How to quantify outcome functions of interval-valued linear programs

Mohsen Mohammadi ∗ Monica Gentili † Milan Hladík ‡ Raffaele Cerulli §¶

June 12, 2021

Abstract

We consider a linear programming problem where the input parameters are described by compact intervals. In this context, an interval linear program is a family of linear programs each associated with a realization of the interval data. In this paper, we investigate the outcome range problem which consists in finding the range of a function (herein referred to as an outcome function) over the set of all possible optimal solutions of an interval-valued linear program. We particularly focus on the case where uncertainty of the underlying interval linear program is only in the right-hand side. For this case, we address the computational complexity of the problem and study some of its theoretical foundations. We also discuss how it relates to other known problems in the literature. Given the computationally expensive nature of the problem, we design three heuristics to solve it. In particular, we present a method based on the reformulation-linearization technique to obtain an outer approximation of the range of an outcome function. We also develop two algorithms, a gradient-restoration based algorithm and a bases inspection approach, to compute an inner approximation of the range of an outcome function. We numerically test our methods on three different datasets and highlight their quality and efficiency.

Keywords: linear programming, interval data, outcome function, uncertainty modeling, heuristics.

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

∗Department of Mechanical and Industrial Engineering, University of Massachusetts Amherst, 160 Governors Drive, Amherst, 01003, USA
†Department of Industrial Engineering, University of Louisville, J. B. Speed Building, 2301 S 3rd St, Louisville, KY 40292, USA
‡Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Malostranské nám. 25, 11800 Prague, Czech Republic
§Department of Mathematics, University of Salerno, Via Giovanni Paolo II, 132, Fisciano 84084, SA, Italy
¶E-mail: mmohammadide@umass.edu (Mohsen Mohammadi), monica.gentili@louisville.edu (Monica Gentili), hladik@kam.mff.cuni.cz (Milan Hladík), and raffaele@unisa.it (Raffaele Cerulli).
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 [7, 11, 30, 57], matrix games with interval-valued payoffs [35, 38], portfolio selection problems with interval approximations of expected returns [32, 33], project management with interval task durations [12], and environmental [8, 37] and waste [56] 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 [24, 49, 51]. Different topics have been studied in interval linear programming (see [24] 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 [9, 17, 23, 44, 49] and its tight approximation [26, 42] and (ii) describing the set of all optimal solutions resulting from all the possible realizations of the interval data [1, 15, 18] and its inner [28] and outer [25] approximations. Recently, Mohammadi and Gentili [43] 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. [46] and Gentili et al. [19, 20, 21] 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 [43]. Similarly, the notion of the outcome function can be of great interest in the telecommunications network design and evacuation planning [59]. Quantifying outcome functions was also mentioned in [34] 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, [6, 54, 58] 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 [2, 4, 13, 40, 47, 52] 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 2). We then discuss the computational complexity of the problem (Section 3). We also study some theoretical aspects, related to property and geometry of scenarios, of this
special class of the problem (Section 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 5). Furthermore, given the hardness of computation of the outcome range problem, we offer some heuristics to solve the problem (Section 6). In particular, we present an outer approximation for our problem using the reformulation-linearization technique (Section 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 [41] where the goal is to improve a current incumbent solution by moving alternately between feasible and infeasible solutions at each iteration (Section 6.2), and for the second algorithm, we develop a guided optimal bases search (Section 6.3). We test our methods on three test beds (Section 7), and finally in the last section, we summarize our findings and discuss some future directions.

2 Problem definition

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

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

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 (1) essentially can be in one of the following forms

\[ z(A, b, c) = \min c^T x \text{ subject to } Ax = b, \quad x \geq 0, \]  
\[ z(A, b, c) = \min c^T x \text{ subject to } Ax \leq b, \]  
\[ z(A, b, c) = \min c^T x \text{ subject to } Ax \leq b, \quad x \geq 0. \]

An interval matrix is a set of matrices

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

where \( \underline{A}, \overline{A} \in \mathbb{R}^{m \times n} \) are given, and comparing matrices is understood componentwise. We denote by \( \mathbb{I} \mathbb{R} \) 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 \( A \in \mathbb{I}R^{m \times n}, b \in \mathbb{I}R^m, \) and \( c \in \mathbb{I}R^n \), we define an interval linear program as a family of linear programs with \( A \in A, b \in b, \) and \( c \in c, \) i.e.,

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

In short, we can write the preceding as

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

We refer to any triplet \((A, b, c)\) with \( A \in A, b \in b, \) and \( c \in c \) as a scenario, and thus LP (1) is associated with the scenario \((A, b, c)\). We denote by \( S(A, b, c) \) the set of optimal
solutions of (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

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

Now consider an extra function $f : \mathbb{R}^n \to \mathbb{R}$ where $f(x) = r^T x$ with $r \in \mathbb{R}^n$. The outcome range problem is the problem of finding the maximum and the minimum values of $f(x)$ over the optimal set ($\Omega$) [43], that is,

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

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

In this sense, the range $[f, \overline{f}]$ 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 [18, 27]. 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 [24]. 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 $b$. This is particularly true for network optimization problems such as transportation problems, maximum flow problems, minimum-cost flow problems, multicommodity network flow problems, etc [3, 45]. We formally consider

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

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$. By LP($b$) we mean the linear program associated with ILP (2) for a given $b \in b$. 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 $b \in b$, respectively. We refer to the optimal set of (2) as $\Omega_b$. Hereinafter, we only focus on solving the following problems

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

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

We refer to this special class of the outcome range problem as ORP$_b$.

