrix and e is a vector of all ones (both with proper dimensions). For the realization $b := (0^T, 1)^T$, the system is solvable, and one of its solution is $x := \frac{1}{n}(e^T, e^T, 1)^T$. Consider the following ILP

$$\min \;\; e^T y + e^T y' \;\; \text{subject to} \;\; Ax + y - y' = \mathbf{b},\ x, y, y' \geq 0.$$

The problem is strongly feasible and the objective function is bounded from below. Therefore, each realization has a finite optimal value and also $\overline{z}$ is finite, since it is attained for one of the realizations [110]. If b is the worst case scenario, then $\overline{z} = 0$ and the original system is strongly solvable. Otherwise, $\overline{z} > 0$, meaning that the system is not strongly solvable. □

Proposition 7 reveals the following results related to ORP$_{\mathbf{b}}$.

Corollary 4.1. *Checking whether $\overline{f}$ ($\underline{f}$) is obtained on a given $b \in \mathbf{b}$ is co-NP-hard.*

Proof. It follows from Proposition 7 by setting $r := \pm c$. □

Corollary 4.2. *Checking whether $\overline{f}$ ($\underline{f}$) is attained for a given $x \in \mathbb{R}^n$ is a co-NP-hard problem.*

Even though ORP$_{\mathbf{b}}$ is an NP-hard problem in general, there are few polynomially solvable cases of the problem. We present below one of these cases.

Proposition 8. *Suppose that in ILP (4.2), coefficient matrix A is a square matrix of full rank. Then, ORP$_{\mathbf{b}}$ is polynomillay solvable.*

Proof. Since we assume that A is a square matrix of full rank, LP(b) has at most one feasible solution for any given $b \in \mathbf{b}$. Thus, we can solve ORP$_\mathbf{b}$ by the means of two linear programs

$$\min(\max)\ r^T A^{-1} b \ \text{ subject to }\ A^{-1}b \geq 0,\ b \in \mathbf{b},$$

in variable b. □

Remark 4.1. For the sake of brevity, hereinafter, we only discuss results related to the computation of $\underline{f}$, that is, problem (4.3). All the results are also applicable to problem (4.4).

4.4 Properties of ORP$_\mathbf{b}$

Here, we investigate some theoretical properties of ORP$_\mathbf{b}$. Before going into the details, let us review some definitions.

Definition 4.1. A vector $x \in \mathbb{R}^n$ is called a weak feasible solution of ILP (4.2) if there exists $b \in \mathbf{b}$ such that $Ax = b,\ x \geq 0$.

Proposition 9. *(see [112] for a proof) The set of all the weak feasible solutions of ILP (4.2) is described as*

$$\mathcal{F} := \{x \in \mathbb{R}^n : \ \ Ax \leq \overline{b},\ -Ax \leq -\underline{b},\ x \geq 0\}.$$

Now let us recall ILP (4.2) for a given $b \in \mathbf{b}$,

$$\min\ c^T x \ \text{ subject to }\ Ax = b,\ x \geq 0. \tag{4.7}$$

By a basis B we mean an index set $B \subseteq \{1, \ldots, m\}$ such that A_B is nonsingular, where a subscript B on a matrix (row vector) denotes the submatrix (subvector) composed of columns indexed by B. That is, set B is the set of indices associated with basic variables. Similarly, an index set $N := \{1, \dots, m\} \setminus B$ indicates indices for nonbasic variables and as a subscript it represents restriction to nonbasic indices. A basis B is an optimal basis of (4.7) if and only if the following conditions hold

$$A_B^{-1} b \geq 0, \tag{4.8a}$$

$$c_N^T - c_B^T A_B^{-1} A_N \geq 0^T. \tag{4.8b}$$

Since vector c is fixed, for a given optimal basis, we call (4.8a) its basis stability region. An ILP problem may have several optimal bases, and consequently several basis stability regions.

Proposition 10. *The set of all scenarios $b \in \mathbf{b}$ for which there exists $x \geq 0$ such that $Ax = b$ is a bounded convex polyhedron.*

Proof. Let $A(\mathcal{F})$ denote the set of all $b \in \mathbf{b}$ for which there exists $x \geq 0$ such that $Ax = b$. Set $A(\mathcal{F})$ is naturally bounded by the box $\mathbf{b}$ and is the image of convex polyhedron $\mathcal{F}$ under a linear map, so it is a bounded convex polyhedron. $\square$

The above proposition implies that the uncertainty space of ILP (4.2) can be seen as a convex polyhedron, even though it is not always easy to describe it more explicitly. Below, we discuss another observation related to the above proposition.

Corollary 4.3. *The union of all the basis stability regions of ILP (4.2) is convex.*

We use this observation in one of our algorithms in Section 4.6. Also, Example 4.4 in Section 4.6.3 gives an illustration of the above corollary.

We now explore a property related to the scenario corresponding to $\underline{f}$.

Definition 4.2. b^* is an *optimal scenario* of (4.3) if $\underline{f} = f(x^*)$, where $x^* \in S(b^*)$.

Definition 4.3. Given an optimal scenario b^* of (4.3), an optimal basis B^* corresponding to $x^* \in S(b^*)$ such that $\underline{f} = f(x^*)$ is a *global optimal basis* of (4.3).

Given a global optimal basis B^*, it is natural to find $\underline{f}$ and b^* by solving the following linear program

$$\min\ r_{B^*}^T A_{B^*}^{-1} b \ \text{ subject to } \ A_{B^*}^{-1} b \geq 0,\ b \in \mathbf{b}, \tag{4.9}$$

in variable b.

Proposition 11. *Optimal scenario b^* is either a vertex of $\mathbf{b}$ or a realization $b^* \in \mathbf{b}$, for which LP(b^*) is degenerated.*

Proof. If no optimal scenario is a vertex of $\mathbf{b}$, then at least one optimal scenario lies on the boundary of at least two polyhedra characterized by the feasible solution set of (4.9). Therefore, the optimal solution of LP(b^*) is degenerated. $\square$

As stated earlier, in practical problems, we are interested in problems with the finite range of the outcome function, i.e., finite $\underline{f}$ and $\overline{f}$. Here we discuss a condition for the finiteness.

Proposition 12. *If $0 = \min r^T x$ subject to $Ax = 0,\ x \geq 0$, then $\underline{f} > -\infty$.*

Proof. By Proposition 9, we know that the set of all the weak feasible solutions is described by

$$Ax \leq \overline{b},\ -Ax \leq -\underline{b},\ x \geq 0.$$

Its recession (or characteristic) cone is described by $Ax = 0,\ x \geq 0$. So if the minimum of $r^T x$ over the recession cone is 0, then the minimum of $r^T x$ over $\mathcal{F}$ is bounded, and therefore also the minimum of $r^T x$ over Ω_b is bounded. $\square$

4.5 How does $\mathbf{ORP_b}$ relate to other problems?

$\mathrm{ORP}_\mathbf{b}$ lies somewhere between the optimal value range problem, multiobjective optimization, and bilevel optimization. In this section, we discuss the relation between $\mathrm{ORP}_\mathbf{b}$ and these problems. We also provide a mixed integer LP formulation of $\mathrm{ORP}_\mathbf{b}$.

4.5.1 Optimal value range problem

Let us recall the optimal value range problem from Section 4.3, that is, problems (4.5) and (4.6). There is a relation between $\mathrm{ORP}_\mathbf{b}$ and the optimal value range problem. Indeed, $\mathrm{ORP}_\mathbf{b}$ can be seen as a generalized form of *a special case* of the optimal value range problem. We formalize the relation between the two problems by means of the following observation.

Observation 4.2. *If $\underline{z}$ and $\overline{z}$ are finite, we then can reformulate the optimal value range problem as*

$$\underline{z} = \min\ c^T x \ \ \textit{subject to}\ \ x \in \Omega_b, \tag{4.10}$$

$$\overline{z} = \max\ c^T x \ \ \textit{subject to}\ \ x \in \Omega_b. \tag{4.11}$$

We see that, under the assumption of Observation 4.2, $\underline{f}$ $(\overline{f})$ would be a generalization of $\underline{z}$ $(\overline{z})$. Indeed, in such a case, if $f(x) = c^T x$ then looking for $[\underline{f}, \overline{f}]$ is equivalent to looking for $[\underline{z}.\overline{z}]$. Note that such a relationship between $\underline{f}$ $(\overline{f})$ and $\underline{z}$ $(\overline{z})$ does not hold in general, as shown in the following example.

Example 4.2. Consider the following ILP problem

$$\min\ 3x_1 + 5x_2 \ \text{ subject to }\ x_1 + x_2 = [-1, 5],\ x_1, x_2 \geq 0,$$

and let $f(x) = 3x_1 + 5x_2$ be also the outcome function. Figure 4.2 depicts the above

ILP problem. From the figure, we see that $\underline{z} = 0$. It is also easy to see that $LP(b = -1)$ is infeasible and, by convention [111], we set $z(b = -1) = \infty$. Thus, the range $[0, \infty)$ gives the optimal value range. However, from Figure 4.2, it is not hard to observe that $f(x)$ ranges in $[0, 15]$, which is different from the optimal value range.

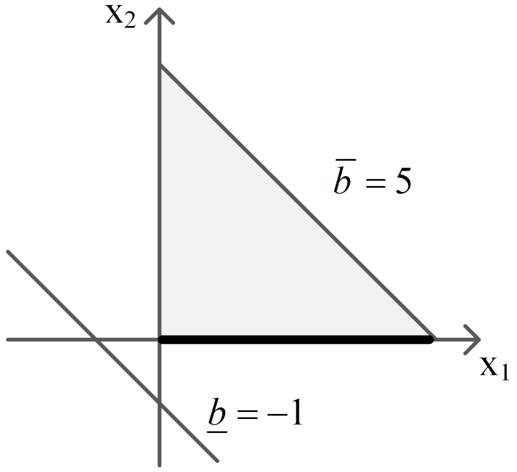


Figure 4.2: (Example 4.2) The set of all the weak feasible solutions and the optimal set are in grey and bold, respectively.

Moreover, apart from the relation in the theory, the two problems are essentially different from the application stand point. In particular, the optimal value range problem returns the best and the worst values of an objective function taking into account all the realizations of the uncertain parameters. This problem provides a decision maker a sense of the risk involved in his/her decision from the operational perspective. However, the outcome range problem consists of computing the best and the worst values of a given (additional) linear function over the set of all optimal solutions of a linear program with interval data. The problem particularly aims at quantifying unintended consequences of optimal decisions in an uncertain environment; hence, providing additional important information on a system to decision makers who are observing it from outside (e.g., policy makers).

4.5.2 Multiobjective optimization

A multiobjective linear programming problem is defined as

$$\min\ Cx \ \text{ subject to } \ \mathcal{M}(A,b), \tag{4.12}$$

where $C \in \mathbb{R}^{v\times n}$ and $\mathcal{M}(A,b)$ is the feasible set defined by linear constraints with matrix coefficient $A \in \mathbb{R}^{m\times n}$ and right-hand side vector $b \in \mathbb{R}^m$. A solution $x^e \in \mathcal{M}(A,b)$ is said to be an *efficient* solution of (4.12) if there exists no $x \in \mathcal{M}(A,b)$ such that $Cx \leq Cx^e$ with at least one strict inequality.

One may see a relation between $\mathrm{ORP}_{\mathbf{b}}$ and multiobjective linear programs with interval right-hand sides. Let us reformulate (4.3) as the following interval multiobjective linear program

$$\begin{aligned} \min\ \ & f_1(x) = r^T x \\ \min\ \ & f_2(x) = c^T x \\ & \text{subject to } \ Ax = \mathbf{b},\ x \geq 0. \end{aligned} \tag{4.13}$$

Definition 4.4. A solution x^e is called a *possibly efficient* [15, 72] solution of (4.13) if it is feasible and efficient for some $b \in \mathbf{b}$.

Problems (4.3) and (4.13) are not equivalent because the latter always deals with a larger set of solutions. We show this by the following example.

Example 4.3. Consider the following linear program with an interval right-hand side

$$\min\ 2x_1 \ \text{ subject to } \ x_1 + x_2 = [0,5],\ x_1, x_2 \geq 0, \tag{4.14}$$

and an outcome function $f(x) = -2x_1 + 5x_2$. Now let us recall the multiobjective

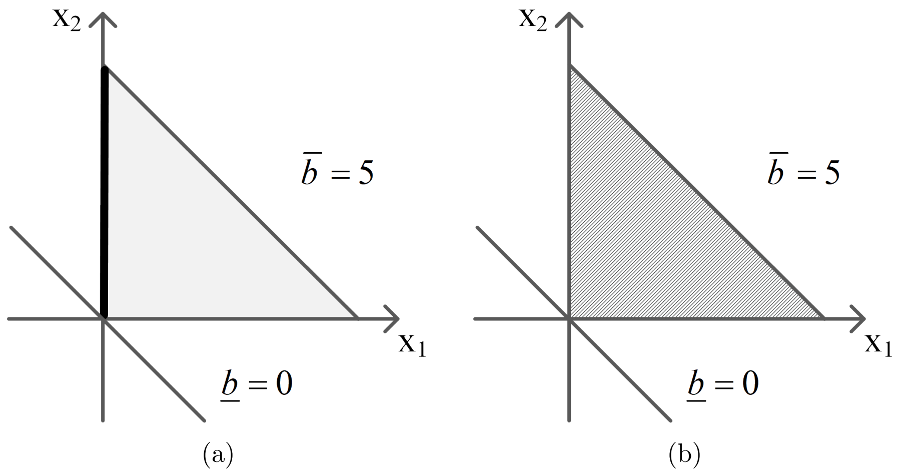


Figure 4.3: (Example 4.3) **(a)** the optimal set of (4.14) is in bold; **(b)** the set of all the possibly efficient solutions of (4.15) is shown by a pattern.

reformulation of (4.3):

$$\min\ (-2x_1 + 5x_2, 2x_1) \ \text{ subject to } \ x_1 + x_2 = [0, 5],\ x_1, x_2 \geq 0. \tag{4.15}$$

Figure 4.3a shows the optimal set of (4.14), while Figure 4.3b depicts the set of all the possibly efficient solutions of (4.15) obtained from all the realizations of the uncertain data. As can be seen, the two sets represented in the figures are not equivalent. From Figure 4.3a, we can observe that $\underline{f} = 0$. By considering the set of all the possibly efficient solutions in Figure 4.3b, the minimum value of the first objective is -10, which is different from what we observe from Figure 4.3a. Thus, we can conclude that (4.3) and (4.13) are not equivalent problems in general. However, the two problems are not totally irrelevant. The following reveals a relation between the two problems.

Proposition 13. *Let $x^* \in \Omega_b$ be given such that $\underline{f} = f(x^*)$. Then x^* is a possibly efficient solution of (4.13).*

Proof. Let b^* be an optimal scenario of (4.3). We can see that in (4.13), the following holds

$$f_2(x^*) \le f_2(\hat{x}), \ \ \forall \hat{x} \in \{x : \ Ax = b^*, \ x \ge 0\}.$$

We also know that

$$\underline{f} = f_1(x^*) \le f_1(\tilde{x}), \ \forall \tilde{x} \in \Omega_b.$$

Therefore, there exists no $\hat{x} \in \{x : \ Ax = b^*, \ x \ge 0\}$ such that either $f_2(\hat{x}) < f_2(x^*)$, or $f_2(\hat{x}) = f_2(x^*)$ and $f_1(\hat{x}) < f_1(x^*)$. Thus, x^* is a possibly efficient solution of (4.13). □

4.5.3 Bilevel optimization

Let us reformulate (4.3) as the following bilevel linear program

$$\begin{aligned}
&\text{“min”} \ \ 0^T b + r^T x \\
&\text{subject to} \\
&b \in \mathbf{b}, \ x \in \arg\min_x \{c^T x \ \text{ subject to } \ Ax = b, \ x \ge 0\}.
\end{aligned} \tag{4.16}$$

Here, b denotes the leader decision vector and x denotes the follower decision vector. In the above formulation we use quotation in the leader problem due to a vagueness that arises when the follower problem admits multiple optimal solutions for any given $b \in \mathbf{b}$. In other words, in case of multiple follower optimal solutions for a given $b \in \mathbf{b}$, it is not clear how the follower behaves.

Let us rewrite the constraint defined by the follower problem as a set-valued mapping

$$\Psi(b) = \arg\min_x \{c^T x \ \text{ subject to } \ Ax = b, \ x \ge 0\},$$

where $\Psi : \mathbb{R}^m \to \mathbb{R}^n$. An equivalent formulation of (4.16) can be seen as

$$\text{“min”} \ \ 0^T b + r^T x \ \text{ subject to } \ b \in \mathbf{b}, \ x \in \Psi(b).$$

Now suppose that we expect the follower to be collaborative and to behave in favor of the leader, choosing the most favorable solution for the leader (i.e., optimistic position). The reaction set of the follower is then given by

$$\Psi^o(b) = \arg\min_x \{0^T b + r^T x : \quad x \in \Psi(b)\}.$$

Bilevel linear program (4.16) under the optimistic position of the follower can be formulated as

$$\min \; 0^T b + r^T x \; \text{ subject to } \; b \in \mathbf{b}, \; x = \Psi^o(b). \tag{4.17}$$

By considering Assumption 4.1 in Section 4.2, it is easy to see that (4.3) and (4.17) are equivalent problems. Thus, ORP$_{\mathbf{b}}$ can be equivalently formulated as two optimistic bilevel linear programs. However, bilevel linear programming is too general to solve ORP$_{\mathbf{b}}$ efficiently as it is known to be strongly NP-hard [55]. We also confirm this by our numerical experiments[3].

4.5.4 Mixed integer LP formulation

Let us here remind you the program (4.7), which corresponds to ILP (4.2) for a given $b \in \mathbf{b}$. The dual of (4.7) reads

$$\max \; b^T y \; \text{ subject to } \; A^T y \le c, \tag{4.18}$$

where $y \in \mathbb{R}^m$ is the vector of dual variables. By using the complementary slackness condition, (x, y) is a pair of primal (4.7) and dual (4.18) optimal solutions if and only if they solve the system

$$Ax = b, \; x \ge 0, \; A^T y \le c, \; x^T(c - A^T y) = 0.$$

[3]Interested readers are referred to Ref. [122] for a review on bilevel optimization.