**Observation 1.** By [14, 15], we know that $\Omega_b$ is path connected. It is bounded if $S(b)$ is non-empty and bounded for some $b \in b$. Therefore, in the case of ORP$_b$, we can say that $f(x)$ is continuous in $[f, \overline{f}]$ if $\Omega_b$ is bounded.

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

**Assumption 1.** We assume that there exists $b \in b$ such that LP($b$) admits a finite optimal value.

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

**Example 1.** Consider the following interval linear program

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

and an outcome function \( f(x) = 3x_1 - 2x_2 \). Figure 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 (\( \Omega_b \)) 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, \( f \) 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 [43]. 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 on a wider range of problem instances.

### 3 Computational complexity

From Figure 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 [25]. This builds complexity of the outcome range problem. In this section, we address complexity of the ORP. 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 (2) reads as

\[
\underline{z} := \inf \{ z(b) : \ b \in \mathbf{b} \}, \tag{5}
\]

\[
\overline{z} := \sup \{ z(b) : \ b \in \mathbf{b} \}, \tag{6}
\]

\footnote{This is a modification of Example 3 in [15].}
where $z$ and $\overline{z}$ can take on finite values, infinity or infeasibility. The range $[z, \overline{z}]$ then gives the optimal value range.

**Theorem 1.** Problem of finding (3) and (4) is NP-hard.

**Proof.** By Theorem 7 in [15] (p. 282), we know that computing the exact interval hull of the optimal set of ILP (2) is NP-hard. Now if we put $f(x) = x_i$, for any $i \in \{1, \ldots, n\}$, we can conclude that ORP$_b$ also is NP-hard.

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

**Proposition 1.** 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 $z$.

**Proof.** By [16], 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

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

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 [48]. 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 1 reveals the following results related to ORP$_b$.

**Corollary 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 1 by setting $r := \pm c$.

**Corollary 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$_b$ is an NP-hard problem in general, there are few polynomially solvable cases of the problem. We present below one of these cases.

**Proposition 2.** Suppose that in ILP (2), coefficient matrix $A$ is a square matrix of full rank. Then, ORP$_b$ is polynomially 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$_b$ 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 $b$. 

6
Remark 1. For the sake of brevity, hereinafter, we only discuss results related to the computation of $f$, that is, problem (3). All the results are also applicable to problem (4).

4 Properties of ORP

Here, we investigate some theoretical properties of ORP. Before going into the details, let us review some definitions.

Definition 1. A vector $x \in \mathbb{R}^n$ is called a weak feasible solution of ILP (2) if there exists $b \in b$ such that $Ax = b$, $x \geq 0$.

Proposition 3. (see [50] for a proof) The set of all the weak feasible solutions of ILP (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 (2) for a given $b \in b$,

$$\min c^T x \text{ subject to } Ax = b, x \geq 0. \quad (7)$$

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 $N := \{1, \ldots, m\} \setminus B$ indicates indices for nonbasic variables and as a subscript it represents restriction to nonbasic indices. A basis $B$ is an optimal basis of (7) if and only if the following conditions hold

$$A_B^{-1}b \geq 0, \quad (8a)$$
$$c_T^N - c_T^B A_B^{-1} A_N \geq 0^T. \quad (8b)$$

Since vector $c$ is fixed, for a given optimal basis, we call (8a) its basis stability region. An ILP problem may have several optimal bases, and consequently several basis stability regions.

Proposition 4. The set of all scenarios $b \in b$ 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 $b \in b$ for which there exists $x \geq 0$ such that $Ax = b$. Set $A(\mathcal{F})$ is naturally bounded by the box $b$ and is the image of convex polyhedron $\mathcal{F}$ under a linear map, so it is a bounded convex polyhedron. \qed

The above proposition implies that the uncertainty space of ILP (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 3. The union of all the basis stability regions of ILP (2) is convex.

We use this observation in one of our algorithms in Section 6. Also, Example 4 in Section 6.3 gives an illustration of the above corollary.

We now explore a property related to the scenario corresponding to $f$. 

7
Definition 2. $b^*$ is an optimal scenario of (3) if $\underline{f} = f(x^*)$, where $x^* \in S(b^*)$.

Definition 3. Given an optimal scenario $b^*$ of (3), an optimal basis $B^*$ corresponding to $x^* \in S(b^*)$ such that $\underline{f} = f(x^*)$ is a global optimal basis of (3).

Given a global optimal basis $B^*$, it is natural to find $\underline{f}$ and $b^*$ by solving the following linear program
\[
\min r^T B^{-1} A \quad \text{subject to} \quad A B^{-1} b \geq 0, \quad b \in b^*,
\] (9)
in variable $b$.

Proposition 5. Optimal scenario $b^*$ is either a vertex of $b$ or a realization $b^* \in b$, for which LP($b^*$) is degenerated.

Proof. If no optimal scenario is a vertex of $b$, then at least one optimal scenario lies on the boundary of at least two polyhedra characterized by the feasible solution set of (9). Therefore, the optimal solution of LP($b^*$) is degenerated.

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

Proof. By Proposition 3, we know that the set of all the weak feasible solutions is described by
\[
Ax \leq \overline{b}, \quad -Ax \leq -\overline{b}, \quad x \geq 0.
\] Its recession (or characteristic) cone is described by $Ax = 0, \quad x \geq 0$. So if the minimum of $r^T x$ over the recession cone is 0, then the minimum of $r^T x$ over $F$ is bounded, and therefore also the minimum of $r^T x$ over $\Omega_b$ is bounded.

5 How does ORP$_b$ relate to other problems?

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

5.1 Optimal value range problem

Let us recall the optimal value range problem from Section 3, that is, problems (5) and (6). There is a relation between ORP$_b$ and the optimal value range problem. Indeed, ORP$_b$ 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 2. If $\underline{z}$ and $\overline{z}$ are finite, we then can reformulate the optimal value range problem as
\[
\underline{z} = \min c^T x \quad \text{subject to} \quad x \in \Omega_b,
\] (10)
\[
\overline{z} = \max c^T x \quad \text{subject to} \quad x \in \Omega_b.
\] (11)
We see that, under the assumption of Observation 2, $f_f$ would be a generalization of $z_z$. Indeed, in such a case, if $f(x) = c^T x$ then looking for $[f_f]$ is equivalent to looking for $[z_z]$. Note that such a relationship between $f_f$ and $z_z$ does not hold in general, as shown in the following example.

**Example 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 2 depicts the above ILP problem. From the figure, we see that $z_z = 0$. It is also easy to see that $LP(b = -1)$ is infeasible and, by convention [49], we set $z(b = -1) = \infty$. Thus, the range $[0, \infty)$ gives the optimal value range. However, from Figure 2, it is not hard to observe that $f(x)$ ranges in $[0, 15]$, which is different from the optimal value range.

![Figure 2: (Example 2) The set of all the weak feasible solutions and the optimal set are in grey and bold, respectively.](image)

Moreover, apart from the relation in the theory, the two problems are essentially different from the application standpoint. 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).

### 5.2 Multiobjective optimization

A multiobjective linear programming problem is defined as

$$\min Cx \text{ subject to } M(A, b), \quad (12)$$
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 (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 ORP$_{b}$ and multiobjective linear programs with interval right-hand sides. Let us reformulate (3) as the following interval multiobjective linear program

$$\begin{align*}
\min & \quad f_1(x) = r^T x \\
\min & \quad f_2(x) = c^T x \\
\text{subject to} & \quad Ax = b, \quad x \geq 0.
\end{align*}$$

(13)

**Definition 4.** A solution $x^e$ is called a possibly efficient solution of (13) if it is feasible and efficient for some $b \in b$.

Problems (3) and (13) are not equivalent because the latter always deals with a larger set of solutions. We show this by the following example.

**Example 3.** Consider the following linear program with an interval right-hand side

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

(14)

and an outcome function $f(x) = -2x_1 + 5x_2$. Now let us recall the multiobjective reformulation of (3):

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

(15)

Figure 3a shows the optimal set of (14), while Figure 3b depicts the set of all the possibly efficient solutions of (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 3a, we can observe that $\underline{f} = 0$. By considering the set of all the possibly efficient solutions in Figure 3b, the minimum value of the first objective is $-10$, which is different from what we observe from Figure 3a. Thus, we can conclude that (3) and (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 7. Let \( x^* \in \Omega_b \) be given such that \( \underline{f} = f(x^*) \). Then \( x^* \) is a possibly efficient solution of (13).

Proof. Let \( b^* \) be an optimal scenario of (3). We can see that in (13), the following holds

\[
f_2(x^*) \leq f_2(\hat{x}), \quad \forall \hat{x} \in \{x : Ax = b^*, x \geq 0\}.
\]

We also know that

\[
\underline{f} = f_1(x^*) \leq f_1(\tilde{x}), \quad \forall \tilde{x} \in \Omega_b.
\]

Therefore, there exists no \( \hat{x} \in \{x : Ax = b^*, x \geq 0\} \) 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 (13).

5.3 Bilevel optimization

Let us reformulate (3) as the following bilevel linear program

\[
\begin{array}{ll}
\text{"min"} & 0^T b + r^T x \\
\text{subject to} & b \in b, \\
& x \in \arg\min_x \{c^T x \text{ subject to } Ax = b, x \geq 0\}. \\
\end{array}
\]

(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 \( b \in b \). In other words, in case of multiple follower optimal solutions for a given \( b \), it is not clear how the follower behaves.