Now we assume, without loss of generality, that b is a vector of variables with given lower and upper bounds (i.e., $b \in \mathbf{b}$). We can then describe the optimal set of ILP (4.2) by the following parametric system

$$Ax = b,\ x \geq 0,\ A^T y \leq c,\ x^T(c - A^T y) = 0,\ b \in \mathbf{b}. \tag{4.19}$$

This is a nonlinear system. Note that $c - A^T y \geq 0$. Thus, let us define an auxiliary variable $h \geq 0$ where $h = c - A^T y$. Then the complementary slackness constraint reads $x^T h = 0$. Since x and h are non-negative, we can restate the constraint as

$$x_i = 0 \vee h_i = 0,\ \forall i \in \{1, \ldots, n\}.$$

Thus, the complementary slackness constraint can be seen naturally as a logical constraint. Let us define a binary variable $\phi \in \{0,1\}^n$. We can rewrite (4.19) as

$$Ax = b,\ x \geq 0,\ c - A^T y = h,\ x \leq \phi L,\ h \leq (1 - \phi)L,\ h \geq 0,\ b \in \mathbf{b},\ \phi \in \{0,1\}^n, \tag{4.20}$$

where L is a sufficiently large constant. We now can solve ORP$_\mathbf{b}$ to optimality by solving two mixed integer linear programs, that is, minimizing (maximizing) $f(x)$ subject to system (4.20).

4.6 Approximating ORP$_\mathbf{b}$

Since the structure of the optimal set (Ω_b) can be very complicated, we often are not able to solve ORP$_\mathbf{b}$ to optimality. Thus, in this section, we present three methods to approximate ORP$_\mathbf{b}$. Particularly, we develop a method based on the reformulation-linearization technique to get an outer approximation of the outcome function range. We also design two iterative improvement algorithms to find an inner approximation of the outcome function range. We now discuss our methods in detail.

4.6.1 Outer approximation: A reformulation-linearization technique

The idea behind our method is that we first describe Ω_b by a nonlinear system, and we then linearize it by utilizing the reformulation-linearization technique (RLT) introduced in [119].

If, instead of the complimentary slackness condition in Section 4.5.4, we consider the strong duality condition in linear programming, we then can describe Ω_b as the set of solutions to the following system

$$Ax = b,\ A^T y \leq c,\ c^T x = b^T y,\ x \geq 0,\ b \in \mathbf{b}, \tag{4.21}$$

in variables x, y, and b. Constraint $c^T x = b^T y$ is the strong duality constraint. This system includes bilinear terms (i.e., $b^T y$), and thus it is very hard to deal with in general. We apply the RLT to linearize the above system. The RLT consists in two steps: (i) the first step generates valid quadratic constraints by suitably multiplying some constraints of the system by a combination of nonegative variable factors, constructed using the problem constraints, and (ii) the second step linearizes the nonlinear terms by defining new variables. The resulting linear system is known to enclose the nonlinear system.

Remark 4.2. System (4.19) has n bilinear terms in n constraints, while system (4.21) has m bilinear terms in one constraint. This makes applying RLT to the former system more cumbersome than the latter system. Hence, we here chose to work with system (4.21).

Reformulation phase. Let $\underline{y}$ and $\overline{y}$ be the lower and the upper bounds on the dual decision vector y, respectively. We define the following bound factors for y

$$\overline{y}_i - y_i \geq 0 \ \ \forall i \in \{1, \dots, m\}, \tag{4.22a}$$

$$y_i - \underline{y}_i \geq 0 \ \ \forall i \in \{1, \dots, m\}. \tag{4.22b}$$

We similarly define the bound factors for b as

$$b_i - \underline{b}_i \geq 0 \ \ \forall i \in \{1, \dots, m\}, \tag{4.23a}$$

$$\overline{b}_i - b_i \geq 0 \ \ \forall i \in \{1, \dots, m\}. \tag{4.23b}$$

The goal is to construct the nonlinear terms $b_i y_i$ in the constraints of (4.21) by using the above bound factors. This can be done in many ways one of which is pairwise multiplying constraints in (4.22) and (4.23). For the sake of illustration, suppose the multiplication of (4.22a) and (4.23a)

$$(\overline{y}_i - y_i)(b_i - \underline{b}_i) \geq 0 \ \ \forall i \in \{1, \dots, m\}.$$

This can be then rearranged into

$$-y_i b_i + \overline{y}_i b_i + \underline{b}_i y_i \geq \overline{y}_i \underline{b}_i \ \ \forall i \in \{1, \dots, m\}.$$

System (4.21) together with all the bilinear constraints, generated as explained above, yields an equivalent reformulation of Ω_b, that is,

$$\begin{aligned}
[\mathrm{R}_1] \qquad & Ax = b,\ A^T y \le c,\ c^T x = b^T y,\ x \ge 0,\ b \in \mathbf{b}, \\
& -y_i b_i + b_i \underline{y}_i + \overline{b}_i y_i \ge \overline{b}_i \underline{y}_i \ \ \forall i \in \{1, \ldots, m\}, \\
& -y_i b_i + b_i \overline{y}_i + \underline{b}_i y_i \ge \underline{b}_i \overline{y}_i \ \ \forall i \in \{1, \ldots, m\}, \\
& y_i b_i - b_i \overline{y}_i - \overline{b}_i y_i \ge -\overline{b}_i \overline{y}_i \ \ \forall i \in \{1, \ldots, m\}, \\
& y_i b_i - b_i \underline{y}_i - \underline{b}_i y_i \ge -\underline{b}_i \underline{y}_i \ \ \forall i \in \{1, \ldots, m\}.
\end{aligned}$$

Note that R_1 is one reformulation of (4.21). We can have a variety of reformulations by multiplying more constraints by the bound factors. In general, the more constraints of this type we construct, the better the resulting approximation would be. That being said, the size of the reformulation gets large dramatically as we multiply more constraints. Therefore, from a computational efficiency standpoint, a proper trade-off is needed.

Here, we also offer another reformulation of (4.21) where, in addition to the valid constraints obtained by pairwise multiplying the bound factors, we construct some additional quadratic constraints by pairwise product of $A^T y \le c$ and (4.23). As an example,

$$(c - A^T y)(b_i - \underline{b}_i) \ge 0 \ \forall i \in \{1, \ldots, m\}.$$

As can be noted, we create new bilinear terms since we have $b_i y$ for each $i \in \{1, \ldots, m\}$. These additional terms also need to be taken care of when we multiply

(4.22) and (4.23). After a simplification, an alternative reformulation of Ω_b reads

$$
\begin{aligned}
[\mathrm{R}_2] \qquad & Ax = b,\ A^T y \leq c,\ c^T x = b^T y,\ x \geq 0,\ b \in \mathbf{b}, \\
& -b_i A^T y + c b_i + \underline{b}_i A^T y \geq c \underline{b}_i \ \ \forall i \in \{1, \ldots, m\}, \\
& b_i A^T y - c b_i - \overline{b}_i A^T y \geq -c \overline{b}_i \ \ \forall i \in \{1, \ldots, m\}, \\
& -y_j b_i + b_i \underline{y}_j + \overline{b}_i y_j \geq \overline{b}_i \underline{y}_j \ \ \forall i, j \in \{1, \ldots, m\}, \\
& -y_j b_i + b_i \overline{y}_j + \underline{b}_i y_j \geq \underline{b}_i \overline{y}_j \ \ \forall i, j \in \{1, \ldots, m\}, \\
& y_j b_i - b_i \overline{y}_j - \overline{b}_i y_j \geq -\overline{b}_i \overline{y}_j \ \ \forall i, j \in \{1, \ldots, m\}, \\
& y_j b_i - b_i \underline{y}_j - \underline{b}_i y_j \geq -\underline{b}_i \underline{y}_j \ \ \forall i, j \in \{1, \ldots, m\}.
\end{aligned}
$$

Linearization phase. We linearize nonlinear constraints in R$_1$ and R$_2$ through an appropriate variable substitution. Particularly, to linearize system R$_1$, we substitute

$$w_i = b_i y_i \ \ \forall i \in \{1, \ldots, m\},$$

and for linearizing system R$_2$, we put

$$w_{ij} = b_i y_j \ \ \forall i \in \{1, \ldots, m\},\ j \in \{1, \ldots, m\}.$$

Let $E(R_1)$ and $E(R_2)$ be respectively the linearized systems R$_1$ and R$_2$ after such a substitution. In our computational experiments discussed in Section 4.7, we assess the performance of the two following outer approximations (i.e., lower bounds)

$$[\mathrm{RLT1}] \qquad \underline{f}^{L_1} = \min r^T x \ \text{ subject to } \ (x, y, b, w) \in E(R_1), \tag{4.24}$$

$$[\mathrm{RLT2}] \qquad \underline{f}^{L_2} = \min r^T x \ \text{ subject to } \ (x, y, b, w) \in E(R_2). \tag{4.25}$$

4.6.2 Inner approximation: A gradient-restoration based algorithm

We here design an algorithm to explore Ω_b. The idea builds from the gradient-restoration algorithm which was originally developed to minimize constrained non-smooth functions [91]. Our algorithm basically alternates between an improving phase and a restoration phase and can be applied to find an upper bound for $\underline{f}$. Figure 4.4 is an illustration on how the algorithm works. From a high level perspective, we start from a feasible point of (4.3), that is, a given $x_k \in \Omega_b$ (Figure 4.4a). We then obtain a new point u in a neighborhood of x_k, namely $N(x_k)$, such that $f(u) \leq f(x_k)$ (Figure 4.4b). If the new point u belongs to Ω_b, then we store the value of $f(u)$ and start over with u as the new feasible point. Otherwise, we apply a restoration phase where we perturb a given $u \notin \Omega_b$ to obtain a new point $\hat{x}$ such that $\hat{x} \in \Omega_b$ (Figure 4.4d). The restoration phase, indeed, ensures that we have an incumbent solution at each iteration. After this phase, a new iteration starts with $\hat{x}$ as the new feasible point. The algorithm iterates until the improvement is not significant. A general pseudo-code of the algorithm is given in Algorithm 4.1. We now discuss each step in detail.

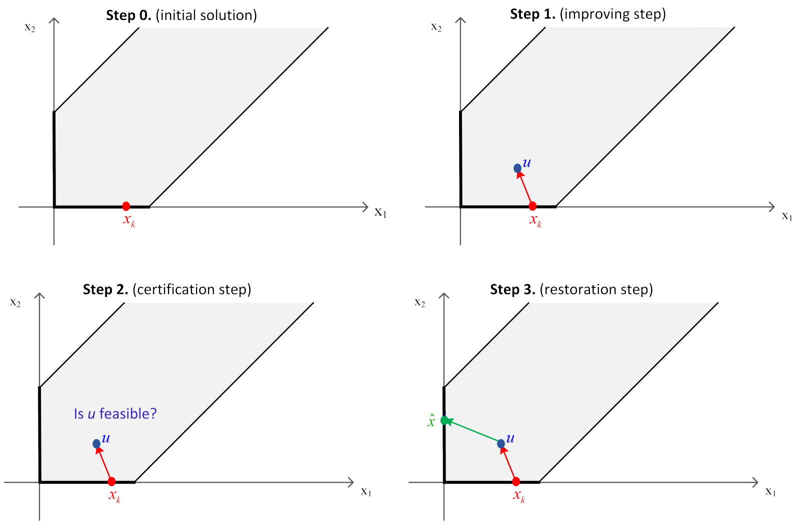

Step 0. (initial solution)
Step 1. (improving step)
Step 2. (certification step)
Step 3. (restoration step)
Is u feasible?
x2
x1
xk
u
x̂

Figure 4.4: An illustration of our proposed Algorithm 4.1. We borrowed the figure from Example 4.1 (Ω_b is in bold).

Algorithm 4.1: A gradient-restoration based algorithm

```
1  **Input:** input data
   **Result:** an upper bound of $\underline{f}$
2  Set $k \leftarrow 0$.
3  **Step 0.** (*initial point*): Find an initial $x_k \in \Omega_b$ and set $\hat{\underline{f}}_k \leftarrow f(x_k)$.
4  **while** we can move to a new point in the weak feasible solution set **do**
5      **Step 1.** (*improving step*)
6      Select $u \in N(x_k)$ such that $f(u) \leq f(x_k)$.
7      **Step 2.** (*certification step*)
8      Set $k \leftarrow k+1$.
9      **if** $u \in \Omega_b$ **then**
10         Put $x_k \leftarrow u$ and $\hat{\underline{f}}_k \leftarrow f(u)$.
11     **else**
12         **Step 3.** *(restoration step)*
13         Retrieve a $\hat{x} \in \Omega$.
14         Set $x_k \leftarrow \hat{x}$ and $\hat{\underline{f}}_k \leftarrow f(\hat{x})$.
15     **end**
16 **end**
```

Step 0. (initial point) We can get an initial point x_0 simply by solving LP(b) to optimality for a random scenario $b \in \mathbf{b}$.

Step 1.(improving step) In the improving step, it is natural to perturb x_k in the direction of $d := -r$. In particular, we perturb x_k as

$$u = x_k + \xi \alpha^{max} d,$$

where $\xi \in (0, 1]$, and α^{max} is the maximum perturbation in direction of d such that weak feasibility of u is preserved. By Proposition 9, the following univariate linear program yields α^{max}

$$\max\ \alpha \ \text{ subject to } \ \underline{b} \leq A(x_k + \alpha d) \leq \overline{b},\ x_k + \alpha d \geq 0,\ \alpha \geq 0.$$

Step 2.(certification step) In this step, the goal is to check whether u belongs to Ω_b. That is, we would like to examine whether u is an optimal solution of ILP (4.2) for some $b \in \mathbf{b}$.

Theorem 4.2. *(see [83]) Let u be a weak feasible solution of ILP (4.2), and assume that $u_{q_1} > 0, \dots, u_{q_p} > 0,\ \ u_{q_{p+1}} = 0, \dots, u_{q_n} = 0$. Then u is an optimal solution of ILP (4.2) for some $b \in \mathbf{b}$ if and only if the following system is solvable*

$$A^T_{q_j,.} t = c_{q_j},\ \ j = 1, \dots, p, \tag{4.26a}$$

$$A^T_{q_j,.} t \le c_{q_j},\ \ j = p+1, \dots, n, \tag{4.26b}$$

where $A_{j,.}$ denotes the j-th row of matrix A, and $t \in \mathbb{R}^m$ is a decision vector.

Note that to check solvability of system (4.26), we can either directly use the well-known Farkas' lemma, or check whether the following linear program with a dummy objective function has a feasible solution

$$\min\ 0^T t \ \text{ subject to } \ (4.26).$$

Step 3. (restoration step) Our aim, in this step, is to retrieve a point from Ω_b. To this end, we first obtain $b := Au$, and we then find an optimal basis of LP(b), say B. The set of all optimal solutions with the optimal basis B is equal to the set of all the weak feasible solutions of the following interval linear system

$$A_B x_B = \mathbf{b},\ x_B \ge 0,\ x_N = 0,$$

which by Proposition 9 can be described by a polyhedral set

$$A_B x_B \le \overline{b},\ -A_B x_B \le -\underline{b},\ x_B \ge 0. \tag{4.27}$$

Therefore, we can retrieve a $\hat{x} \in \Omega_b$ by solving the following linear program

$$\min\ r_B^T x_B \ \text{ subject to } \ (4.27).$$

Remark 4.3. If $\alpha^{max} = 0$, we are on the border of the weak feasible solution set of ILP (4.2). Hence, to be able to move, we relax the weak feasibility condition and perturb x_k as

$$u' = x_k + \xi d.$$

The resulting u' is not a weak feasible solution of (4.2), i.e., $u' \notin \mathcal{F}$. To gain weak feasibility, we project u' onto the set $\mathcal{F}$ by solving the following least square problem

$$\min_{u \in \mathcal{F}} \|u - u'\|_2,$$

in variable u. The above projection indeed returns the vector u that we need for Step 1; however, it might not necessary preserve the condition of $f(u) \leq f(x_k)$. For the sake of avoiding getting trapped at border points, in this case, we relax that condition.

4.6.3 Inner approximation: A bases inspection approach

In this section, we design an algorithm in the scenario space. Given an optimal basis B, we can compute the lowest achievable value of f at the basis stability region of B by solving a linear program, that is,

$$\min\ r_B^T A_B^{-1} b \ \text{ subject to } \ A_B^{-1} b \geq 0,\ b \in \mathbf{b}, \tag{4.28}$$

in variable b. Basically, this implies that the box $\mathbf{b}$ can be decomposed into subparts according to its basis stability regions, and thus the outcome function can be evaluated for each subpart.

Example 4.4. Consider the following ILP problem

$$\min\ (-7, 9, -10, -5)x \ \text{ subject to } \begin{pmatrix} -9 & 2 & 12 & 12 \\ -10 & 2 & 15 & 19 \end{pmatrix} x = \begin{pmatrix} [1, 6] \\ [4, 7] \end{pmatrix},\ x \geq 0,$$

and an outcome function

$$f(x) = (11, -5, 4, -10)x.$$

The above problem possesses three optimal bases, $B_1 = \{1, 2\}$, $B_2 = \{2, 3\}$, and $B_3 = \{1, 3\}$. For each optimal basis, the constraint $A_B^{-1}b \geq 0$ from (4.28) reads

$$\begin{aligned} B_1 : &\quad b_1 - b_2 \geq 0,\ \ b_1 - 0.9b_2 \geq 0, \\ B_2 : &\quad b_1 - 0.8b_2 \geq 0,\ \ -b_1 + b_2 \geq 0, \\ B_3 : &\quad -b_1 + 0.8b_2 \geq 0,\ -0.67b_1 + 0.6b_2 \geq 0. \end{aligned}$$

Solving (4.28), the lowest achievable values of $f(x)$ at B_1, B_2, and B_3 are -38, -15, and 1.07, respectively. Hence, we have $\underline{f} = -38$. Figure 4.5 depicts the interval vector $\mathbf{b}$ and its subparts corresponding to the basis stability regions of the three optimal bases.

One natural way to approximate $\underline{f}$ is to compute the lowest achievable value of f for as many optimal bases as we can, and then to take the tightest approximation among all. Although the number of bases are exponentially large in general, an ILP problem might possess a limited number of optimal bases.