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

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

where \( \Psi : \mathbb{R}^m \to \mathbb{R}^n \). An equivalent formulation of (16) can be seen as

\[
\begin{array}{ll}
\text{"min"} & 0^T b + r^T x \\
\text{subject to} & b \in b, x \in \Psi(b). \\
\end{array}
\]

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

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

Bilevel linear program (16) under the optimistic position of the follower can be formulated as

\[
\begin{array}{ll}
\text{min} & 0^T b + r^T x \\
\text{subject to} & b \in b, x = \Psi^o(b). \\
\end{array}
\]

(17)

By considering Assumption 1 in Section 2, it is easy to see that (3) and (17) are equivalent problems. Thus, ORP\( b \) can be equivalently formulated as two optimistic bilevel linear programs. However, bilevel linear programming is too general to solve ORP\( b \) efficiently as it is known to be strongly NP-hard [22]. We also confirm this by our numerical experiments\(^3\).

\(^3\)Interested readers are referred to Ref. [55] for a review on bilevel optimization.
5.4 Mixed integer LP formulation

Let us here remind you the program (7), which corresponds to ILP (2) for a given \( b \in b \). The dual of (7) reads

\[
\max \, b^T y \text{ subject to } A^T y \leq c, \quad (18)
\]

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 (7) and dual (18) optimal solutions if and only if they solve the system

\[
Ax = b, \ x \geq 0, \ A^T y \leq c, \ x^T (c - A^T y) = 0.
\]

Now we assume, without loss of generality, that \( b \) is a vector of variables with given lower and upper bounds (i.e., \( b \in b \)). We can then describe the optimal set of ILP (2) by the following parametric system

\[
Ax = b, \ x \geq 0, \ A^T y \leq c, \ x^T (c - A^T y) = 0, \ b \in b. \quad (19)
\]

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

\[
x_i = 0 \lor h_i = 0, \ \forall i \in \{1, \ldots, n\}.
\]

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 (19) as

\[
Ax = b, \ x \geq 0, \ c - A^T y = h, \ x \leq \phi L, \ h \leq (1 - \phi) L, \ h \geq 0, \ b \in b, \ \phi \in \{0, 1\}^n, \quad (20)
\]

where \( L \) is a sufficiently large constant. We now can solve ORP\(_b\) to optimality by solving two mixed integer linear programs, that is, minimizing (maximizing) \( f(x) \) subject to system (20).

6 Approximating ORP\(_b\)

Since the structure of the optimal set \((\Omega_b)\) can be very complicated, we often are not able to solve ORP\(_b\) to optimality. Thus, in this section, we present three methods to approximate ORP\(_b\). 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.

6.1 Outer approximation: A reformulation-linearization technique

The idea behind our method is that we first describe \( \Omega_b \) by a nonlinear system, and we then linearize it by utilizing the reformulation-linearization technique (RLT) introduced in [53].

If, instead of the complimentary slackness condition in Section 5.4, we consider the strong duality condition in linear programming, we then can describe \( \Omega_b \) as the set of solutions to the following system

\[
Ax = b, \ A^T y \leq c, \ c^T x = b^T y, \ x \geq 0, \ b \in b, \quad (21)
\]
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 nonnegative 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 2.** System (19) has $n$ bilinear terms in $n$ constraints, while system (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 (21).

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

\[
\overline{y}_i - y_i \geq 0 \quad \forall i \in \{1, \ldots, m\},
\]

(22a)

\[
y_i - y_i \geq 0 \quad \forall i \in \{1, \ldots, m\}.
\]

(22b)

We similarly define the bound factors for $b$ as

\[
b_i - \overline{b}_i \geq 0 \quad \forall i \in \{1, \ldots, m\},
\]

(23a)

\[
\overline{b}_i - b_i \geq 0 \quad \forall i \in \{1, \ldots, m\}.
\]

(23b)

The goal is to construct the nonlinear terms $b_i y_i$ in the constraints of (21) by using the above bound factors. This can be done in many ways one of which is pairwise multiplying constraints in (22) and (23). For the sake of illustration, suppose the multiplication of (22a) and (23a)

\[
(\overline{y}_i - y_i) (b_i - \overline{b}_i) \geq 0 \quad \forall i \in \{1, \ldots, m\}.
\]

This can be then rearranged into

\[-y_i b_i + \overline{y}_i b_i + \overline{b}_i y_i \geq \overline{y}_i \overline{b}_i \quad \forall i \in \{1, \ldots, m\}.\]

System (21) together with all the bilinear constraints, generated as explained above, yields an equivalent reformulation of $\Omega_b$, that is,

\[
[R_1] \quad Ax = b, \quad A^T y \leq c, \quad c^T x = b^T y, \quad x \geq 0, \quad b \in b,
\]

\[-y_i b_i + \overline{y}_i b_i + \overline{b}_i y_i \geq \overline{y}_i \overline{b}_i \quad \forall i \in \{1, \ldots, m\},
\]

\[-y_i b_i + b_i \overline{y}_i + \overline{b}_i y_i \geq \overline{b}_i \overline{y}_i \quad \forall i \in \{1, \ldots, m\},
\]

\[y_i b_i - b_i \overline{y}_i - \overline{b}_i y_i \geq -\overline{b}_i \overline{y}_i \quad \forall i \in \{1, \ldots, m\},
\]

\[y_i b_i - \overline{b}_i y_i - \overline{b}_i y_i \geq -\overline{b}_i \overline{y}_i \quad \forall i \in \{1, \ldots, m\}.\]

Note that $R_1$ is one reformulation of (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 (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 $A^Ty \leq c$ and (23). As an example,

$$(c - A^Ty)(b_i - b_i') \geq 0 \forall i \in \{1, \ldots, m\}.$$  

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 (22) and (23).

After a simplification, an alternative reformulation of $\Omega_b$ reads

$$\text{[R2]} \quad A x = b, \quad A^Ty \leq c, \quad c^Tx = b^Ty, \quad x \geq 0, \quad b \in \mathbf{b},$$

$$-b_i A^Ty + c b_i + b_i A^Ty \geq c b_i' \quad \forall i \in \{1, \ldots, m\},$$

$$b_i A^Ty - c b_i - b_i A^Ty \geq -c b_i' \quad \forall i \in \{1, \ldots, m\},$$

$$-y_j b_i + b_i y_j + b_i y_j \geq b_i y_j' \quad \forall i, j \in \{1, \ldots, m\},$$

$$-y_j b_i + b_i y_j + b_i y_j \geq b_i y_j' \quad \forall i, j \in \{1, \ldots, m\},$$

$$y_j b_i - b_i y_j - b_i y_j \geq -b_i y_j' \quad \forall i, j \in \{1, \ldots, m\},$$

$$y_j b_i - b_i y_j - b_i y_j \geq -b_i y_j' \quad \forall i, j \in \{1, \ldots, m\}.$$ 

**Linearization phase.** We linearize nonlinear constraints in R1 and R2 through an appropriate variable substitution. Particularly, to linearize system R1, we substitute

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

and for linearizing system R2, we put

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

Let $E(R_1)$ and $E(R_2)$ be respectively the linearized systems R1 and R2 after such a substitution. In our computational experiments discussed in Section 7, we assess the performance of the two following outer approximations (i.e., lower bounds)

$$\text{[RLT1]} \quad f^{L_1} = \min_r r^T x \text{ subject to } (x, y, b, w) \in E(R_1),$$

$$\text{[RLT2]} \quad f^{L_2} = \min_r r^T x \text{ subject to } (x, y, b, w) \in E(R_2).$$

### 6.2 Inner approximation: A gradient-restoration based algorithm

We here design an algorithm to explore $\Omega_b$. The idea builds from the gradient-restoration algorithm which was originally developed to minimize constrained non-smooth functions [41]. Our algorithm basically alternates between an improving phase and a restoration phase and can be applied to find an upper bound for $f$. Figure 4 is an illustration on how the algorithm works. From a high level perspective, we start from a feasible point of (3), that is, a given $x_k \in \Omega_b$ (Figure 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 4b). If the new point $u$ belongs to $\Omega_b$, 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 $u \not\in \Omega_b$ to obtain a new point $\hat{x}$ such that $\hat{x} \in \Omega_b$ (Figure 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 1. We now discuss each step in detail.

**Algorithm 1: A gradient-restoration based algorithm**

**Input:** input data  
**Result:** an upper bound of \( f \)  
Set \( k \leftarrow 0 \).

**Step 0.** (initial point): Find an initial \( x_k \in \Omega_b \) and set \( \hat{f}_k \leftarrow f(x_k) \).

while we can move to a new point in the weak feasible solution set do

**Step 1.** (improving step) 
Select \( u \in N(x_k) \) such that \( f(u) \leq f(x_k) \).

**Step 2.** (certification step) 
Set \( k \leftarrow k + 1 \).  
if \( u \in \Omega_b \) then  
    Put \( x_k \leftarrow u \) and \( \hat{f}_k \leftarrow f(u) \).
else
    **Step 3.** (restoration step)  
    Retrieve a \( \hat{x} \in \Omega \).  
    Set \( x_k \leftarrow \hat{x} \) and \( \hat{f}_k \leftarrow f(\hat{x}) \).
end

end

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

**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

\[
u = x_k + \xi \alpha_{\text{max}} d,
\]

where \( \xi \in (0, 1] \), and \( \alpha_{\text{max}} \) is the maximum perturbation in direction of \( d \) such that weak feasibility of \( u \) is preserved. By Proposition 3, the following univariate linear
program yields $\alpha^{\text{max}}$

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

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

**Theorem 2.** (see [36]) Let $u$ be a weak feasible solution of ILP (2), and assume that $u_{q_1} > 0, \ldots, u_{q_p} > 0, \; u_{q_p+1} = 0, \ldots, u_{q_n} = 0$. Then $u$ is an optimal solution of ILP (2) for some $b \in b$ if and only if the following system is solvable

$$A^{T}_{q_j}t = c_{q_j}, \; j = 1, \ldots, p, \quad (26a)$$
$$A^{T}_{q_j}t \leq c_{q_j}, \; j = p+1, \ldots, n, \quad (26b)$$