Here, we present a guided optimal bases search. We start with a random scenario $b_\theta \in \mathbf{b}$. We find an optimal basis of LP(b_θ), namely, B. LP (4.28) returns the lowest achievable f at the optimal Basis B. Let $\tilde{b}_\theta$ be an optimal solution of (4.28). Now to go to another optimal basis, we slightly change $\tilde{b}_\theta$ in the direction of the derivative of

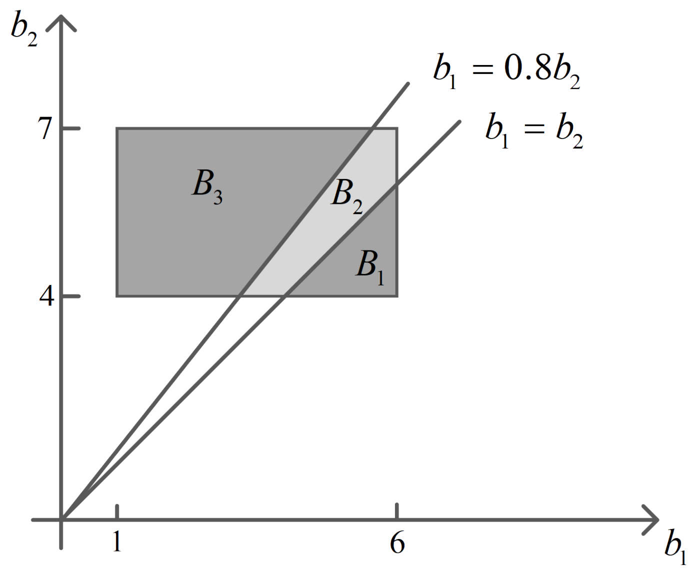


Figure 4.5: (Example 4.4) The decomposition of interval vector $\mathbf{b}$ according to the basis stability regions of the optimal bases B_1, B_2, and B_3.

outcome function, i.e., $\ell_B := -r_B^T A_B^{-1}$. That is,

$$b' = \tilde{b}_\theta + \lambda \ell_B, \tag{4.29}$$

where λ is a small positive constant. Hence, the outcome function locally acts as a linear function of $-r_B^T A_B^{-1}$. We start the next iteration of the algorithm with b' as the new b_θ. We repeat the above procedure until we are not able to move to a new optimal basis. Algorithm 4.2 summarizes the steps of our proposed method.

Remark 4.4. In the case that we get stuck in one basis stability region and cannot move to another one, we choose another extreme point (different from the optimal one) of the current basis stability region and proceed along a different direction. Specifically, this can be done by selecting an arbitrary objective vector for (4.28). After solving (4.28) with the arbitrary objective vector, we compute b' according to the new ℓ_B.

Remark 4.5. For the case $b' \notin \mathbf{b}$, we project it onto the set $\mathbf{b}$ by solving

$$\min_{b \in \mathbf{b}} \|b - b'\|_2,$$

in variable b.

Algorithm 4.2: A bases inspection algorithm

```
1  Input: input data
   Result: an upper bound of $\underline{f}$
2  Set $\theta \leftarrow 0$.
3  Generate a random scenario $b_\theta \in \mathbf{b}$.
4  while it is possible to move to a new optimal basis do
5      Find an optimal basis of LP($b_\theta$)
6      Compute an approximation of $\underline{f}$ by (4.28).
7      Determine $\tilde{b}_\theta$ and $\ell_B$.
8      Find $b'$ using (4.29).
9      if $b' \notin \mathbf{b}$ then
10         Project $b'$ onto $\mathbf{b}$.
11     end
12     Put $\theta \leftarrow \theta + 1$.
13     Set $b_\theta \leftarrow b'$.
14 end
```

4.7 Computational experiments

In this section, we empirically evaluate the performance of our outer and inner approximation methods. Since there exists no standard algorithm in the literature to address the outcome range problem, we assess the quality of our methods against some standard solvers. In particular, we use FMINCON solver in MATLAB to solve the nonlinear formulation of $\text{ORP}_\mathbf{b}$, that is, optimizing $f(x)$ subject to nonlinear system (4.21). FMINCON returns an inner approximation of the range of an outcome function. Additionally, to solve $\text{ORP}_\mathbf{b}$ to optimality, we use CPLEX to solve the MILP reformulation (see (4.20)) and YALMIP [87] to solve the bilevel reformulation (see (4.17)). It is not surprising that these general purpose solvers can only handle the problem efficiently up to a certain point. We show this in the following sections.

4.7.1 Test instances

We evaluate our methods on three test beds. Our first test bed contains the set of randomly generated problem instances. The problem instances in this case are general with no specific structure. For the second test bed, we generated problem instances with a special structure where the coefficient matrix is an arc-node incidence matrix. Finally, for the third test bed, we considered four problems from the MIPLIB 2010 repository [77]. These problems were also used in [81] as benchmark problems.

Test bed 1. Given the size parameters (m, n), we randomly generated entries of $A \in \mathbb{Z}^{m\times n}$ in $[-10, 10]$. The boundedness of the feasible solution set was ensured by generating the last row of A in $[0, 10]$. We also randomly generated a solution vector $x_0 \in \mathbb{Z}^n$ in $[0, 10]$. We then constructed interval vector $\mathbf{b}$ as $\underline{b} = Ax_0$ and $\overline{b} = \underline{b} + \delta e$, where δ is the interval width and e is an all-ones vector with the convenient dimension. This procedure guaranties weak feasibility of the problem instances. We similarly randomly generated vectors $c, r \in \mathbb{Z}^n$ in $[-10, 10]$. We generated 30 problem instances for each triplet (m, n, δ), for a total of 2,100 problem instances.

Test bed 2. We here used a similar procedure to the first test bed except that matrix A was constructed such that it is an arc-node incidence matrix. Specifically, given (ω, ρ), we considered the coefficient matrix of a balanced transportation problem with ω origins and ρ destinations. Since the rank of matrix A in transportation problems is $\omega + \rho - 1$, to proceed further, we eliminated one row from matrix A for each problem instance. In other words, the problem instances in this case are basically transportation problems missing one constraint. The interval vector $\mathbf{b}$ was then composed as in test bed 1. Also, we randomly generated the all-integer vector c in $[1, 30]$ and the all-integer vector r in $[-10, 10]$ (both with convenient dimensions). Similar to the first test bed, we generated 30 problem instances for each combination of ω, ρ and δ, for a

Table 4.1: Details of the problems chosen from the MIPLIB 2010 repository.

problem	number of constraints	number of variables	non-zeros in A	non-zeros in c
enlight13	169	338	962	169
enlight15	225	450	1,290	225
mik-250-1-100-1	401	652	6,002	251
roll3000	3,460	4,452	33,837	1

total of 360 problem instances.

Test bed 3. We used four problems from the MIPLIB 2010 repository, namely, enlight13, enlight15, mik-250-1-100-1, and roll3000. These problems are mixed integer programs containing a combination of inequality and equality constraints. We relaxed integrality constraints and transformed all the problems to the standard form LP with equality constraints (i.e., Type I). Table 4.1 reports size of the problems and number of non-zero elements in input parameters. As noted above, these problem instances were used in [81] as a test bed. Lee et al. [81] assume that A, c are fixed but a set of right-hand side parameter values are given. They generated the set of right-hand side vectors from different multivariate distributions with different sample sizes. In our test bed, we took a similar approach and generated the right-hand side vectors using a multivariate normal distribution with different sample sizes.[4] We then computed the interval hull of the generated vectors to construct the interval vector $\mathbf{b}$. We refer readers to Appendices in [34, 81] for details of sampling procedures for each problem. We also randomly generated the all-integer vector r in $[-10, 10]$ for enlight13, enlight15 and roll3000 and in $[0, 500]$ for mik-250-1-100-1 (all with proper dimensions). In total, for this test bed, we deal with 32 problem instances.

[4]Sample sizes here are similar to those considered in [81].

4.7.2 Implementation details

For the first two test beds, we tried the gradient-restoration based algorithm (GR) and the bases inspection approach (BI) with two randomly generated initial points and chose the best solution among the two resulting outputs. However, for test bed 3, due to the large size of the problem instances, we ran the two algorithms with only one initial point, and we also imposed a limit of 50 on the number of basis stability regions that BI can check. We set parameter ξ in BI to 0.2 and parameter λ in GR to 0.1. In our MILP reformulation, for test bed 1, we considered a conservative constant of $L = 1,000$ for each problem instance, while we set L, for test bed 2, to the maximum value of vector $\overline{b}$ in each problem instance. For our reformulation-linearization technique, we applied the contractor algorithm in [59] to get bounds on y, i.e., $[\underline{y}, \overline{y}]$.

There are five internal stopping conditions in FMINCON, namely, maximum iterations, maximum function evaluations, step tolerance, function tolerance, and constraint tolerance. We set the first two conditions to $1,000,000$, the step tolerance to $(1.000E-10)$, and function and constraint tolerances to $(1.000E-6)$. Moreover, we fed the solver with an initial solution which is an optimal solution of LP(b) for a random $b \in \mathbf{b}$. We additionally considered a time limit of 30 minutes for solving a problem instance. For each problem instance, in the case that the solver reached the time limit without converging to a solution before meeting its internal stopping conditions, we considered the returned solution a valid one if it was within the feasibility tolerance. For the cases the solver met either of its internal stopping conditions before reaching the time limit, we saved the solution if it was a feasible one, and we started over using another initial solution. For these particular cases, we continued in this way until either the solver normally converged to a solution or the time limit was reached. Finally, we returned the best solution among all the saved solutions (if there was any).

We implemented our algorithms using MATLAB (R2019b). We used CPLEX 12.9 to solve linear programs, mixed integer programs and least square problems. We also used YALMIP solver to solve the bilevel reformulation. We carried out our experiments on a computer with an Intel(R) Core (TM) i7-4790 CPU processor at 3.60 GHZ with 32.00 GB of RAM.

4.7.3 Numerical results

We here discuss results of our computational experiments. Let us recall that we compare results obtained by the reformulation-linearization technique (RLT1 & RLT2), bases inspection approach (BI), gradient-restoration based algorithm (GR), FMINCON solver used for solving the nonlinear formulation (FMIN), CPLEX solver employed for solving the MILP formulation, and YALMIP solver applied for solving the bilevel formulation. It is also worth reminding that we only report the results related to the computation of $\underline{f}$.

We divided the results related to the first test bed into two parts. The first part corresponds to the results obtained over a set of smaller size problem instances for which we were able to solve $\underline{f}$ to optimality efficiently. The second part reports the results computed over a set of larger size problem instances for which we could not obtain the exact value of $\underline{f}$ in a reasonable time. Given an estimation $\hat{\underline{f}}$, we computed its gap from the optimal value by

$$\text{gap} := \left| \frac{\hat{\underline{f}} - \underline{f}}{\underline{f}} \right|.$$

Hence, the lower the gap, the better the performance. To reduce the effects of possible outliers on our results, for the cases that the average gap was greater than 1, we reported the average gap after eliminating the maximum gap (i.e., an average on 29 problem instances). We marked these cases by an asterisk (*) in the tables.

Table 4.2 shows the results related to smaller size problem instances in test bed 1. In the table, the first two columns show the characteristics of the problem instances, that is, the size of problem instances ($m \times n$) and the interval width (δ). The first five grouped columns report the average gap from the optimum, and the last seven grouped columns display the average running time (sec.). Every value in the table denotes an average on 30 problem instances, except those values marked by an asterisk which correspond to an average on 29 problem instances. As it was expected, we can see that the average running times of CPLEX and YALMIP dramatically increase by increasing the size of the problem instances, with an average of, respectively, 134.37 seconds and 363.56 seconds for the problem instances of size 40×60. However, BI , GR and RLT1 are quite fast with a maximum average computation time of less than one second. As can be noted, computation times of FMIN and RLT2 grow by changing the size of problem instances, but in a less steeper manner compared to CPLEX and YALMIP. In particular, the average running time of FMIN ranges between 0.23 and 3.92 seconds and that of RLT2 between 0.22 and 4.72 seconds. Regarding the gap from the optimal value, BI returns tighter inner approximations than FMIN in 34 rows out of a total of 35, and in the worst case its average gap from the optimum is 0.072. Moreover, GR outperforms FMIN in 28 rows out of 35 rows, while in the worst case its average gap is 0.44 which is higher than that of BI. In short, the order of algorithms in terms of the tightness of the inner approximations is BI, GR, and FMIN. In contrast, RLT1 and RLT2 do not tend to return very tight approximations in general. However, the quality of RLT2 seems more reasonable than RLT1, which is also paid off by the computation time.

Table 4.3 highlights the results related to larger size problem instances in test bed 1. Our findings from Table 4.2 show that RLT1 & RLT2 do not return a tight approximation for smaller size problem instances. Thus, for this set of problem instances, we only focus on the performance of the inner approximation methods. As mentioned earlier, the optimal values for these problem instances are not known. Hence, we benchmarked the results of FMIN to evaluate the other two algorithms. Particularly, given an inner approximation ($\underline{f}^*$) attained by BI or GR, if it outperforms an estimation found by FMIN ($\underline{f}^{**}$), i.e., $\underline{f}^* \leq \underline{f}^{**}$, we computed their relative distance by

$$\text{distance} := \frac{\underline{f}^{**} - \underline{f}^*}{\left|\underline{f}^{**}\right|}.$$

Thus, a higher distance measure indicates a better performance of BI or GR compared to FMIN. In Table 4.3, for the algorithms BI and GR, we report the number of problem instances for which each algorithm outperforms FMIN (column freq.) and also the average distance from FMIN (column avg. dist.). The last three columns display the average running times for the three methods. As an example, for the problem size 50×70 with $\delta = 5$, GR outperformed on 22 problem instances out of a total of 30 problem instances, and the average distance from FMIN computed on those 22 problem instances is 0.0457. It is worth noting that in each row of the table, the average running time was computed on all the 30 problem instances. In total, BI outperformed on 1,005 problem instances out of a total of 1,050 problem instances with the maximum average distance of 0.1368, and GR outperformed on 861 problem instances with the maximum average distance of 0.2273. While the frequency of the outperformance is persistent for BI, the number tends to drop for GR as δ increases. Moreover, BI and GR converged to a solution very fast with a maximum average running time of, respectively, 1.05 seconds and 0.3 of a second. However, FMIN is less computationally efficient with the average running time ranges from 3.53 seconds to 185.20 seconds.

We represent the results related to test bed 2 in Table 4.4. The problem instances in test bed 2 are of importance since their matrix A is an arc-node incidence matrix, i.e., coefficient matrices are sparse. For test bed 2, we focused on the problem instances for which we were able to get the optimal values thorough CPLEX. For this class of problem instances, we tried wider δ compared to the previous test bed. Similar to Table 4.2, Table 4.4 reports the average gap from the optimum and the average running time. The algorithms showed the same behaviour here as they did in test bed 1, that is, BI and GR are still promising in computing a cheap but tight approximation, while FMIN returned reasonable solutions in a much longer time (see, for example, problem instances of size 39×400 in the table). Our findings also confirm that while RLT1 and RLT2 are fast, they do not yield very usable estimations.

Problem instances in test bed 3 are distinguished from the other two test beds in the following aspects: (i) they are larger in size, (ii) sparsity occurs in both A and c, and (iii) interval vector $\mathbf{b}$ was constructed differently. Given the poor performance of RLT1 and RLT2, we again here concentrate on BI, GR, and FMIN. For problems enlight13 and enlight15, FMIN was able to return a solution; however, it failed to converge to even a feasible solution within the time limit for mik-250-1-100-1 and roll3000. Thus, for problems enlight13 and enlight15, we benchmarked FMIN and summarized our results in Table 4.5. In the table, the first column represents the model names, the second column denotes the number of generated samples based on which we constructed interval vector $\mathbf{b}$, and the third column shows an average δ obtained over all the right hand sides. It is worth noting that each row of the table corresponds to only one problem instance, that is, Table 4.5 reports results for 16 problem instances. The following two columns denote the distance measure. Note that a negative value in the table means outperformance of FMIN. The last three columns report the running time. As the results in the table read, BI returns a better solution compared to FMIN on every problem instance. GR outperformed FMIN on 10 problem instances. Furthermore, GR is the cheapest algorithm among all with a maximum running time of 0.32 of a second, while FMIN is the most expensive one with the running times range from 91.46 seconds to 857.06 seconds.

Table 4.6 outlines the results on mik-250-1-100-1 and roll3000. Since FMIN was not able to return any solution within the time limit, we here benchmarked GR and reported the obtained distance measures for BI (column 4). We also displayed the running times in the last two columns of the table. Similar to Table 4.5, Table 4.6 corresponds to 16 problem instances. For mik-250-1-100-1, both algorithms run fast, but BI outperformed GR, in terms of quality of solutions, on five problem instances out of a total of eight problem instances. In the case of roll3000, we can see from the table that BI and GR yielded more or less same solutions, but BI took significantly longer time than GR to converge, with an average running time of 196.19 seconds for BI versus that of 7.96 seconds for GR.

4.8 Concluding remarks

Quantifying extra functions (i.e., outcome functions) over optimal solutions of an optimization problem can be of great value in practice since it provides decision makers with additional information on a system. This becomes even more relevant when input parameters of an optimization problem are subject to uncertainty, which can often cause the optimal solutions to change and consequently impair results of outcome functions. In this paper, we considered uncertainty in the form of interval data. In particular, we addressed the outcome range problem which consists of finding the lower and upper bounds of an outcome function of interest over the set of all the possible optimal solutions of a linear program with interval data. We narrowed down our study on linear programming problems with interval right-hand sides, motivated by the fact that uncertainty usually, in real-world problems, only affects the right-hand sides of the constraints. We investigated the outcome range problem for this special case. We discussed the computational complexity of the problem, showing that the problem is non-trivial. We also explored some of its theoretical properties related to some characteristics of the problem in the scenario space. The outcome range problem can have overlaps with other known problems such as optimal value range problem, bilevel optimization, and multiobjective optimization. We formally established the relation between the outcome range problem and the aforementioned problems. Moreover, we developed several heuristics to solve the problem. Specifically, we presented a nonlinear formulation of the outcome range problem and employed the reformulation-linearization technique to linearize the problem. Our numerical experiments show that this approach does not lead to a very tight outer approximation and could be even computationally inefficient depending how we perform the reformulation phase. To estimate the range of an outcome function from inside, we designed a gradient-restoration based algorithm and a bases inspection approach. Our results show that these algorithms are promising both in terms of quality of the solution and the running time.

We see several future directions for this work. From the computational perspective, there is room for improvement of the outer approximation. Having a tight outer approximation is particularly important in designing an exact algorithm for the problem. From the theoretical standpoint, it seems promising to study the outcome range problem with the uncertainty set described as a general convex polyhedron. We already showed in Section 4.4 that the union of all scenarios for which there exists a weakly feasible solution is a bounded convex polyhedron. Hence, it would be interesting to investigate the problem under such a generalization. Furthermore, in practical applications, uncertainty may also affect the objective function coefficients. Thus, another future direction could be addressing the outcome range problem where intervals occur in the objective function and the right-hand side vector of the underlying linear program.