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 (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

$$\min 0^T t \text{ subject to (26)}.\quad (26)$$

**Step 3.** (restoration step) Our aim, in this step, is to retrieve a point from $\Omega_b$. 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

$$A_B x_B = b, \; x_B \geq 0, \; x_N = 0,$$

which by Proposition 3 can be described by a polyhedral set

$$A_B x_B \leq \bar{b}, \; -A_B x_B \leq -\underline{b}, \; x_B \geq 0. \quad (27)$$

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

$$\min r^T x_B \text{ subject to (27)}.$$  

**Remark 3.** If $\alpha^{\text{max}} = 0$, we are on the border of the weak feasible solution set of ILP (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 (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.
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,

\[
\min r_B^T A_B^{-1} b \quad \text{subject to} \quad A_B^{-1} b \geq 0, \quad b \in b,
\]

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

**Example 4.** Consider the following ILP problem

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

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 (28) reads

- \( B_1 \): \( b_1 - b_2 \geq 0, \quad b_1 - 0.9b_2 \geq 0, \)
- \( B_2 \): \( b_1 - 0.8b_2 \geq 0, \quad -b_1 + b_2 \geq 0, \)
- \( B_3 \): \( -b_1 + 0.8b_2 \geq 0, \quad -0.67b_1 + 0.6b_2 \geq 0. \)

Solving (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 \( f = -38 \). Figure 5 depicts the interval vector \( b \) and its subparts corresponding to the basis stability regions of the three optimal bases.

**Figure 5:** (Example 4) The decomposition of interval vector \( b \) according to the basis stability regions of the optimal bases \( B_1, B_2, \) and \( B_3 \).
One natural way to approximate $f$ 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 $b_0 \in \mathbf{b}$. We find an optimal basis of $\text{LP}(b_0)$, namely, $B$. $\text{LP}$ (28) returns the lowest achievable $f$ at the optimal Basis $B$. Let $\tilde{b}_0$ be an optimal solution of (28). Now to go to another optimal basis, we slightly change $\tilde{b}_0$ in the direction of the derivative of outcome function, i.e., $\ell_B := -r_B^T A_B^{-1}$. That is,

$$b' = \tilde{b}_0 + \lambda \ell_B,$$

where $\lambda$ 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 $b_0$. We repeat the above procedure until we are not able to move to a new optimal basis. Algorithm 2 summarizes the steps of our proposed method.

**Remark 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 (28). After solving (28) with the arbitrary objective vector, we compute $b'$ according to the new $\ell_B$.

**Remark 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 2: A bases inspection algorithm**

<table>
<thead>
<tr>
<th>Input:</th>
<th>input data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Result:</strong></td>
<td>an upper bound of $f$</td>
</tr>
<tr>
<td>Set $\theta \leftarrow 0$.</td>
<td></td>
</tr>
<tr>
<td>Generate a random scenario $b_0 \in \mathbf{b}$.</td>
<td></td>
</tr>
<tr>
<td><strong>while</strong> it is possible to move to a new optimal basis <strong>do</strong></td>
<td></td>
</tr>
<tr>
<td>Find an optimal basis of $\text{LP}(b_0)$</td>
<td></td>
</tr>
<tr>
<td>Compute an approximation of $f$ by (28).</td>
<td></td>
</tr>
<tr>
<td>Determine $\tilde{b}_0$ and $\ell_B$.</td>
<td></td>
</tr>
<tr>
<td>Find $b'$ using (29).</td>
<td></td>
</tr>
<tr>
<td><strong>if</strong> $b' \notin \mathbf{b}$ <strong>then</strong></td>
<td></td>
</tr>
<tr>
<td>Project $b'$ onto $\mathbf{b}$.</td>
<td></td>
</tr>
<tr>
<td><strong>end</strong></td>
<td></td>
</tr>
<tr>
<td>Put $\theta \leftarrow \theta + 1$.</td>
<td></td>
</tr>
<tr>
<td>Set $b_0 \leftarrow b'$.</td>
<td></td>
</tr>
<tr>
<td><strong>end</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 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 ORP_b, that is, optimizing $f(x)$ subject to nonlinear system (21). FMINCON returns an inner approximation of the range of an outcome function. Additionally, to solve ORP_b to optimality, we use CPLEX to solve the MILP reformulation (see (20)) and YALMIP [39] to solve the bilevel reformulation (see (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.

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 [31]. These problems were also used in [34] 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 $b$ as $b_0 = Ax_0$ and $\bar{b} = \bar{b} + \delta e$, where $\delta$ is the interval width and $e$ is an all-ones vector with the convenient dimension. This procedure guarantees 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 $(m,n,\delta)$, 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 $(\omega,\rho)$, we considered the coefficient matrix of a balanced transportation problem with $\omega$ origins and $\rho$ 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 $b$ 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 $\omega, \rho$ and $\delta$, for a 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 1 reports size of the problems and number of non-zero elements in input parameters. As noted above, these problem instances were used in [34] as a test bed. Lee et al. [34] 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

\(^4\)Sample sizes here are similar to those considered in [34].
Table 1: Details of the problems chosen from the MIPLIB 2010 repository.

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

interval hull of the generated vectors to construct the interval vector $b$. We refer readers to Appendices in [10, 34] 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.

### 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 $\xi$ in BI to 0.2 and parameter $\lambda$ 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 $b$ in each problem instance. For our reformulation-linearization technique, we applied the contractor algorithm in [25] 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 $b \in b$. 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.


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 $f$.

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 $f$ 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 $f$ in a reasonable time. Given an estimation $\hat{f}$, we computed its gap from the optimal value by

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

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 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 ($\delta$). 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 $40 \times 60$. 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 3 highlights the results related to larger size problem instances in test bed 1. Our findings from Table 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 \((f^*)\) attained by BI or GR, if it outperforms an estimation found by FMIN \((f^{**})\), i.e., \(f^* \leq f^{**}\), we computed their relative distance by

\[
\text{distance} := \frac{f^{**} - f^*}{|f^{**}|}.
\]

Thus, a higher distance measure indicates a better performance of BI or GR compared to FMIN. In Table 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 \(\times\) 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 \(\delta\) 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. 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 \(\delta\) compared to the previous test bed. Similar to Table 2, Table 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 \(\times\) 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 \(b\) 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 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 \(b\), and the third column shows an average \(\delta\) 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 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 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 5, Table 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.

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 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 2: Results related to smaller size problem instances in test bed 1. An asterisk (*) denotes an average on 29 problem instances.
Table 3: Results related to larger size problem instances in test bed 1.

<table>
<thead>
<tr>
<th>size ((m \times n))</th>
<th>(\delta)</th>
<th>BI freq. avg. dist.</th>
<th>GR freq. avg. dist.</th>
<th>average running time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BI</td>
<td>GR</td>
<td>FMIN</td>
</tr>
<tr>
<td>50 \times 70</td>
<td>0.01</td>
<td>30 0.0001</td>
<td>30 0.0001</td>
<td>0.1056 0.0643</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>30 0.0002</td>
<td>30 0.0002</td>
<td>0.0972 0.0669</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>30 0.0047</td>
<td>29 0.0041</td>
<td>0.1054 0.0616</td>
</tr>
<tr>
<td>75 \times 100</td>
<td>0.01</td>
<td>30 0.0000</td>
<td>30 0.0000</td>
<td>0.1245 0.1069</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>30 0.0002</td>
<td>29 0.0002</td>
<td>0.1269 0.1073</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>30 0.0047</td>
<td>26 0.0066</td>
<td>0.1526 0.1036</td>
</tr>
<tr>
<td>100 \times 125</td>
<td>0.01</td>
<td>30 0.0001</td>
<td>30 0.0001</td>
<td>0.1686 0.1247</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>30 0.0014</td>
<td>30 0.0014</td>
<td>0.2454 0.1892</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>27 0.0049</td>
<td>22 0.0065</td>
<td>0.2735 0.1812</td>
</tr>
<tr>
<td>150 \times 200</td>
<td>0.01</td>
<td>30 0.0004</td>
<td>30 0.0004</td>
<td>0.3495 0.2680</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>30 0.0015</td>
<td>28 0.0016</td>
<td>0.3621 0.2661</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>30 0.0117</td>
<td>26 0.0083</td>
<td>0.4948 0.2640</td>
</tr>
</tbody>
</table>

Table 4: Comparing algorithms on test bed 2. An asterisk (*) denotes an average on 29 problem instances.

<table>
<thead>
<tr>
<th>size ((m \times n))</th>
<th>(\delta)</th>
<th>BI</th>
<th>GR</th>
<th>FMIN</th>
<th>RLT1</th>
<th>RLT2</th>
<th>MILP</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 \times 9</td>
<td>0.0000</td>
<td>0.0013</td>
<td>0.0000</td>
<td>0.7280</td>
<td>0.0019</td>
<td>0.0014</td>
<td>0.0014</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.0000</td>
<td>0.0100</td>
<td>0.0800</td>
<td>0.4368</td>
<td>0.1088</td>
<td>0.1217</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.0056</td>
<td>0.0101</td>
<td>0.2415 0.6541</td>
<td>0.0677</td>
<td>0.1818</td>
<td>0.0853</td>
</tr>
<tr>
<td>9 \times 25</td>
<td>0.0000</td>
<td>0.0010</td>
<td>0.0007</td>
<td>0.9433* 0.0729*</td>
<td>0.1826</td>
<td>0.1354</td>
<td>0.3142</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.0037</td>
<td>0.0283</td>
<td>0.0446 0.7666*</td>
<td>0.0391</td>
<td>0.2063</td>
<td>0.1474</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.3420</td>
<td>0.0355 0.1561</td>
<td>0.4322* 0.6522*</td>
<td>0.5269</td>
<td>0.1480</td>
<td>0.0835</td>
</tr>
<tr>
<td>19 \times 100</td>
<td>0.0123</td>
<td>0.0008</td>
<td>0.0325</td>
<td>0.9667* 0.0897*</td>
<td>0.5164</td>
<td>0.1187</td>
<td>1.3325</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.0075</td>
<td>0.0410</td>
<td>0.0650 2.2524*</td>
<td>0.2577</td>
<td>0.5117</td>
<td>0.1302</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.0046</td>
<td>0.0606 0.0269</td>
<td>0.8467* 0.8467*</td>
<td>0.9457</td>
<td>0.1727</td>
<td>1.0992</td>
</tr>
<tr>
<td>39 \times 400</td>
<td>0.0032</td>
<td>0.0051</td>
<td>0.0176</td>
<td>0.9740 0.1146*</td>
<td>0.6752</td>
<td>0.1740 33.7814</td>
<td>0.0954 2.7904</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.0489</td>
<td>0.0355 0.0222</td>
<td>1.9538* 0.2681*</td>
<td>0.5799</td>
<td>0.1943 33.1993</td>
<td>0.0966 2.8882</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.0838</td>
<td>0.0804 0.2677</td>
<td>5.8244* 0.8112*</td>
<td>0.6733</td>
<td>0.1586 30.9603</td>
<td>0.0966 2.9259</td>
</tr>
</tbody>
</table>