Table 4.2: Results related to *smaller* size problem instances in test bed 1. An asterisk (*) denotes an average on 29 problem instances.

size ($m \times n$)	δ	average gap from the optimum					average running time (sec.)						
		BI	GR	FMIN	RLT1	RLT2	BI	GR	FMIN	RLT1	RLT2	MILP	Bilevel
	0.01	0.0000	0.0000	0.0000	0.0000	0.0000	0.0873	0.0363	0.2859	0.1890	0.2944	0.0213	0.0770
	0.1	0.0000	0.0000	0.0001	0.0000	0.0000	0.0822	0.0359	0.2261	0.0935	0.2305	0.0197	0.0651
	0.5	0.0000	0.0000	0.0004	0.0000	0.0000	0.0841	0.0383	0.2486	0.0927	0.2248	0.0219	0.0624
5×10	1	0.0000	0.0000	0.0005	0.0031	0.0001	0.0774	0.0379	0.2487	0.0926	0.2341	0.0199	0.0653
	2	0.0000	0.0000	0.0090	0.0238	0.0000	0.0794	0.0342	0.2507	0.0928	0.2237	0.0192	0.0642
	3	0.0000	0.0158	0.0182	0.0864	0.0224	0.0798	0.0359	0.2467	0.0902	0.2273	0.0177	0.0643
	5	0.0000	0.1689	0.0061	0.6498	0.0000	0.0821	0.0365	0.2265	0.0935	0.2280	0.0209	0.0650
	0.01	0.0000	0.0000	0.0000	0.0000	0.0000	0.0746	0.0373	0.3526	0.0908	0.3861	0.0505	0.0837
	0.1	0.0000	0.0000	0.0000	0.0036	0.0000	0.0787	0.0352	0.3590	0.0913	0.4004	0.1012	0.1107
	0.5	0.0000	0.0006	0.0017	0.0045	0.0000	0.0760	0.0376	0.5149	0.0908	0.3873	0.0428	0.0878
10×20	1	0.0000	0.0000	0.0018	0.0431	0.0000	0.0751	0.0353	0.4407	0.0927	0.3916	0.0537	0.0893
	2	0.0000	0.0001	0.0012	0.1455	0.0000	0.0780	0.0357	0.6320	0.0906	0.3974	0.0427	0.1030
	3	0.0000	0.0007	0.0026	0.0486	0.0008	0.0876	0.0351	0.5148	0.0895	0.3920	0.0411	0.0906
	5	0.0034	0.0046	0.0017	0.3498*	0.0102	0.0864	0.0363	0.3419	0.0890	0.3898	0.0475	0.0927
	0.01	0.0000	0.0000	0.0000	0.0000	0.0000	0.0799	0.0530	0.9929	0.0937	1.7302	0.2901	0.3169
	0.1	0.0000	0.0000	0.0000	0.0016	0.0000	0.0839	0.0496	1.1156	0.0980	1.7686	0.2953	0.3120
	0.5	0.0003	0.0003	0.0006	0.1139	0.0020	0.0951	0.0471	1.9141	0.0983	1.8384	0.2760	0.3469
30×40	1	0.0000	0.0006	0.0033	0.3051	0.0030	0.0830	0.0463	2.0208	0.0912	2.2073	0.2547	0.3036
	2	0.0003	0.0018	0.0022	0.5165	0.0075	0.0929	0.0484	1.5685	0.0935	1.8376	0.3004	0.3397
	3	0.0013	0.0602	0.0061	0.6378	0.0379	0.1068	0.0454	1.7416	0.0967	1.9524	0.2687	0.3311
	5	0.0718	0.1550	0.1295	1.1998*	0.0959	0.1096	0.0437	1.6457	0.0978	1.9345	0.2906	0.3659
	0.01	0.0000	0.0000	0.0000	0.0000	0.0000	0.0817	0.0516	1.6714	0.0972	2.5672	3.8792	5.1899
	0.1	0.0000	0.0000	0.0002	0.0041	0.0000	0.0835	0.0510	2.1046	0.1002	2.7249	4.7702	6.2908
	0.5	0.0000	0.0001	0.0011	0.2572	0.0010	0.0865	0.0477	2.5242	0.1017	2.7380	4.4642	5.1958
35×50	1	0.0015	0.0022	0.0027	0.2257	0.0008	0.0913	0.0512	2.6540	0.0979	2.8993	3.4865	5.5339
	2	0.0000	0.0000	0.0268	0.4280*	0.0192	0.0943	0.0525	3.1656	0.0996	2.9291	3.6767	5.5078
	3	0.0006	0.0149	0.0064	0.6389*	0.0489	0.1055	0.0561	2.8826	0.0991	2.8787	3.3620	4.8888
	5	0.0016	0.4397	0.4686	0.9214*	0.0669*	0.1246	0.0508	2.0565	0.0971	2.9162	4.3270	5.7803
	0.01	0.0000	0.0000	0.0001	0.0002	0.0000	0.0923	0.0575	2.0462	0.0996	3.8424	75.6026	187.0493
	0.1	0.0084	0.0084	0.0088	0.0219	0.0002	0.0909	0.0580	2.7956	0.1004	4.0631	179.0062	462.5520
	0.5	0.0001	0.0001	0.0081	0.2882	0.0003	0.0935	0.0607	3.9244	0.1038	4.5062	84.4636	295.5671
40×60	1	0.0000	0.0031	0.0019	0.1636	0.0029	0.1022	0.0617	3.5551	0.0993	4.5801	128.3584	364.8113
	2	0.0016	0.0006	0.0645	0.5998*	0.0415*	0.1111	0.0569	3.4730	0.1016	4.3808	68.0187	199.7742
	3	0.0001	0.0080	0.0214	0.9694*	0.2194	0.1178	0.0558	3.6880	0.1082	4.6443	164.9271	370.6350
	5	0.0060	0.0246*	0.0121	0.7669*	0.9710	0.1376	0.0577	3.3003	0.1026	4.7164	240.2099	664.5236

Table 4.3: Results related to *larger* size problem instances in test bed 1.

size ($m \times n$)	δ	BI		GR		average running time (sec.)		
		freq.	avg. dist.	freq.	avg. dist.	BI	GR	FMIN
	0.01	30	0.0001	30	0.0001	0.1056	0.0643	3.5349
	0.1	30	0.0002	30	0.0002	0.0972	0.0669	4.2482
	0.5	30	0.0047	29	0.0041	0.1054	0.0616	4.6234
50×70	1	29	0.0040	28	0.0040	0.1083	0.0669	4.2916
	2	28	0.0018	25	0.0007	0.1277	0.0662	3.9946
	3	30	0.0132	25	0.0108	0.1412	0.0667	3.9562
	5	29	0.1368	22	0.0457	0.1747	0.0680	4.3333
	0.01	30	0.0000	30	0.0000	0.1245	0.1069	8.2529
	0.1	30	0.0002	29	0.0002	0.1269	0.1073	12.7975
	0.5	30	0.0010	26	0.0006	0.1526	0.1036	9.2031
75×100	1	29	0.0033	27	0.0022	0.1529	0.1033	11.8664
	2	28	0.0110	18	0.0053	0.1834	0.1044	9.9408
	3	24	0.0955	21	0.2273	0.2098	0.1074	12.9671
	5	29	0.0236	24	0.0132	0.2686	0.1130	8.2542
	0.01	30	0.0001	30	0.0001	0.1686	0.1247	20.7063
	0.1	30	0.0003	30	0.0003	0.1718	0.1243	25.0344
	0.5	30	0.0257	22	0.0349	0.2060	0.1219	21.4619
100×125	1	30	0.0061	25	0.0052	0.2425	0.1212	22.3853
	2	25	0.0066	21	0.0044	0.2783	0.1228	16.8147
	3	28	0.0273	21	0.0162	0.2994	0.1316	15.4256
	5	28	0.0252	17	0.0340	0.4149	0.1292	16.1904
	0.01	30	0.0001	30	0.0001	0.2284	0.1941	38.6138
	0.1	30	0.0014	30	0.0014	0.2454	0.1892	42.4585
	0.5	27	0.0049	22	0.0005	0.2735	0.1812	31.8195
125×150	1	29	0.0037	26	0.0029	0.3142	0.1848	25.3423
	2	28	0.0101	22	0.0029	0.3819	0.1926	31.2585
	3	27	0.1368	19	0.1881	0.4442	0.2014	28.9812
	5	25	0.0391	18	0.0158	0.6013	0.2035	27.3384
	0.01	30	0.0004	30	0.0004	0.3495	0.2680	136.6920
	0.1	30	0.0015	28	0.0016	0.3621	0.2661	185.1950
	0.5	30	0.0117	26	0.0083	0.4948	0.2640	143.9452
150×200	1	30	0.0414	25	0.0127	0.5107	0.2671	106.9154
	2	26	0.0485	20	0.0302	0.6148	0.2761	99.1450
	3	29	0.0225	21	0.0187	0.7456	0.2829	75.4403
	5	27	0.0690	14	0.0390	1.0509	0.3005	86.3719

Table 4.4: Comparing algorithms on test bed 2. An asterisk (*) denotes an average on 29 problem instances.

size ($m \times n$)	δ	average gap from the optimum					average running time (sec.)					
		BI	GR	FMIN	RLT1	RLT2	BI	GR	FMIN	RLT1	RLT2	MILP
	5	0.0000	0.0013	0.0000	0.7280	0.0019	0.2309	0.0969	0.2498	0.1933	0.2980	0.0130
5×9	10	0.0000	0.0100	0.0080	0.4368	0.1088	0.2174	0.0837	0.1449	0.0940	0.2301	0.0120
	20	0.0096	0.0101	0.2415	0.6541*	0.0677	0.1818	0.0853	0.1048	0.0888	0.2248	0.0144
	5	0.0000	0.0010	0.0007	0.9433*	0.0729	0.1826	0.1354	0.3142	0.0890	0.3383	0.0261
9×25	10	0.0037	0.0283	0.0446	0.7666*	0.0391	0.2063	0.1474	0.2767	0.0892	0.3358	0.0297
	20	0.3420	0.0355	0.1561	4.0432*	0.6522*	0.3269	0.1480	0.3345	0.0883	0.3373	0.0281
	5	0.0123	0.0008	0.0325	0.9667*	0.0997	0.5164	0.1187	1.3525	0.0908	0.7341	0.4781
19×100	10	0.0075	0.0410	0.0650	2.2254*	0.2577	0.5117	0.1302	1.4178	0.0890	0.7360	0.4854
	20	0.0046	0.0606	0.0269	6.0464*	0.8460*	0.9457	0.1727	1.5092	0.0926	0.7524	0.5937
	5	0.0032	0.0051	0.0176	0.9740	0.1146	0.6752	0.1740	33.7814	0.0954	2.7094	54.9777
39×400	10	0.0489	0.0355	0.0222	1.9538*	0.2681	0.5799	0.1943	33.1993	0.0966	2.8882	73.8604
	20	0.0838	0.0804	0.2677	5.8244*	0.8112*	0.6733	0.1585	30.9603	0.0966	2.9259	118.7083

Table 4.5: Results obtained from test bed 3 (part I)

model	no. of samples	average δ	distance from FMIN		running time (sec.)		
			BI	GR	FMIN	BI	GR
enlight13	100	8.10	0.0046	-0.0904	91.4576	0.6921	0.3063
	500	9.66	0.0658	-0.0633	119.6879	1.8702	0.1685
	1,000	10.54	0.0431	-0.0905	187.4409	1.9958	0.0900
	5,000	11.88	0.1463	-0.0021	173.4079	1.8712	0.1357
	10,000	12.40	0.0562	-0.0822	104.4437	2.0197	0.1277
	50,000	13.67	0.0945	0.0906	212.5396	1.9538	0.2463
	100,000	14.10	0.1743	0.0291	161.4438	1.7396	0.1236
	500,000	15.26	0.1349	0.0801	180.5201	1.8871	0.1176
enlight15	100	9.40	0.1773	-0.1880	272.7402	2.3203	0.2195
	500	11.35	0.2083	0.0140	384.2658	2.4595	0.1049
	1,000	12.23	0.0927	0.0388	412.5042	2.3922	0.1653
	5,000	13.81	0.4270	0.2473	857.0598	2.4483	0.1562
	10,000	14.42	0.4120	0.2953	408.9520	2.4486	0.1694
	50,000	15.93	0.6289	0.4405	437.0599	2.4954	0.1734
	100,000	16.43	0.3316	0.1691	442.2985	2.4671	0.1622
	500,000	17.75	0.2305	0.1942	455.2568	2.5617	0.3205

Table 4.6: Results obtained from test bed 3 (part II).

model	no. of samples	average δ	distance from GR	running time (sec.)	
			BI	BI	GR
mik-250-1-100-1	100	1,958.06	0.0093	5.7573	0.2289
	500	2,272.89	0.0064	5.8112	0.2984
	1,000	2,426.95	0.0104	5.8371	0.3118
	5,000	2,756.56	0.0118	5.7987	0.4776
	10,000	2,884.56	0.0069	5.7627	0.2503
	50,000	3,168.52	-0.0081	5.6830	0.3584
	100,000	3,303.37	-0.0105	5.7294	0.3359
	500,000	3,550.52	-0.0085	5.7810	0.3148
roll3000	100	0.0011	0.0000	194.2224	8.4392
	500	0.0013	0.0000	194.5361	7.8458
	1,000	0.0014	0.0000	201.1792	8.0117
	5,000	0.0016	0.0000	194.5419	7.9430
	10,000	0.0017	0.0000	193.9927	7.9305
	50,000	0.0018	0.0000	194.5831	7.8395
	100,000	0.0019	0.0000	194.9193	7.8310
	500,000	0.0020	0.0000	201.5826	7.8350

CHAPTER 5

FULL LINK FLOW OBSERVABILITY OF TRAFFIC NETWORKS UNDER MEASUREMENT ERROR

5.1 Introduction

The continuous growth in the demand for transportation is the main reason for traffic congestion, air pollution, greenhouse gas emissions, vehicle accidents, and increased fuel costs in large urban areas. Monitoring traffic flow volumes on a network allows to control and manage these undesirable situations. Information on traffic flows is usually obtained through locating sensors on the traffic network. In this context, the problem of optimally locating sensors on traffic networks (referred to as the sensor location problem) to gather data for monitoring and managing traffic volumes has rapidly gained attention since its first introduction in the late 1990s [134] (see [52] for a comprehensive survey on the relevant topics).

Following the classification by Gentili and Mirchandani [52], there are two classes of sensor location problems:

- *Flow-Observability Problems:* locating sensors to fully (partially) observe flow volumes on a traffic network;

- *Flow-Estimation Problems:* locating sensors to estimate flow volumes on a traffic network.

This classification is derived from the observation that the location of sensors on a traffic network can be translated into a system of linear equations where columns (variables) represent traffic flows of interest, and rows correspond to the deployment of sensors and the structure of the network. If the system of linear equations admits a unique solution, we say the network is observable, and, under the assumption of error-free monitored flows, we can uniquely determine all flows of the network. In this context, the flow-observability problem is the problem of finding the optimal placement for sensors which allows the full observability of traffic flows on a traffic network. If the network is not observable (i.e., the system is underdetermined), we have an infinite number of solutions. In this case, the interest is in locating sensors to obtain the best flow estimates according to some quality metrics, see, e.g., [32, 37, 80, 132, 133, 135, 140]. Generally, the flow-estimation problem arises when full observability of a network is not possible due to a budget constraint which limits the number of sensors to be located or when we are interested in determining origin–destination flow volumes by locating counting sensors on the links of the network.

In this paper, our focus is on the flow-observability problem. In general, this class of the problem can be differentiated according to (i) the type of sensors used, (ii) the location of sensors, (iii) the available a priori information, and (iv) the traffic flows of interest. There are two main classes of sensors considered in the literature: counting sensors and scanning sensors. Counting sensors, based on vehicle counts, can monitor volumes, density, occupancy, and speed of traffic volumes; examples of this class includes inductive loop detectors, magnetic detectors, piezoelectric pads, pneumatic road tubes, among others [52]. Scanning sensors, on the other hand, are able

Table 5.1: Contributions in the literature addressing different configurations of the full flow-observability problem.

configurations	sensor type	location	flows of interest	publication
1	counting	node	link flows	[11, 12, 13, 33, 96]
2	counting	link	link flows	[17, 18, 56, 66, 67, 68, 100]
3	counting	node/link	route flows/ link flows/OD trips	[115]
4	scanning	link	route flows	[20, 23, 26, 51]
5	mixed	link	route flows	[39, 118]
6	scanning	link	route flows/OD trips	[54]

to uniquely identify a vehicle, by taking images of moving flows or using Automatic Vehicle Identification readers. Sensors can be located either on links or on nodes depending on the type of sensors. Available a priori information include, among others, link chose proportions (fraction of traffic volumes using a given link), split ratios (fraction of outgoing flows of a given node using a specific outgoing link), set of routes. Such information can be attained through survey data, historical data, data from previously located sensors, or calibrated static traffic assignment models. Finally, traffic flows of interest are usually origin-destination (OD) flow volumes, link flow volumes, route flow volumes, or a combination of them. Table 5.1 summarizes the contributions in the literature addressing various configurations of the full flow-observability problem. It is also worth mentioning that a special class of the flow-observability problem is the partial flow-observability problem where sensors are placed sufficient that a predefined number of flows of interest can be deduced from sensor readings, see, e.g., [21, 24, 92, 101, 115].

Here, we particularly ***focus on the problem of locating counting sensors on a subset of the links of the network to fully observe all the link flow volumes***. Hu et al. [67] considered the full link flow observability problem assuming route information is given and proposed a Gaussian elimination method to solve it. Ng [100] presented a node-based approach to address the problem where the route information is not needed; he also extended the approach to the partial link flow observability problem [101]. Also, the graphical approach proposed in [56] to solve the link flow observability problem does not require knowledge of route information; the problem was basically reduced to the problem of finding a spanning tree. Castillo et al. [17, 18], using the concept of linearly independent paths, determined an upper bound on the number of sensors to be located on the links to allow full link flow observability. Hu et al. [68] studied the problem assuming a priori information on turning ratios at nodes and developed a greedy algorithm to solve it. Rubin and Gentili [115] proposed a general mathematical formulation of the flow-observability problem, limited to locating counting sensors, but allowing any type of a priori information, any type of flows of interest, and any type of placements for sensors; their approach can be adopted to solve the full link flow observability problem.

A common assumption to all the existing works on the flow observablity problem is that the data gathered from the sensors does not contain measurement errors (i.e., errors in traffic counts). To the best of our knowledge, there are only few papers addressing the presence of errors in the sensor data. Castillo et al. [20] consider scanning errors when locating vehicle-ID sensors to observe route flow volumes. This case errors occur because visibility and weather conditions have a major impact on the efficiency of the image-based vehicle-ID sensors. They propose the idea of locating more than one sensor on the same link to identify (and reduce) the number of assignment errors in the image scanning process. In a more recent study, Xu et al. [130] studied the problem of locating sensors on a subset of the links of the traffic network to observe all the link flows, where they implicitly consider measurement errors. In particular, they proposed a robust approach minimizing the number of unobserved links involved in the nodal flow conservation equations. They discussed that the results of their indirect approach will have a lower chance of accumulating measurement errors. Unlike the latter approach, this paper *explicitly* addresses the measurement errors in the full link flow observability problem. We propose an optimization approach which describes the measurement errors by compact intervals and seeks for locations for sensors with minimum uncertainty in the inferred link flows. Given the complexity of our proposed optimization problem, we present a local search algorithm to solve it. Through numerical experiments, we show that our approach returns more reliable inferred link flows compared to those obtained using the approach proposed in [130]. In short, we summarize our contributions as follows:

- we motivate and propose an optimization problem to address the full link flow observability problem considering the measurement errors;
- we show that our approach outperforms the only existing approach in this context;
- we develop an algorithmic scheme to solve our problem and show its efficiency on some real traffic networks.