25
Table 5: Results obtained from test bed 3 (part I)

<table>
<thead>
<tr>
<th>model</th>
<th>no. of samples</th>
<th>average δ</th>
<th>distance from FMIN</th>
<th>running time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BI</td>
<td>GR</td>
<td>FMIN</td>
</tr>
<tr>
<td>enlight13</td>
<td>100</td>
<td>8.10</td>
<td>0.0046</td>
<td>-0.0904</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>9.66</td>
<td>0.0658</td>
<td>-0.0633</td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>10.54</td>
<td>0.0431</td>
<td>-0.0905</td>
</tr>
<tr>
<td></td>
<td>5,000</td>
<td>11.88</td>
<td>0.1463</td>
<td>-0.0021</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>12.40</td>
<td>0.0562</td>
<td>-0.0822</td>
</tr>
<tr>
<td></td>
<td>50,000</td>
<td>13.67</td>
<td>0.0945</td>
<td>0.0906</td>
</tr>
<tr>
<td></td>
<td>100,000</td>
<td>14.10</td>
<td>0.1743</td>
<td>0.0291</td>
</tr>
<tr>
<td></td>
<td>500,000</td>
<td>15.26</td>
<td>0.1349</td>
<td>0.0801</td>
</tr>
<tr>
<td>enlight15</td>
<td>100</td>
<td>9.40</td>
<td>0.1773</td>
<td>-0.1880</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>11.35</td>
<td>0.2083</td>
<td>0.0388</td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>12.23</td>
<td>0.0927</td>
<td>0.2953</td>
</tr>
<tr>
<td></td>
<td>5,000</td>
<td>13.81</td>
<td>0.4270</td>
<td>0.4405</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>14.42</td>
<td>0.3316</td>
<td>0.1691</td>
</tr>
<tr>
<td></td>
<td>50,000</td>
<td>15.75</td>
<td>0.2305</td>
<td>0.1942</td>
</tr>
</tbody>
</table>

Table 6: Results obtained from test bed 3 (part II)

<table>
<thead>
<tr>
<th>model</th>
<th>no. of samples</th>
<th>average δ</th>
<th>distance from GR</th>
<th>running time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BI</td>
<td>GR</td>
<td>FMIN</td>
</tr>
<tr>
<td>milk-250-1-100-1</td>
<td>100</td>
<td>1,958.06</td>
<td>0.0093</td>
<td>5.7573</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>2,272.89</td>
<td>0.0064</td>
<td>5.8112</td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>2,426.95</td>
<td>0.0104</td>
<td>5.8371</td>
</tr>
<tr>
<td></td>
<td>5,000</td>
<td>2,756.56</td>
<td>0.0118</td>
<td>5.7987</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>2,884.56</td>
<td>0.0069</td>
<td>5.7627</td>
</tr>
<tr>
<td></td>
<td>50,000</td>
<td>3,168.52</td>
<td>-0.0081</td>
<td>5.6830</td>
</tr>
<tr>
<td></td>
<td>100,000</td>
<td>3,303.37</td>
<td>-0.0105</td>
<td>5.7294</td>
</tr>
<tr>
<td></td>
<td>500,000</td>
<td>3,550.52</td>
<td>-0.0085</td>
<td>5.7810</td>
</tr>
<tr>
<td>roll3000</td>
<td>100</td>
<td>0.0011</td>
<td>0.0000</td>
<td>194.2224</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.0013</td>
<td>0.0000</td>
<td>194.5361</td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>0.0014</td>
<td>0.0000</td>
<td>201.1792</td>
</tr>
<tr>
<td></td>
<td>5,000</td>
<td>0.0016</td>
<td>0.0000</td>
<td>194.5419</td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>0.0017</td>
<td>0.0000</td>
<td>193.9927</td>
</tr>
<tr>
<td></td>
<td>50,000</td>
<td>0.0018</td>
<td>0.0000</td>
<td>194.5831</td>
</tr>
<tr>
<td></td>
<td>100,000</td>
<td>0.0019</td>
<td>0.0000</td>
<td>194.9193</td>
</tr>
<tr>
<td></td>
<td>500,000</td>
<td>0.0020</td>
<td>0.0000</td>
<td>201.5826</td>
</tr>
</tbody>
</table>

References