This paper is structured as follows. Section 5.2 presents the link flow inference problem. Section 5.3 motivates the need for directly addressing the measurement errors in the link flow observability problem and proposes an optimization approach to handle the measurement errors in this context. In Section 5.4, we develop a local search algorithm to solve our optimization problem. Our numerical study is presented in Section 5.5 to show the applicability of our approach. Finally, we summarize our concluding remarks in Section 5.6.

5.2 Link flow inference

The link flow inference problem is the problem of placing sensors on a subset of the links of the network sufficient that the unobserved link flows (i.e., those links not equipped with sensors) can be inferred from the observed link flows. We here adopt the node-based approach by Ng [100] which does not require route information. A traffic network can be represented by means of a graph $\mathcal{H} = (N, E)$ where N represents the nodes in the network (e.g., intersections points) and E represents links (e.g., streets of the network). For a given traffic network represented by a graph $\mathcal{H}$, let us consider the graph $\mathcal{G} = (N^*, E)$ where $N^* \subseteq N$ denotes non-centroid nodes. Centroid nodes in the traffic network are the nodes originating or absorbing the traffic flows, and non-centroid nodes are all other nodes in the network. Let us assume that $|N^*| = n$ and $|E| = m$ with $m >> n$ (this is a realistic assumption in traffic networks). Let $A \in \mathbb{R}^{n\times m}$ be the non-centroid node-link incidence matrix; we define the entries as

$$a_{ij} = \begin{cases} 1 & j \in I(i) \\ -1 & j \in O(i) \\ 0 & \text{otherwise} \end{cases},$$

where $I(i)$ and $O(i)$ respectively represent the set of incoming and outgoing links at node $i \in N^*$. We can then write the flow conservation equations for the non-centroid nodes in the matrix form as

$$Af = 0, \tag{5.1}$$

where $f \in \mathbb{R}^m$ is the decision vector corresponding to the link flows. Let us partition A into $A_U \in \mathbb{R}^{n\times n}$ and $A_O \in \mathbb{R}^{n\times(m-n)}$ such that A_U is non-singular. We also properly partition f into f_U and f_O. We then have

$$(A_U, A_O)\begin{pmatrix} f_U \\ f_O \end{pmatrix} = 0,$$

which can be rearranged as

$$f_U = -A_U^{-1}A_O f_O. \tag{5.2}$$

This implies that by locating sensors on links O, we can infer link flows of links U. From the algebraic standpoint, the set of solutions to (5.1) describes the null space of A and (5.2) can be reduced to finding a basis for the column space of matrix A.

Assumption 5.1. *We assume that the network is connected, and there exists at least one centroid node in the network.*

This is a realistic assumption since in a given traffic network, there are usually nodes that originate or absorb the traffic flows; basically, having centroid nodes make the traffic flow observability problem meaningful.

Remark 5.1. We know that the rank of the node incidence matrix of a connected directed graph with p nodes (including centroid and non-centroid nodes) is $p-1$ (see [98]). Given Assumption 5.1 and the assumption of $m >> n$, we can say that the rank of matrix A is n. Hence, it is always possible to partition A to A_U and A_O where A_U is non-singular. Also, this implies that to uniquely observe all the link flows, we need to locate a minimum of $(m-n)$ sensors on the links of the network. That is, we have $|U| = n$ and $|O| = m - n$ [100].

Example 5.1. Consider the network of Figure 5.1 with six nodes and nine links. Let us assume that nodes 1 and 3 are centroid nodes in this network. Hence, we can write the non-centroid node-link incidence matrix as

$$A = \begin{array}{c} \\ 2 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{pmatrix} 1 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{pmatrix} \end{array},$$

where rows correspond to non-centroid nodes and columns denote links. Suppose we

Table 5.2: (Example 5.1) True link flows

links	1	2	3	4	5	6	7	8	9
true flow	27	7	5	11	11	5	18	12	6

locate sensors on links in $O^1 = \{1, 2, 5, 6, 9\}$; we then partition A into

$$A_{O^1} = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad A_{U^1} = \begin{pmatrix} -1 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}.$$

Now let us assume the sensors located on links in O^1 are error-free, and therefore they returns the true link flows (they are reported in Table 5.2). Then, by (5.2) we can infer the unobserved link flows, that is,

$$f_3 = 5,\ f_4 = 11,\ f_7 = 18,\ f_8 = 12.$$

As an alternative set of sensor placements, if we locate sensors on links in $O^2 = \{1, 3, 5, 6, 8\}$, we similarly can partition matrix A and infer the following unobserved flows:

$$f_2 = 7,\ f_4 = 11,\ f_7 = 18,\ f_9 = 6.$$

Indeed, for this simple example, 66 different sets of sensor placements with minimum cardinality allow full observability of the network. However, in the case of erroneous observed flows, not all the 66 sets of sensor placements lead to a meaningful flow inference. We further elaborate this in the following section.

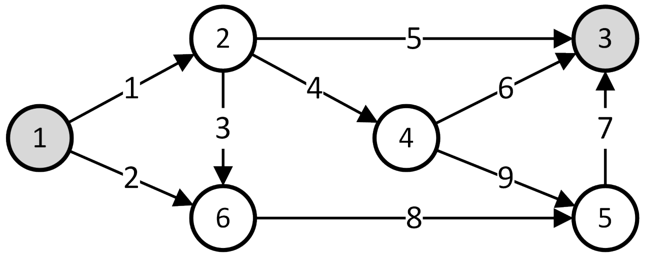


Figure 5.1: A network example.

5.3 Link flow inference under measurement error

Suppose in Example 5.1 the sensor on link 1 reports 20 instead of 27. For the sensor placement O^1, the inferred link flows would be then

$$f_3 = -2,\ f_4 = 11,\ f_7 = 11,\ f_8 = 5,$$

and for the placement O^2, we would have

$$f_2 = 7,\ f_4 = 4,\ f_7 = 11,\ f_9 = -1.$$

The two sets of the inferred link flows are not valid estimations as they contain negative link flows, resulting in useless information. However, inferred link flows obtained by another sensor placement $O^3 = \{1, 2, 4, 8, 9\}$ is:

$$f_3 = 5,\ f_5 = 4,\ f_6 = 5,\ f_7 = 18.$$

As can be noted, only the inferred flow of link 5 was impacted by the error, but still the inferred flows are useful. A natural question that arises in this context is

> *where to locate sensors such that the resulting inferred flows are valid and the effect of erroneous data on the unobserved link flows is minimized?*

To answer the above question effectively, we need to explicitly address the measurement errors in sensor readings. To this end, we assume that readings of sensors lie in predefined intervals. These intervals correspond to some a priori knowledge of the link flows. This is a reasonable assumption which reflects the level of trust of the decision maker on the available information. Indeed, prior flow estimates are usually available through empirical observations, or they can be attained by using some models. A confidence interval that describes the uncertainty or accuracy of the prior estimates is usually associated with them [126]. Let us assume a priori link flows are described by compact intervals, that is, we can denote the readings as

$$\mathbf{f} = [\underline{f}, \overline{f}] := \{f \in \mathbb{R}^m : \ \underline{f} \leq f \leq \overline{f}\},$$

where $\underline{f}, \overline{f} \in \mathbb{R}^m$ are given and inequality "$\leq$" is understood componentwise. Let the symbol $\mathbb{IR}$ denote the set of all real intervals. For two real intervals $\mathbf{f}_i, \mathbf{f}_j \in \mathbb{IR}$ and constant $\alpha \in \mathbb{R}$, we have the following interval arithmetic operations

$$\mathbf{f}_i \pm \mathbf{f}_j = [\underline{f}_i \pm \underline{f}_j, \overline{f}_i \pm \overline{f}_j], \quad \alpha.\mathbf{f}_i = [\min\{\alpha\underline{f}_i, \alpha\overline{f}_i\}, \max\{\alpha\underline{f}_i, \alpha\overline{f}_i\}]. \tag{5.3}$$

Thus, for a specific set of sensor placements O, we can infer the interval link flow on links U by (5.3) as

$$\mathbf{f}_U^* = [\underline{f}_u^*, \overline{f}_u^*] = -A_U^{-1} A_O \mathbf{f}_O, \tag{5.4}$$

where $\mathbf{f}_U^*$ denotes the vector of inferred interval link flows.

Remark 5.2. In real-life, sensor readings can only report integer values in interval $\mathbf{f}$. Hence, it is reasonable to assume the a priori interval vector $\mathbf{f}$ has integer upper and lower bounds.

Definition 5.1. A network with unobserved links U and observed links $O = E \backslash U$ is called weakly observable if $\mathbf{f}_U^* = -A_U^{-1} A_O \mathbf{f}_O$ admits non-negative flows for some $f_U^* \in \mathbf{f}_U^*$.

There could be many different sets U and $O = E\backslash U$ for which the network is weakly observable. We denote by $\mathcal{W}$ the set of all possible U and O allowing weak observability, that is,

$$\mathcal{W} := \{U \subset E : \; Af = 0, \; f_U \geq 0, \; f_O \in \mathbf{f}_O, \; O = E\backslash U\}.$$

Our goal is to find a set of sensor placements such that the variability of the inferred link flows is minimum. There are different ways of evaluating such a variability (see Section 5.5), one of which is, for example, obtained by considering the width of the resulting inferred interval link flows associated with the unobserved links. Specifically, given an interval vector of inferred flows $[\underline{f}^*_U, \overline{f}^*_U]$, a measure for assessing the variability of the inferred flows can be attained by

$$\sum_{i=1}^{n} (\overline{f}^*_{U(i)} - \underline{f}^*_{U(i)}); \tag{5.5}$$

the lower the measure, the better the sensor placement.

As an illustration, let us consider that in Example 5.1, we have the a priori interval link flows as presented in Table 5.3. In the real-world context, this information is usually available from historical data . Note that depending on the quality of the available information, the interval link flows do not necessarily contain the true link flows. For example, in Table 5.3, interval flows corresponding to links $\{5, 6, 8, 9\}$ do not contain the true flow of the links (see Table 5.2). Table 5.4 summarizes the set of all observed and inferred link flows for which the network is weakly observable. As it can be seen, with the given data, out of 66 sets of sensor placements, only 22 sets of sensor placements ensure weak observability of the network. The last column in the table display the values of equation (5.5) associated with each location set. The set of sensor placements $\{2, 3, 4, 5, 7\}$ minimizes the variability in the inferred flows.

Table 5.3: A priori interval link flows.

links	1	2	3	4	5	6	7	8	9
$[\underline{f}, \overline{f}]$	[22,32]	[6,8]	[4,6]	[9,13]	[21,29]	[15,21]	[14,22]	[22,31]	[16,22]

Table 5.4: The set of all observed and inferred link flows leading to weak observability of the network.

no.	observed link flows	inferred link flows	variability
1	$\{f_1, f_2, f_3, f_4, f_7\}$	$\mathbf{f_5}^* = [3,19]$, $\mathbf{f_6}^* = [-3,13]$, $\mathbf{f_8}^* = [10,14]$, $\mathbf{f_9}^* = [0,12]$	48
2	$\{f_1, f_2, f_3, f_5, f_7\}$	$\mathbf{f_4}^* = [-13,7]$, $\mathbf{f_6}^* = [-23,5]$, $\mathbf{f_8}^* = [10,14]$, $\mathbf{f_9}^* = [0,12]$	64
3	$\{f_1, f_2, f_3, f_6, f_7\}$	$\mathbf{f_4}^* = [15,33]$, $\mathbf{f_5}^* = [-15,11]$, $\mathbf{f_8}^* = [10,14]$, $\mathbf{f_9}^* = [0,12]$	60
4	$\{f_1, f_2, f_4, f_5, f_7\}$	$\mathbf{f_3}^* = [-20,2]$, $\mathbf{f_6}^* = [-23,5]$, $\mathbf{f_8}^* = [-14,10]$, $\mathbf{f_9}^* = [4,36]$	106
5	$\{f_1, f_2, f_4, f_7, f_8\}$	$\mathbf{f_3}^* = [14,25]$, $\mathbf{f_5}^* = [-16,9]$, $\mathbf{f_6}^* = [9,30]$, $\mathbf{f_9}^* = [-17,0]$	74
6	$\{f_1, f_2, f_6, f_7, f_8\}$	$\mathbf{f_3}^* = [14,25]$, $\mathbf{f_4}^* = [-2,21]$, $\mathbf{f_5}^* = [-15,11]$, $\mathbf{f_9}^* = [-17,0]$	77
7	$\{f_1, f_2, f_6, f_7, f_9\}$	$\mathbf{f_3}^* = [-16,0]$, $\mathbf{f_4}^* = [31,43]$, $\mathbf{f_5}^* = [-15,11]$, $\mathbf{f_8}^* = [-8,6]$	68
8	$\{f_1, f_3, f_4, f_7, f_8\}$	$\mathbf{f_2}^* = [16,27]$, $\mathbf{f_5}^* = [3,19]$, $\mathbf{f_6}^* = [9,30]$, $\mathbf{f_9}^* = [-17,0]$	65
9	$\{f_1, f_3, f_5, f_7, f_8\}$	$\mathbf{f_2}^* = [16,27]$, $\mathbf{f_4}^* = [-13,7]$, $\mathbf{f_6}^* = [-13,24]$, $\mathbf{f_9}^* = [-17,0]$	85
10	$\{f_1, f_3, f_6, f_7, f_8\}$	$\mathbf{f_2}^* = [16,27]$, $\mathbf{f_4}^* = [-2,21]$, $\mathbf{f_5}^* = [-5,30]$, $\mathbf{f_9}^* = [-17,0]$	86
11	$\{f_1, f_4, f_5, f_7, f_8\}$	$\mathbf{f_2}^* = [20,51]$, $\mathbf{f_3}^* = [-20,2]$, $\mathbf{f_6}^* = [9,30]$, $\mathbf{f_9}^* = [-17,0]$	91
12	$\{f_2, f_3, f_4, f_5, f_7\}$	$\mathbf{f_1}^* = [34,48]$, $\mathbf{f_6}^* = [-3,13]$, $\mathbf{f_8}^* = [10,14]$, $\mathbf{f_9}^* = [0,12]$	46
13	$\{f_2, f_3, f_5, f_6, f_7\}$	$\mathbf{f_1}^* = [42,66]$, $\mathbf{f_4}^* = [15,33]$, $\mathbf{f_8}^* = [10,14]$, $\mathbf{f_9}^* = [0,12]$	58
14	$\{f_2, f_3, f_5, f_6, f_9\}$	$\mathbf{f_1}^* = [56,78]$, $\mathbf{f_4}^* = [31,43]$, $\mathbf{f_7}^* = [26,36]$, $\mathbf{f_8}^* = [10,14]$	48
15	$\{f_2, f_4, f_5, f_7, f_8\}$	$\mathbf{f_1}^* = [44,67]$, $\mathbf{f_3}^* = [14,25]$, $\mathbf{f_6}^* = [9,30]$, $\mathbf{f_9}^* = [-17,0]$	72
16	$\{f_2, f_5, f_6, f_7, f_8\}$	$\mathbf{f_1}^* = [42,66]$, $\mathbf{f_3}^* = [14,25]$, $\mathbf{f_4}^* = [-2,21]$, $\mathbf{f_9}^* = [-17,0]$	75
17	$\{f_2, f_5, f_6, f_7, f_9\}$	$\mathbf{f_1}^* = [42,66]$, $\mathbf{f_3}^* = [-16,0]$, $\mathbf{f_4}^* = [31,43]$, $\mathbf{f_8}^* = [-8,6]$	66
18	$\{f_2, f_5, f_6, f_8, f_9\}$	$\mathbf{f_1}^* = [66,97]$, $\mathbf{f_3}^* = [14,25]$, $\mathbf{f_4}^* = [31,43]$, $\mathbf{f_7}^* = [38,53]$	69
19	$\{f_3, f_4, f_5, f_7, f_8\}$	$\mathbf{f_1}^* = [34,48]$, $\mathbf{f_2}^* = [16,27]$, $\mathbf{f_6}^* = [9,30]$, $\mathbf{f_9}^* = [-17,0]$	63
20	$\{f_3, f_5, f_6, f_7, f_8\}$	$\mathbf{f_1}^* = [23,56]$, $\mathbf{f_2}^* = [16,27]$, $\mathbf{f_4}^* = [-2,21]$, $\mathbf{f_9}^* = [-17,0]$	84
21	$\{f_3, f_5, f_6, f_7, f_9\}$	$\mathbf{f_1}^* = [56,78]$, $\mathbf{f_2}^* = [-14,2]$, $\mathbf{f_4}^* = [31,43]$, $\mathbf{f_8}^* = [-8,6]$	64
22	$\{f_3, f_5, f_6, f_8, f_9\}$	$\mathbf{f_1}^* = [56,78]$, $\mathbf{f_2}^* = [16,27]$, $\mathbf{f_4}^* = [31,43]$, $\mathbf{f_7}^* = [38,53]$	60

Our goal is then to solve the following optimization problem:

$$\min_{U \in \mathcal{W}} \sum_{i=1}^{n} (\overline{f}^*_{U(i)} - \underline{f}^*_{U(i)}). \tag{5.6}$$

Note that sensor readings are not known until after locating them, and they provide a point value for the observed flows, and hence we can infer point estimations for unobserved links. One of the main advantages of using our approach is that the resulting sensor placements reduce the chance of having negative point estimations for unobserved links.

5.4 Local search algorithm

As can be noticed, the difficulty in solving problem (5.6) relies on the fact that the feasible set $\mathcal{W}$ is not explicitly known. As such, we design a local search algorithm to solve the problem. Recall that a feasible solution of the problem is a set $U \in \mathcal{W}$, that is, a basis of the column space of matrix A. From a high-level perspective, starting from an initial feasible solution, we iteratively search in the neighborhood of a basis to find a set of sensor placements with lower variability in the inferred link flows. If we find a better set of sensor placements, we move to the new basis and start over; otherwise, we continue our search. We continue this procedure until a predefined stopping condition is met. To move from one basis to a neighboring one, our algorithm uses a simplex-like tableau to explore different sensor location candidates. Particularly, we consider a simplex tableau where basic variables represent the unobserved link flows, while non-basic variables denote the observed link flows. We now discuss the steps of the algorithm in detail. A general pseudo code of the algorithm is also given in Algorithm 6.

Initial tableau. To get an initial tableau, we can put matrix A into its reduced row echelon form (RREF) for a random order of columns. We then consider the link flows corresponding to the basis of the column space as the unobserved links and check weather the network is weakly observable for this case. If so, the RREF can be used to set up the initial simplex-like tableau, where basic variables are those associated with the basis of the column space. Otherwise, we repeat the process with a different order of columns of matrix A. We continue this procedure until we get a valid initial tableau (i.e., an initial set of sensor placements which allows weak obervability of the network). Note that we can get a different basis of the column space with a different order of columns of matrix A (Ng [100] also mentioned that matrix A_U is not necessarily unique).

Neighborhood Search. We define the neighborhood function as the function of unobserved links (i.e., the basic variables in the tableau). Specifically, two sets of unobserved links are neighbors if they are different in only one link. Hence, our neighborhood is defined as

$$N(U) = \{U' \subseteq E : \ |U' \cap U| = n-1\}.$$

In a tableau, a number of neighbors is equal to the number of non-zero elements in non-basic columns. To move to a new basis, we first check whether the new set of unobserved link flows will lead to weak observability of the network. This can be done by solving the following linear program with a dummy objective function:

$$\min \ 0^T f \ \text{ subject to } \ Af = 0, \ f_O \in \mathbf{f}_O, \ fu \geq 0. \tag{5.7}$$

If the network is weakly observable, we then compute the corresponding objective function using (5.4). We apply a *first improvement* strategy, and therefore we move to the new basis if the objective function value is better than the current value. Note that we do not need to explicitly compute the term $A_U^{-1}A_O$ in (5.4) in each iteration since it can be derived from the tableau after performing a pivoting operation. From the computational standpoint, the simplex-like tableau plays an important role as it enables us (i) to ensure A_U is non-singular and (ii) to compute $A_U^{-1}A_O$ without explicitly computing matrix inverse. To show how our algorithm works, we now discuss one iteration of the algorithm on an example.

Example 5.2. Consider the network of Example 5.1 for which the interval link flows are reported in Table 5.3. Suppose we initiate the algorithm with the set of unobserved link flows $\{1, 2, 4, 8\}$ with the objective function value of 64. Table 5.5 represents the initial tableau, where the rows are unobserved (basic) links. According to the table,

Algorithm 5.1: A local search algorithm

```
1  Input: $A$, $\overline{f}$, $\underline{f}$
   Result: a set of sensor placements
2  while a stopping condition is not reached do
3      Initialize a simplex-like tableau
4      Compute the corresponding objective function value
5      while we can improve the objective function do
6          Select a neighboring basis from the tableau
7          if the network is weakly observable for the selected basis then
8              Compute the objective function value
9              if it is better than the current value then
10                 Update the objective function value and the tableau
11             end
12         end
13     end
14 end
```

Table 5.5: (Example 5.2)An initial tableau.

unobserved link/all links	1	2	3	4	5	6	7	8	9
1	1	0	-1	0	-1	-1	0	0	-1
2	0	1	1	0	0	0	-1	0	1
4	0	0	0	1	0	-1	0	0	-1
8	0	0	0	0	0	0	-1	1	1

there are 11 candidate neighboring bases (i.e., number of non-zero values in columns $\{3, 5, 6, 7, 9\}$). The algorithm starts with the column labeled '3' and by (5.7), checks that by moving to neighboring basis $\{3, 2, 4, 8\}$, the network is not weakly observable, this is also the case for neighboring basis $\{1, 3, 4, 8\}$. The algorithm continues in this way until it reaches to the column labeled '7' where the neighboring basis$\{1, 7, 4, 8\}$ leads to weak observabilty of the network with the objective function value of 48. Since the algorithm found a better solution, it moves to the new basis and updates the tableau (the first improvement strategy).

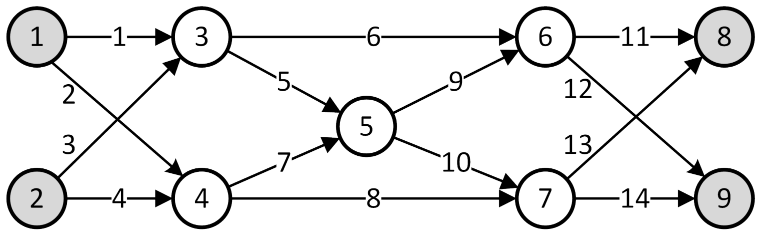


Figure 5.2: A network example.

5.5 Experimental results

In this section, we empirically evaluate the performance of our approach. We compare our algorithm with the approach taken by Xu et al. [130]. They implicitly/indirectly consider the propagation of measurement errors in the link flow inference; they particularly relate the error accumulation to the number of unobserved links connected to the non-centroid nodes of the network saying that the fewer unobserved links connected to the non-centroid nodes, the lower uncertainty in the inferred link flows. This is based on their experimental observation that for a node that has more than one unobserved link, we need to borrow from other flow conservation equations for the full observability of the link flows, resulting in higher uncertainty in the inferred link flows. They basically proposed two optimization models (referred to as min-max and min-sum models) to minimize the number of unobserved links connected to non-centroid nodes. In particular, min-max and min-sum models are binary integer linear programs which minimize the largest and the cumulative number of unobserved links connected to all non-centroid nodes, respectively. Both models are presented in Appendix A. Although the models can be solved to optimality using off-the-shelf solvers, in the case of directly considering the measurement errors, they may return unusable results (that is, sensor locations such that the resulting inferred link flows are not valid). We further explain this in the following example.

Example 5.3. We consider the parallel highway network in Figure 5.2 which was also considered by Xu et al. [130]. This network has 14 links and 9 nodes, where nodes $\{1, 2, 8, 9\}$ are centroid nodes. Both min-max and min-sum models select links $\{3, 4, 9, 11, 13\}$ as the unobserved links (see [130]). The largest number of unobserved links connected to non-centroid nodes is 2 (the objective function value of min-max model), corresponding to node 6 and links $\{9, 11\}$, while the cumulative number of unobserved links connected to each node is 6 (the objective function value of min-max model). Now suppose that interval link flows representing the measurement errors are given and reported in the second and third rows of Table 5.6. For the sake of illustration, let us assume that we also have the true link flows and they are presented in the last row of the table. Now let us compute interval flows of links $\{3, 4, 9, 11, 13\}$ by (5.4), i.e.,

$$\mathbf{f_3}^* = [-166, -45],\ \mathbf{f_4}^* = [89, 145],\ \mathbf{f_9}^* = [-29, 50],\ \mathbf{f_{11}}^* = [-68, 54],\ \mathbf{f_{13}}^* = [72, 180].$$

As can be seen, interval $\mathbf{f_3}^*$ does not have any positive realization. That is, for any value of the flows monitored by the sensors located on links $\{3, 4, 9, 11, 13\}$, the resulting inferred flows on the non-observed links are not valid. Thus, the solution returned by the two models is not meaningful. This is particularly the case when the interval flows do not contain the true flows. In this example, interval link flows of links $\{1, 3, 5, 6, 8, 13\}$ do not include the true flows. Moreover, the optimal solution of problem (5.6) for this example, by exploring all $U \in \mathcal{W}$, is $U = \{1, 8, 10, 12, 14\}$ with inferred interval flows of

$$\mathbf{f_1}^* = [62, 118],\ \mathbf{f_8}^* = [47, 101],\ \mathbf{f_{10}}^* = [33, 90],\ \mathbf{f_{12}}^* = [15, 70],\ \mathbf{f_{14}}^* = [18, 127],$$

and the objective function value of 331. It should be also noted that the largest number of unobserved links connected to the non-centroid nodes for this solution is 3 and the cumulative number of unobserved links connected to each non-centroid nodes is 7. That is, the set of sensor placements yielding the lowest variability in the inferred

Table 5.6: (Example 5.3)Interval link flows and true flows

links	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\overline{f}$	269	25	41	101	91	56	37	125	56	118	62	78	74	95
$\underline{f}$	192	17	29	67	64	39	25	89	38	78	42	52	52	63
true flow	160	21	24	84	114	70	31	74	47	98	52	65	93	79

link flows does not admit a good objective function value in min-max and min-sum models. This example reveals the need for directly considering the measurement errors in the link flow observability problem.

Remark 5.3. For the sake of having a meaningful comparison between our approach and that of Xu et al. [130], in our numerical study, we assume that the a priori interval link flows always include the true link flows; this ensures that the solutions by min-max and min-sum models remain meaningful (i.e., positive) when we explicitly considering the measurement errors.

5.5.1 Test networks and implementation

We applied our approach to solve the full link flow observability problem under measurement error for five different traffic networks , which were used as test networks in previously published works. We particularly consider the following networks:

- **Parallel network:** this network was used in [67, 100, 130] as a test network for the link flow observability problem. As explained above, this network has 14 links 5 non-centroid nodes and 4 centroid nodes.
- **Nguyen-Dupuis network:** This network was introduced by Nguyen and Dupuis [102] and later used in other papers as a test network, e.g., [17, 23, 24, 115]. It has 13 nodes and 38 links; we randomly set one of the node as a centroid node (to meet Assumption 5.1).

- **Sioux Falls network:** The traffic network of Sioux Falls in South Dakota, United States was studied in [56, 67, 101, 117, 130, 134], among others, and contains 24 nodes and 76 links. Similar to the Nguyen-Dupuis network, we removed one row from the node-link incidence matrix to meet Assumption 5.1.

- **Irvine network:** This network is a large traffic network in California, United States containing 162 nodes and 496 links, where 67 of the nodes are centroid. The network was used as a test network in many papers, including [31, 117, 130, 139].

- **Cuenca network:** This is another large traffic network widely used in the literature, see, for example, [19, 20, 22, 24, 25, 92]. The network has 232 nodes and 672 links; we considered 7 nodes as centroid nodes.

Given a traffic network, we generate the a priori link flows by solving the following linear program with a dummy objective function

$$\min\ 0^T f \ \text{ subject to } \ Af = 0,\ f \geq \ell,$$

where $\ell \in \mathbb{Z}^m$ is a randomly generated vector. Let f^t be a solution to the above program. For the purpose of our experimental study, we assumed that vector f^t is the vector of the true link flows of the traffic network, and we composed the interval flows as $\mathbf{f} = [0.8f^t, 1.2f^t]$. Hence, as outlined in the next section, we can compare the results of our approach and those obtained by the min-max and min-sum models against the true link flows. In our test networks, we generated the all-integer vector ℓ in $[50, 100]$ for the parallel network, in $[100, 500]$ for the Sioux Falls network, and in $[500, 1000]$ for Nguyen-Dupuis, Irvine and Cuenca networks.

We set the stopping condition for our local search algorithm to 15 minutes of running time or trying 3 different initial tableaux, whichever reaches first. We implemented our algorithm using MATLAB (2019b), and we used CPLEX 12.9 to solve min-max and min-sum models. We carried out our experiments on a computer with an Intel (R) Core (TM) i7-4790 CPU processor at 3.60 GHZ with 32.00 GB of RAM.

5.5.2 Analysis of results

In this section, we discuss results of our computational experiments. Let us recall that we compare the results obtained by our local search algorithm against those of min-max and min-sum models. We applied the methods on five test networks, and we divide the results into two parts. In the first part, we evaluate the results according to the error accumulation (see (5.5)) and the running times. For the second part, we discuss the variability of the solutions using different metrics. The optimal value of (5.6) is not known; hence, we benchmarked the results of min-max and min-sum models to evaluate our algorithm. Particularly, given the error accumulation by our local search algorithm (f^{ls}), we computed its relative gap from the error accumulation (f^m) yielded by either min-max or min-sum models (by applying (5.5)) as

$$\text{gap} := \frac{f^m - f^{ls}}{f^m}.$$

Thus, a higher gap measure indicates a better performance of our local search algorithm compared to min-max and min-sum models. In Table 5.7, the first three columns report the networks and their specifications. The following two columns report the relative gap of the results of the local search algorithm from those of min-max and min-sum models. Columns 6-8 show the running times, and the last column denotes the average number of explored basis by the local search algorithm over all iterations. As it was expected, the min-max and min-sum models can be solved very fast using the CPLEX solver; for all the networks, the running time of the solver is less than one second. On the other hand, the local search algorithm takes longer time to converge, and for the Irvine and Cuenca networks, the algorithm reaches to the time limit (15 minutes). Regarding the gap metric, the local search outperforms min-max and min-sum models on all the networks with the gap ranges between 0.03 and 0.90.

Table 5.7: Results related to gap and running times

network	# nodes	# links	gap		running time (sec)			avg. # bases
			min-sum	min-max	LS	min-sum	min-max	
Parallel	9	14	0.03	0.11	0.15	0.02	0.02	49.33
Ng.-Dep.	13	38	0.33	0.39	1.13	0.00	0.03	269
S. Falls	24	76	0.90	0.88	18.58	0.02	0.00	2027
Irvine	229	496	0.22	0.21	900	0.00	0.02	7622
Cuenca	232	672	0.26	0.17	900	0.02	0.02	4249

Now that we have the solutions, we can compare the results according to different measures of variability. We here define a metric which measures the non-negative portion of inferred interval link flows, that is,

$$\text{ratio} := \frac{1}{n}\sum_{i\in N^*}\frac{w^+(\mathbf{f}^*_{U(i)})}{w(\mathbf{f}^*_{U(i)})},$$

where $w(.)$ denotes the interval width (i.e., $(\overline{f}^*_{U(i)} - \underline{f}^*_{U(i)})$), and $w^+(.)$ computes the non-negative portion of the interval width. Thus, a higher ratio shows that a larger part of the inferred interval link flows lies on the positive orthant, meaning that it is less likely to get negative flows. As explained earlier, for the sake of evaluating the quality of the results, we assumed that we have the knowledge of the true link flows. Hence, we can compute the overestimated portion of the inferred interval link flows by

$$\text{plus} := \frac{1}{n}\sum_{i\in N^*}\frac{\sigma^+_{U(i)}}{f^t_{U(i)}},$$

where $\sigma^+_{U(i)} = (\overline{f}^*_{U(i)} - f^t_{U(i)})$. Similarly, we can determine the underestimated portion

Table 5.8: Results related to the variability metrics

network	ratio			plus/minus		
	LS	min-sum	min-max	LS	min-sum	min-max
Parallel	1.000	1.000	0.951	0.448	0.492	0.798
Ng.-Dep.	0.993	0.873	0.874	0.686	1.338	1.317
S. Falls	0.971	0.820	0.822	0.722	1.968	1.930
Irvine	0.922	0.879	0.872	1.132	1.508	1.544
Cuenca	0.863	0.835	0.843	1.767	2.316	2.092

of the inferred interval link flows as

$$\text{minus} := \frac{1}{n} \sum_{i \in N^*} \frac{\sigma^-_{U(i)}}{f^t_{U(i)}},$$

where $\sigma^-_{U(i)} = (f^t_{U(i)} - \underline{f}^*_{U(i)})$. Thus, the lower the plus/minus, the better the estimation. Table 5.8 summarizes the results of the above metrics; each grouped columns represent one of the metric. Note that the plus and minus metrics are similar in our experiments because of the way we generated the interval link flows (i.e., the true link flows are always the center of the interval flows), and thus we report them together in the table. The inferred interval link flows by the local search algorithm admit a bigger portion of positive flows than those by the min-max and min-sum models. As can be noted, the ratio tends to drop as the size of the network increases. This is also the case with the plus/minus metric where the local search outperforms the two models. The outperformance is more significant for larger size traffic networks.

5.6 Concluding remarks

The full link flow observability problem is the problem of locating counting sensors on a subset of the links of a given traffic network to fully observe all the link flows of the network. There are few studies in the literature proposing a minimum cardinality for such a subset. However, this subset of minimum cardinality need not be unique; basically, there could be several subsets with a same minimum number of links allowing full observability of the network. In the case of error-free data, any of those subsets can be used to uniquely observe all the link flows. However, in this paper, we showed that failing to take into account the sensor measurement error may result in getting useless information (i.e., negative link flows). This is due to the fact that the measurement errors of the links equipped with sensors are accumulated and propagated to the unobserved links through the inference process, resulting in having negative link flows. In this paper, we proposed an optimization approach to address this problem. Particularly, we assumed that we have a priori interval link flows and presented an optimization problem which minimizes the variability of the inferred interval link flows. The feasible set of our optimization problem is the set of all possible sets of sensor placements with minimum cardinality for which the inferred interval link flows admit positive realizations. This ensures that we do not get all-negative link flows; however, this feasible set cannot be described explicitly in general. Thus, we developed a local search algorithm which with the help of a simplex-like tableau, explores the feasible set without explicitly constructing it to find a set of sensor placements with minimum variability in inferred link flows. A future direction of this work can be an extension of our approach to address the measurement errors in a more general form of the flow-observability problem, allowing any type of sensors, any type of a priori information, any type of flows of interest, and any type of placements for sensors (nodes or links).

CHAPTER 6

SUMMARY AND FUTURE DIRECTIONS

In this dissertation, we considered linear programming problems where input data can vary in some given real compact intervals. We focused on two main problems in this context: the optimal value range problem and the optimal solution set problem. Given an interval linear program, the former is the problem of finding the best and the worst optimal values over all the realizations of the interval data, while the latter determines the set of all possible optimal solutions considering all the realizations of the interval data. The goal of this dissertation was to contribute to the computational and applications aspects of the two problems and of the interval optimization literature in general.

In Chapter 2, we revisited the optimal value range problem for equality-constrained interval linear programs for which computing the worst optimal value is known to be NP-hard. We developed three heuristics to solve the problem where one of the methods returns a lower bound to the worst optimal value, while the other two methods compute an upper bound of the worst optimal value. Through numerical experiments, we showed that our greedy approach of getting a lower bound is promising in computing a cheap but tight bound. However, our enclosure-based methods of finding an upper bound may not return a reliable bound; hence, there is an opportunity for improvement.

Our work in Chapter 3 was motivated by a real-world problem, the healthcare access measurement problem, where access to healthcare services is evaluated using functions of the optimal matching of patients and providers, obtained by solving a linear program. The question which intrigued our research was: *How to evaluate access to healthcare services when input parameters of the linear program matching patients and providers are uncertain?* To answer the question, we introduced a new problem in the context of interval linear programming – *the outcome range problem* – where the goal is quantifying unintended/further consequences of optimal decisions made under uncertainty, modeled by an *extra* linear function (referred to as an outcome function). Specifically, the outcome range problem finds the upper and lower bounds of a function (other than the objective function) over all possible optimal solutions of an interval linear program. We then focused on programs with interval right-hand sides and studied the computational complexity of the problem for this case, showing the problem is Np-hard even for this particular case. We also addressed some of its theoretical properties aimed at characterizing the optimal scenarios. Given the complexity of the problem, we designed and numerically tested two heuristics to approximate the range of an outcome function. Finally, we show the applicability of our approach in the real-world context by using the outcome range problem to evaluate access to the primary care service for children in the State of Mississippi, United States.

In Chapter 4, we extended our work on the outcome range problem. We again focused on the case where the uncertainty of the underlying linear program occurred only in the right-hand side vector and studied the computational complexity of checking the optimality of a certain solution or scenario to the outcome range problem. We also investigated how our problem stands in a broader optimization context with respect to well-known classes of optimization problems, including bilevel optimization, multiobjective optimization, and the optimal value range problem. We developed three new heuristics to solve the problem and numerically tested the algorithms on three different datasets. Our results indicated that our two methods of computing an inner approximation of the range of an outcome function are promising in computing good quality solutions with low running times. In contrast, our method of finding an outer approximation of the range of an outcome function did not return tight approximations.

In Chapter 5, we studied an application of interval optimization in the context of traffic management. We particularly considered the full link flow observability problem, which is the problem of locating the minimum number of sensors on a subset of links of a traffic network to observe traffic flows on all links in the network, and we investigated the case where the data from sensors contain errors. Our work in this chapter is the first attempt to explicitly address the measurement errors in the full link flow observability problem. We motivated the need to explicitly considering the measurement errors and proposed an optimization approach to handle them. Our optimization approach describes the measurement errors in the observed link flows by real compact intervals and seeks a set of sensor placements minimizing the variability in the inferred link flows. We presented a local search algorithm to solve our optimization problem. The results on several real traffic networks demonstrated that our approach is promising in selecting a set of sensor placements with low error accumulation in the inferred link flows.

We summarize some possible future directions of our work as follows:

- Computing a tight upper bound on the worst optimal value of the equality-constrained interval linear program can be of interest. Having a tight upper bound is very useful in designing an exact method for the problem.

- One may consider a more general uncertainty set and address the outcome range problem in this context. For example, we can describe the uncertainty set as a general convex polyhedron, or in the case of interval right-hand sides, we can describe each right-hand side entry as a linear combination of interval parameters $\mathbf{b}_1, \ldots, \mathbf{b}_k$ (i.e., $B\mathbf{b}$).

- Computing a tight outer approximation of the range of an outcome function seems challenging. Hence, it would be interesting to work on a method that returns a tight outer approximation efficiently.

- In this dissertation, we studied the outcome range problem where the interval uncertainty occurs only in the right-hand sides of the underlying linear program. A future direction could be addressing the problem where the interval uncertainty occurs in technological coefficients (matrix A) or objective function coefficients (vector c).

- In Chapter 5, we addressed the measurement errors in the full link flow observability problem. One can generalize the approach to address measurement errors in a more general form of the flow observability problem.

REFERENCES

[1] Allahdadi, M., and Golestane, A. K. Monte carlo simulation for computing the worst value of the objective function in the interval linear programming. International Journal of Applied and Computational Mathematics 2, 4 (2016), 509–518. https://doi.org/10.1007/s40819-015-0074-2.

[2] Allahdadi, M., and Nehi, H. M. The optimal solution set of the interval linear programming problems. Optimization Letters 7, 8 (2013), 1893–1911. https://doi.org/10.1007/s11590-012-0530-4.

[3] Atamtürk, A., and Zhang, M. Two-stage robust network flow and design under demand uncertainty. Operations Research 55, 4 (2007), 662–673. https://doi.org/10.1287/opre.1070.0428.

[4] Bandi, C., and Bertsimas, D. Tractable stochastic analysis in high dimensions via robust optimization. Mathematical programming 134, 1 (2012), 23–70. https://doi.org/10.1007/s10107-012-0567-2.

[5] Beeck, H. Linear programming with inexact data. Technical Report TUM-ISU-7830, Technical University of Munich, Munich (1978).

[6] Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. Robust optimization, vol. 28. Princeton University Press, 2009. https://www2.isye.gatech.edu/~nemirovs/FullBookDec11.pdf.

[7] Ben-Tal, A., and Nemirovski, A. Robust solutions of linear programming problems contaminated with uncertain data. Mathematical programming 88, 3 (2000), 411–424. `https://doi.org/10.1007/PL00011380`.

[8] Bertsekas, D. P. Network optimization: continuous and discrete models. Athena Scientific Belmont, MA, 1998.

[9] Bertsimas, D., and Goyal, V. On the power and limitations of affine policies in two-stage adaptive optimization. Mathematical Programming 134, 2 (2012), 491–531. `https://doi.org/10.1007/s10107-011-0444-4`.

[10] Bertsimas, D., and Sim, M. The price of robustness. Operations Research 52, 1 (2004), 35–53. `https://doi.org/10.1287/opre.1030.0065`.

[11] Bianco, L., Cerrone, C., Cerulli, R., and Gentili, M. Locating sensors to observe network arc flows: exact and heuristic approaches. Computers & Operations Research 46 (2014), 12–22. `https://doi.org/10.1016/j.cor.2013.12.013`.

[12] Bianco, L., Confessore, G., and Gentili, M. Combinatorial aspects of the sensor location problem. Annals of Operations Research 144, 1 (2006), 201–234. `https://doi.org/10.1007/s10479-006-0016-9`.

[13] Bianco, L., Confessore, G., and Reverberi, P. A network based model for traffic sensor location with implications on o/d matrix estimates. Transportation Science 35, 1 (2001), 50–60. `https://doi.org/10.1287/trsc.35.1.50.10140`.

[14] Birge, J. R., and Louveaux, F. Introduction to stochastic programming. Springer Science & Business Media, 2011.

[15] Bitran, G. R. Linear multiple objective problems with interval coefficients. Management Science 26 (1980), 694–706.

[16] Boland, N., Charkhgard, H., and Savelsbergh, M. A new method for optimizing a linear function over the efficient set of a multiobjective integer program. European Journal of Operational Research 260, 3 (2017), 904–919. https://doi.org/10.1016/j.ejor.2016.02.037.

[17] Castillo, E., Calviño, A., Lo, H. K., Menéndez, J. M., and Grande, Z. Non-planar hole-generated networks and link flow observability based on link counters. Transportation Research Part B: Methodological 68 (2014), 239–261. https://doi.org/10.1016/j.trb.2014.06.015.

[18] Castillo, E., Calvino, A., Menendez, J. M., Jimenez, P., and Rivas, A. Deriving the upper bound of the number of sensors required to know all link flows in a traffic network. IEEE Transactions on Intelligent Transportation Systems 14, 2 (2013), 761–771. https://doi.org/10.1109/TITS.2012.2233474.

[19] Castillo, E., Gallego, I., Menéndez, J. M., and Jimenez, P. Link flow estimation in traffic networks on the basis of link flow observations. Journal of Intelligent Transportation Systems 15, 4 (2011), 205–222. https://doi.org/10.1080/15472450.2011.620487.

[20] Castillo, E., Gallego, I., Menéndez, J. M., and Rivas, A. Optimal use of plate-scanning resources for route flow estimation in traffic networks. IEEE Transactions on Intelligent Transportation Systems 11, 2 (2010), 380–391. https://doi.org/10.1109/TITS.2010.2042958.

[21] Castillo, E., Gallego, I., Sanchez-Cambronero, S., and Rivas, A. Matrix tools for general observability analysis in traffic networks. IEEE

Transactions on Intelligent Transportation Systems 11, 4 (2010), 799–813. https://doi.org/10.1109/TITS.2010.2050768.

[22] Castillo, E., Jiménez, P., Menéndez, J. M., Rivas, A., and Gallego, I. A ternary-arithmetic topological based algebraic method for networks traffic observability. Applied Mathematical Modelling 35, 11 (2011), 5338–5354. https://doi.org/10.1016/j.apm.2011.04.044.

[23] Castillo, E., Menéndez, J. M., and Jiménez, P. Trip matrix and path flow reconstruction and estimation based on plate scanning and link observations. Transportation Research Part B: Methodological 42, 5 (2008), 455–481. https://doi.org/10.1016/j.trb.2007.09.004.

[24] Castillo, E., Nogal, M., Rivas, A., and Sánchez-Cambronero, S. Observability of traffic networks: optimal location of counting and scanning devices. Transportmetrica B: Transport Dynamics 1, 1 (2013), 68–102. https://doi.org/10.1080/21680566.2013.780987.

[25] Castillo, E., Rivas, A., Jiménez, P., and Menéndez, J. M. Observability in traffic networks. plate scanning added by counting information. Transportation 39, 6 (2012), 1301–1333. https://doi.org/10.1007/s11116-012-9390-0.

[26] Cerrone, C., Cerulli, R., and Gentili, M. Vehicle-id sensor location for route flow recognition: Models and algorithms. European Journal of Operational Research 247, 2 (2015), 618–629. https://doi.org/10.1016/j.ejor.2015.05.070.

[27] Cerulli, R., D'Ambrosio, C., and Gentili, M. Best and worst values of the optimal cost of the interval transportation problem. In International

Conference on Optimization and Decision Science (2017), Springer, pp. 367–374. https://doi.org/10.1007/978-3-319-67308-0_37.

[28] Cheng, G., Huang, G., and Dong, C. Convex contractive interval linear programming for resources and environmental systems management. Stochastic Environmental Research and Risk Assessment 31, 1 (2017), 205–224. https://doi.org/10.1007/s00477-015-1187-1.

[29] Cheng, G., Huang, G., Dong, C., Baetz, B., and Li, Y. Interval recourse linear programming for resources and environmental systems management under uncertainty. Journal of Environmental Informatics 30, 2 (2017). https://doi:10.3808/jei.201500312.

[30] Chinneck, J., and Ramadan, K. Linear programming with interval coefficients. Journal of the Operational Research Society 51, 2 (2000), 209–220. https://doi.org/10.1057/palgrave.jors.2600891.

[31] Chootinan, P., Chen, A., and Recker, W. Improved path flow estimator for origin–destination trip tables. Transportation Research Record 1923, 1 (2005), 9–17. https://doi.org/10.1177%2F0361198105192300102.

[32] Cipriani, E., Fusco, G., Gori, S., and Petrelli, M. Heuristic methods for the optimal location of road traffic monitoring. In 2006 IEEE Intelligent Transportation Systems Conference (2006), IEEE, pp. 1072–1077. https://doi.org/10.1109/ITSC.2006.1707364.

[33] Confessore, G., Dell'Olmo, P., and Gentili, M. Experimental evaluation of approximation and heuristic algorithms for the dominating paths problem. Computers & Operations Research 32, 9 (2005), 2383–2405. https://doi.org/10.1016/j.cor.2004.03.008.

[34] Curry, S. Statistical inference for optimization models: Sensitivity analysis and uncertainty quantification, 2019. Ph.D. Thesis, Georgia Institute of Technology, School of Industrial and Systems Engineering, Atlanta, US. http://hdl.handle.net/1853/62265.

[35] Dantzig, G. B. Linear programming under uncertainty. Management science 1, 3-4 (1955), 197–206. https://doi.org/10.1287/mnsc.1040.0261.

[36] D'Ambrosio, C., Gentili, M., and Cerulli, R. The optimal value range problem for the interval (immune) transportation problem. Omega 95 (2020), 102059. https://doi.org/10.1016/j.omega.2019.04.002.

[37] Ehlert, A., Bell, M. G., and Grosso, S. The optimisation of traffic count locations in road networks. Transportation Research Part B: Methodological 40, 6 (2006), 460–479. https://doi.org/10.1016/j.trb.2005.06.001.

[38] Fortin, J., Zieliński, P., Dubois, D., and Fargier, H. Criticality analysis of activity networks under interval uncertainty. Journal of Scheduling 13, 6 (2010), 609–627. https://doi.org/10.1007/s10951-010-0163-3.

[39] Fu, C., Zhu, N., Ling, S., Ma, S., and Huang, Y. Heterogeneous sensor location model for path reconstruction. Transportation Research Part B: Methodological 91 (2016), 77–97. https://doi.org/10.1016/j.trb.2016.04.013.

[40] Gabrel, V., Murat, C., and Remli, N. Linear programming with interval right hand sides. International Transactions in Operational Research 17, 3 (2010), 397–408. https://doi.org/10.1111/j.1475-3995.2009.00737.x.

[41] Gal, T. Linear parametric programming—a brief survey. In Sensitivity, Stability and Parametric Analysis. Springer, 1984, pp. 43–68. https://doi.org/10.1007/BFb0121210.

[42] Gal, T. Postoptimal Analyses, Parametric Programming, and Related Topics: degeneracy, multicriteria decision making, redundancy. Walter de Gruyter, 2010.

[43] Garajová, E. The optimal solution set of interval linear programming problems. Master's thesis, Charles Univesity, Faculty of Mathematics and Physics, Prague, Czech Republic. `https://is.cuni.cz/webapps/zzp/detail/168259/?lang=en`, 2016.

[44] Garajová, E., and Hladík, M. On the optimal solution set in interval linear programming. Computational Optimization and Applications 72, 1 (2019), 269–292. `https://doi.org/10.1007/s11590-012-0530-4`.

[45] Garajová, E., Hladík, M., and Rada, M. On the properties of interval linear programs with a fixed coefficient matrix. In International Conference on Optimization and Decision Science (2017), Springer, pp. 393–401. `https://doi.org/10.1007/978-3-319-67308-0_40`.

[46] Garajová, E., Hladík, M., and Rada, M. The best, the worst and the semi-strong: optimal values in interval linear programming. Croatian Operational Research Review (2019), 201–209. `https://doi.org/10.17535/crorr.2019.0018`.

[47] Garajová, E., Hladík, M., and Rada, M. Interval linear programming under transformations: optimal solutions and optimal value range. Central European Journal of Operations Research 27, 3 (2019), 601–614. `https://doi.org/10.1007/s10100-018-0580-5`.

[48] Gentili, M., Harati, P., and Serban, N. Projecting the impact of the affordable care act provisions on accessibility and availability of primary care

providers for the adult population in georgia. American Journal of Public Health 106, 8 (2016), 1470–1476. https://doi.org/10.2105/AJPH.2016.303222.

[49] Gentili, M., Harati, P., Serban, N., O'connor, J., and Swann, J. Quantifying disparities in accessibility and availability of pediatric primary care across multiple states with implications for targeted interventions. Health Services Research 53, 3 (2018), 1458–1477. https://doi.org/10.1111/1475-6773.12722.

[50] Gentili, M., Isett, K., Serban, N., and Swann, J. Small-area estimation of spatial access to care and its implications for policy. Journal of Urban Health 92, 5 (2015), 864–909. https://doi.org/10.1007/s11524-015-9972-1.

[51] Gentili, M., and Mirchandani, P. B. Locating active sensors on traffic networks. Annals of Operations Research 136, 1 (2005), 229–257. https://doi.org/10.1007/s10479-005-2047-z.

[52] Gentili, M., and Mirchandani, P. B. Locating sensors on traffic networks: Models, challenges and research opportunities. Transportation Research Part C: Emerging Technologies 24 (2012), 227–255. https://doi.org/10.1016/j.trc.2012.01.004.

[53] Gerlach, W. Zur lösung linearer ungleichungssysteme bei störimg der rechten seite und der koeffizientenmatrix. Mathematische Operationsforschung und Statistik. Series Optimization 12, 1 (1981), 41–43. https://doi.org/10.1080/02331938108842705.

[54] Hadavi, M., and Shafahi, Y. Vehicle identification sensor models for origin–destination estimation. Transportation Research Part B: Methodological 89 (2016), 82–106. https://doi.org/10.1016/j.trb.2016.03.011.

[55] Hansen, P., Jaumard, B., and Savard, G. New branch-and-bound rules for linear bilevel programming. SIAM Journal on Scientific and Statistical Computing 13, 5 (1992), 1194–1217. https://doi.org/10.1137/0913069.

[56] He, S.-x. A graphical approach to identify sensor locations for link flow inference. Transportation Research Part B: Methodological 51 (2013), 65–76. https://doi.org/10.1016/j.trb.2013.02.006.

[57] Hladík, M. Optimal value range in interval linear programming. Fuzzy Optimization and Decision Making 8, 3 (2009), 283–294. https://doi.org/10.1007/s10700-009-9060-7.

[58] Hladík, M. Interval linear programming: A survey. In In Chapter 2. In: Mann ZA (ed) Linear programming—new frontiers in theory and applications. Nova Science Publishers, New York, 2012, pp. 85–120.

[59] Hladík, M. An interval linear programming contractor. In: J. Ramik and D. Stavarek (eds.), Proceedings 30th Int. Conf. Mathematical Methods in Economics 2012, Karvina, Czech Republic (2012), 284–289. Silesian University in Opava.

[60] Hladík, M. Weak and strong solvability of interval linear systems of equations and inequalities. Linear Algebra and its Applications 438, 11 (2013), 4156–4165. https://doi.org/10.1016/j.laa.2013.02.012.

[61] Hladík, M. How to determine basis stability in interval linear programming. Optimization Letters 8, 1 (2014), 375–389. https://doi.org/10.1007/s11590-012-0589-y.

[62] Hladík, M. On approximation of the best case optimal value in interval linear programming. Optimization Letters 8, 7 (2014), 1985–1997. https://doi.org/10.1007/s11590-013-0715-5.

[63] Hladík, M. Transformations of interval linear systems of equations and inequalities. Linear and Multilinear Algebra 65, 2 (2017), 211–223. https://doi.org/10.1080/03081087.2016.1180339.

[64] Hladík, M. The worst case finite optimal value in interval linear programming. Croatian Operational Research Review 9, 2 (2018), 245–254. https://doi.org/10.17535/crorr.2018.0019.

[65] Hladík, M. Two approaches to inner estimations of the optimal solution set in interval linear programming. In Proceedings of the 2020 4th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence (New York, NY, USA, 2020), ISMSI '20, Association for Computing Machinery, p. 99–104. https://doi.org/10.1145/3396474.3396479.

[66] Hu, S.-R., and Liou, H.-T. A generalized sensor location model for the estimation of network origin–destination matrices. Transportation Research Part C: Emerging Technologies 40 (2014), 93–110. https://doi.org/10.1016/j.trc.2014.01.004.

[67] Hu, S.-R., Peeta, S., and Chu, C.-H. Identification of vehicle sensor locations for link-based network traffic applications. Transportation Research Part B: Methodological 43, 8-9 (2009), 873–894. https://doi.org/10.1016/j.trb.2009.02.008.

[68] Hu, S.-R., Peeta, S., and Liou, H.-T. Integrated determination of network origin–destination trip matrix and heterogeneous sensor selection and location strategy. IEEE Transactions on Intelligent Transportation Systems 17, 1 (2015), 195–205. https://doi.org/10.1109/TITS.2015.2473691.

[69] Huang, G. H., Baetz, B. W., and Patry, G. G. Grey integer programming: an application to waste management planning under uncertainty.

European Journal of Operational Research 83, 3 (1995), 594–620. `https://doi.org/10.1016/0377-2217(94)00093-R`.

[70] Infanger, G. Planning under uncertainty solving large-scale stochastic linear programs. Tech. rep., Stanford Univ., CA (United States). Systems Optimization Lab., 1992.

[71] Inuiguchi, M., Ichihashi, H., and Tanaka, H. Fuzzy programming: a survey of recent developments. In Stochastic versus fuzzy approaches to multiobjective mathematical programming under uncertainty. Springer, 1990, pp. 45–68. `https://doi.org/10.1007/978-94-009-2111-5_4`.

[72] Inuiguchi, M., and Sakawa, M. Possible and necessary efficiency in possibilistic multiobjective linear programming problems and possible efficiency test. Fuzzy Sets and Systems 78, 2 (1996), 231–241.

[73] Jansson, C. Rigorous lower and upper bounds in linear programming. SIAM Journal on Optimization 14, 3 (2004), 914–935. `https://doi.org/10.1137/S1052623402416839`.

[74] Jansson, C., and Rump, S. M. Rigorous solution of linear programming problems with uncertain data. Zeitschrift für Operations Research 35, 2 (1991), 87–111. `https://doi.org/10.1007/BF02331571`.

[75] Juman, Z., and Hoque, M. A heuristic solution technique to attain the minimal total cost bounds of transporting a homogeneous product with varying demands and supplies. European Journal of Operational Research 239, 1 (2014), 146–156. `https://doi.org/10.1016/j.ejor.2014.05.004`.

[76] Kall, P., Wallace, S. W., and Kall, P. Stochastic programming. Springer, 1994.

[77] Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R. E., Danna, E., Gamrath, G., Gleixner, A. M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D. E., and Wolter, K. MIPLIB 2010. Mathematical Programming Computation 3, 2 (2011), 103–163. http://mpc.zib.de/index.php/MPC/article/view/56/28.

[78] Kumar, P., Panda, G., and Gupta, U. An interval linear programming approach for portfolio selection model. International Journal of Operational Research 27, 1-2 (2016), 149–164. https://doi.org/10.1504/IJOR.2016.078463.

[79] Lai, K. K., Wang, S., Xu, J., Zhu, S., and Fang, Y. A class of linear interval programming problems and its application to portfolio selection. IEEE Transactions on Fuzzy Systems 10, 6 (2002), 698–704. https://doi.org/10.1109/TFUZZ.2002.805902.

[80] Larsson, T., Lundgren, J. T., and Peterson, A. Allocation of link flow detectors for origin-destination matrix estimation—a comparative study. Computer-Aided Civil and Infrastructure Engineering 25, 2 (2010), 116–131. https://doi.org/10.1111/j.1467-8667.2009.00625.x.

[81] Lee, I., Curry, S., and Serban, N. Solving large batches of linear programs. INFORMS Journal on Computing 31, 2 (2019), 302–317. https://doi.org/10.1287/ijoc.2018.0838.

[82] Li, D. Interval-valued matrix games. In Linear Programming Models and Methods of Matrix Games with Payoffs of Triangular Fuzzy Numbers. Springer, 2016, pp. 3–63. https://doi.org/10.1007/978-3-662-48476-0.

[83] Li, W., Luo, J., Wang, Q., and Li, Y. Checking weak optimality of the solution to linear programming with interval right-hand side. Optimization Letters 8, 4 (2014), 1287–1299. https://doi.org/10.1007/s11590-013-0654-1.

[84] Li, Y., Huang, G. H., Guo, P., Yang, Z., and Nie, S.-L. A dual-interval vertex analysis method and its application to environmental decision making under uncertainty. European Journal of Operational Research 200, 2 (2010), 536–550. https://doi.org/10.1016/j.ejor.2009.01.013.

[85] Liu, S.-T., and Kao, C. Matrix games with interval data. Computers and Industrial Engineering 56, 4 (2009), 1697–1700. https://doi.org/10.1016/j.cie.2008.06.002.

[86] Lodwick, W. A., and Kacprzyk, J. Fuzzy optimization: Recent advances and applications, vol. 254. Springer, 2010.

[87] Löfberg, J. Yalmip : A toolbox for modeling and optimization in matlab. In In Proceedings of the CACSD Conference (Taipei, Taiwan, 2004). https://doi.org/10.1109/CACSD.2004.1393890.

[88] Lovász, L. A new linear programming algorithm—better or worse than the simplex method? The Mathematical Intelligencer 2, 3 (1980), 141–146. https://doi.org/10.1007/BF03023055.

[89] Luo, H., Ding, X., Peng, J., Jiang, R., and Li, D. Complexity results and effective algorithms for worst-case linear optimization under uncertainties. INFORMS Journal on Computing (2020). https://doi.org/10.1287/ijoc.2019.0941.

[90] McCormick, G. P. Computability of global solutions to factorable nonconvex programs: Part i — convex underestimating problems. Mathematical Programming 10, 1 (1976), 147–175. https://doi.org/10.1007/BF01580665.

[91] Miele, A., Pritchard, R. E., and Damoulakis, J. Sequential gradient-restoration algorithm for optimal control problems. Journal of Optimization Theory and Applications 5, 4 (1970), 235–282. https://doi.org/10.1007/BF00927913.

[92] Mínguez, R., Sánchez-Cambronero, S., Castillo, E., and Jiménez, P. Optimal traffic plate scanning location for od trip matrix and route estimation in road networks. Transportation Research Part B: Methodological 44, 2 (2010), 282–298. https://doi.org/10.1016/j.trb.2009.07.008.

[93] Mohammadi, M., and Gentili, M. Bounds on the worst optimal value in interval linear programming. Soft Computing 23, 21 (2019), 11055–11061. https://doi.org/10.1007/s00500-018-3658-z.

[94] Mohammadi, M., and Gentili, M. The outcome range problem in interval linear programming. Computers & Operations Research 129 (2021), 105160. https://doi.org/10.1016/j.cor.2020.105160.

[95] Mohammadi, M., Gentili, M., Hladík, M., and Cerulli, R. How to quantify outcome functions of interval-valued linear programs. Under revision for *INFORMS Journal on Computing*.

[96] Morrison, D. R., and Martonosi, S. E. Characteristics of optimal solutions to the sensor location problem. Annals of Operations Research 226, 1 (2015), 463–478. https://doi.org/10.1007/s10479-014-1638-y.

[97] Mraz, F. Calculating the exact bounds of optimal values in LP with interval coefficients. Annals of Operations Research 81 (1998), 51–62. https://doi.org/10.1023/A:1018985914065.

[98] Nemhauser, G. L., and Wolsey, L. A. Integer and Combinatorial Optimization. Wiley-Interscience, USA, 1988.

[99] Neumaier, A. A simple derivation of the hansen-bliek-rohn-ning-kearfott enclosure for linear interval equations. Reliable Computing 5, 2 (1999), 131–136. https://doi.org/10.1023/A:1009997221089.

[100] Ng, M. Synergistic sensor location for link flow inference without path enumeration: A node-based approach. Transportation Research Part B: Methodological 46, 6 (2012), 781–788. https://doi.org/10.1016/j.trb.2012.02.001.

[101] Ng, M. Partial link flow observability in the presence of initial sensors: Solution without path enumeration. Transportation Research Part E: Logistics and Transportation Review 51 (2013), 62–66. https://doi.org/10.1016/j.tre.2012.12.002.

[102] Nguyen, S., and Dupuis, C. An efficient method for computing traffic equilibria in networks with asymmetric transportation costs. Transportation Science 18, 2 (1984), 185–202. https://doi.org/10.1287/trsc.18.2.185.

[103] Nobles, M., Serban, N., and Swann, J. Spatial accessibility of pediatric primary healthcare: measurement and inference. The Annals of Applied Statistics 8, 4 (2014), 1922–1946. https://doi.org/10.1214/14-AOAS728.

[104] Novotná, J., Hladík, M., and Masařík, T. Duality gap in interval linear programming. Journal of Optimization Theory and Applications 184, 2 (2020), 565–580. https://doi.org/10.1007/s10957-019-01610-y.

[105] Oettli, W., and Prager, W. Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides. Numerische Mathematik 6, 1 (1964), 405–409. https://doi.org/10.1007/BF01386090.

[106] Park, Y. S., Lim, S. H., Egilmez, G., and Szmerekovsky, J. Environmental efficiency assessment of u.s. transport sector: A slack-based data

envelopment analysis approach. Transportation Research Part D: Transport and Environment 61 (2018), 152 – 164. `https://doi.org/10.1016/j.trd.2016.09.009`.

[107] Peng, J., and Zhu, T. A nonlinear semidefinite optimization relaxation for the worst-case linear optimization under uncertainties. Mathematical Programming 152, 1-2 (2015), 593–614. `https://doi.org/10.1007/s10107-014-0799-4`.

[108] Prékopa, A. Stochastic programming, vol. 324. Springer Science & Business Media, 2013.

[109] Ramik, J. Fuzzy linear optimization. In Linear Optimization Problems with Inexact Data. Springer, 2006, pp. 117–164. `https://doi.org/10.1007/0-387-32698-7_5`.

[110] Rohn, J. Proofs to "Solving interval linear systems". Freiburger Intervall-Berichte 84/7, Albert-Ludwigs-Universität, Freiburg, 1984.

[111] Rohn, J. Interval linear programming. In Linear Optimization Problems with Inexact Data, M. Fiedler et al., Ed. Springer, 2006, pp. 79–100. `https://doi.org/10.1007/0-387-32698-7_3`.

[112] Rohn, J. Solvability of systems of interval linear equations and inequalities. In Linear Optimization Problems with Inexact Data. Springer, 2006, pp. 35–77. `https://doi.org/10.1007/0-387-32698-7_2`.

[113] Rohn, J. A handbook of results on interval linear problems, 2012. Technical Report 1163. Institute of Computer Science. Academy of Sciences of the Czech Republic. Prague. `http://www.nsc.ru/interval/Library/Surveys/ILinProblems.pdf`.

[114] Roos, E., and den Hertog, D. Reducing conservatism in robust optimization. INFORMS Journal on Computing (2020). https://doi.org/10.1287/ijoc.2019.0913.

[115] Rubin, P., and Gentili, M. An exact method for locating counting sensors in flow observability problems. Transportation Research Part C: Emerging Technologies 123 (2021), 102855. https://doi.org/10.1016/j.trc.2020.102855.

[116] Rump, S. INTLAB - INTerval LABoratory. In Developments in Reliable Computing, T. Csendes, Ed. Kluwer Academic Publishers, Dordrecht, 1999, pp. 77–104. http://www.ti3.tuhh.de/rump/.

[117] Salari, M., Kattan, L., Lam, W. H., Lo, H., and Esfeh, M. A. Optimization of traffic sensor location for complete link flow observability in traffic network considering sensor failure. Transportation Research Part B: Methodological 121 (2019), 216–251. https://doi.org/10.1016/j.trb.2019.01.004.

[118] Shan, D., Sun, X., Liu, J., and Sun, M. Optimization of scanning and counting sensor layout for full route observability with a bi-level programming model. Sensors 18, 7 (2018), 2286. https://doi.org/10.3390/s18072286.

[119] Sherali, H. D., and Alameddine, A. A new reformulation-linearization technique for bilinear programming problems. Journal of Global Optimization 2, 4 (1992), 379–410. https://doi.org/10.1007/BF00122429.

[120] Siarry, P. Metaheuristics. Springer, 2016. https://doi.org/10.1007/978-3-319-45403-0.

[121] Sierra Altamiranda, A., and Charkhgard, H. A new exact algorithm to optimize a linear function over the set of efficient solutions for biobjective

mixed integer linear programs. INFORMS Journal on Computing 31, 4 (2019), 823–840. https://doi.org/10.1287/ijoc.2018.0851.

[122] SINHA, A., MALO, P., AND DEB, K. A review on bilevel optimization: from classical to evolutionary approaches and applications. IEEE Transactions on Evolutionary Computation 22, 2 (2017), 276–295. https://doi.org/10.1109/TEVC.2017.2712906.

[123] SOYSTER, A. L. Convex programming with set-inclusive constraints and applications to inexact linear programming. Operations Research 21, 5 (1973), 1154–1157. https://doi.org/10.1287/opre.21.5.1154.

[124] SUN, W., AN, C., LI, G., AND LV, Y. Applications of inexact programming methods to waste management under uncertainty: current status and future directions. Environmental Systems Research 3, 1 (2014), 15. https://doi.org/10.1186/s40068-014-0015-9.

[125] VAJDA, S. Mathematical Programming. Addison-Wesley, Reading Mass., USA, 1961.

[126] WANG, N., GENTILI, M., AND MIRCHANDANI, P. Model to locate sensors for estimation of static origin–destination volumes given prior flow information. Transportation Research Record 2283, 1 (2012), 67–73. https://doi.org/10.3141%2F2283-07.

[127] WANG, X., AND HUANG, G. Violation analysis on two-step method for interval linear programming. Information Sciences 281 (2014), 85–96. https://doi.org/10.1016/j.ins.2014.05.019.

[128] WINSTON, W. L., VENKATARAMANAN, M., AND GOLDBERG, J. B. Introduction to Mathematical Programming, vol. 1. Thomson/Brooks/Cole Duxbury; Pacific Grove, CA, 2003.

[129] Xie, F., Butt, M., Li, Z., and Zhu, L. An upper bound on the minimal total cost of the transportation problem with varying demands and supplies. Omega 68 (2017), 105–118. https://doi.org/10.1016/j.omega.2016.06.007.

[130] Xu, X., Lo, H. K., Chen, A., and Castillo, E. Robust network sensor location for complete link flow observability under uncertainty. Transportation Research Part B: Methodological 88 (2016), 1–20. https://doi.org/10.1016/j.trb.2016.03.006.

[131] Yamamoto, Y. Optimization over the efficient set: overview. Journal of Global Optimization 22, 1-4 (2002), 285–317. https://doi.org/10.1023/A:1013875600711.

[132] Yang, H., Iida, Y., and Sasaki, T. An analysis of the reliability of an origin-destination trip matrix estimated from traffic counts. Transportation Research Part B: Methodological 25, 5 (1991), 351–363. https://doi.org/10.1016/0191-2615(91)90028-H.

[133] Yang, H., Yang, C., and Gan, L. Models and algorithms for the screen line-based traffic-counting location problems. Computers & Operations Research 33, 3 (2006), 836–858. https://doi.org/10.1016/j.cor.2004.08.011.

[134] Yang, H., and Zhou, J. Optimal traffic counting locations for origin–destination matrix estimation. Transportation Research Part B: Methodological 32, 2 (1998), 109–126. https://doi.org/10.1016/S0191-2615(97)00016-7.

[135] Yim, P. K., and Lam, W. H. Evaluation of count location selection methods for estimation of od matrices. Journal of Transportation Engineering 124, 4 (1998), 376–383. https://doi.org/10.1061/(ASCE)0733-947X(1998)124:4(376).

[136] Zheng, Y., Lee, I., and Serban, N. Regularized optimization with spatial coupling for robust decision making. European Journal of Operational Research 270, 3 (2018), 898–906. https://doi.org/10.1016/j.ejor.2017.10.037.

[137] Zhou, F., Huang, G. H., Chen, G.-X., and Guo, H.-C. Enhanced-interval linear programming. European Journal of Operational Research 199, 2 (2009), 323–333. https://doi.org/10.1016/j.ejor.2008.12.019.

[138] Zhou, G., Chung, W., and Zhang, Y. Measuring energy efficiency performance of china's transport sector: A data envelopment analysis approach. Expert Systems with Applications 41, 2 (2014), 709–722. https://doi.org/10.1016/j.eswa.2013.07.095.

[139] Zhou, X., and List, G. F. An information-theoretic sensor location model for traffic origin-destination demand estimation applications. Transportation Science 44, 2 (2010), 254–273. https://doi.org/10.1287/trsc.1100.0319.

[140] Zhu, S., Guo, Y., Chen, J., Li, D., and Cheng, L. Integrating optimal heterogeneous sensor deployment and operation strategies for dynamic origin-destination demand estimation. Sensors 17, 8 (2017), 1767. https://doi.org/10.3390/s17081767.

APPENDIX A

SUPPLEMENTS TO CHAPTER 5

Here, we briefly discuss the min-max and min-sum models proposed in [130]. Let us recall that we define graph $\mathcal{G} = (N^*, E)$ where N^* denotes non-centroid nodes and E represents the links. Let x_a for all $a \in E$ be a binary decision variable, where x_a is 1 if link a is selected as an unobserved link, 0 otherwise. We also define parameter γ_a^i as the node-link indicator: $\gamma_a^i = 1$ if link a is connected to node i and 0 otherwise. Xu et al. [130] used the concept of the new links, originally introduced in [17], to ensure that matrix A_U is non-singular for a given U. According to the concept, we assign to each non-centroid node a set of new links that are not already assigned to another non-centroid node. Let E_i be the set of new links connected to non-centroid node $i \in N^*$; it follows the following rules:

$$E = \bigcup_{i \in N^*} E_i,$$

$$E_i \cap A_{N^*/i} = \emptyset.$$

There are different ways of creating the set of new links E_i. We applied the node ranking procedure presented in [130]. The min-max formulations is as follows:

$$\min \max_{i \in N^*} \{ \sum_{a \in E} \gamma_a^i x_a \}$$

$$\text{subject to}$$

$$\sum_{a \in E_i} x_a = 1, \; i \in N^*, \tag{A.1}$$

$$x_a = \{0, 1\}, \; a \in E. \tag{A.2}$$

We can reformulate the above program by introducing an additional variable y, that is,

$$\min \; y \;\; \text{subject to} \;\; \sum_{a \in E} \gamma_a^i x_a \leq y \; \forall i \in N^*, \; (A.1) - (A.2).$$

The above formulation addresses the worst case (the largest number of the unobserved links connected to all non-centroid nodes). However, the min-sum model considers the cumulative number of unobserved links connected to each non-centroid node, i.e.,

$$\min \; \sum_{i \in N^*} \sum_{a \in E} \gamma_a^i x_a \;\; \text{subject to} \;\; (A.1) - (A.2).$$

.

CURRICULUM VITAE

NAME:	Mohsen Mohammadi
ADDRESS:	Department of Industrial Engineering University of Louisville, Louisville, KY, 40292
EDUCATION:	Ph.D., Industrial Engineering University of Louisville, Louisville, KY, 2021
	M.S., Industrial Engineering Azad University, Najafabad, Iran, 2013
	B.S., Industrial Engineering Azad University, Najafabad, Iran, 2010
AWARDS:	Industrial Engineering Doctoral Dissertation Award, J. B. Speed School of Engineering, University of Louisville, 2021
	Graduate Dean's Citation Award, University of Louisville, 2021
	Doctoral Fellowship Award, Department of Industrial Engineering, University of Louisville, 2021
	National Science Foundation Innovation Corp Award, University of Louisville, 2018
	The 2nd place, Excellence in Health Disparities Award, Research!Louisville 2018
	Graduate Dean's Recognition for the publishing academy, University of Louisville, 2017
	School of Interdisciplinary and Graduate Studies Fellowship Award, University of Louisville, 2016

.