RAIRO-Oper. Res. 56 (2022) 4281–4301
https://doi.org/10.1051/ro/2022198

RAIRO Operations Research
www.rairo-ro.org

FIX-AND-OPTIMIZE METAHEURISTICS FOR MINMAX REGRET BINARY INTEGER PROGRAMMING PROBLEMS UNDER INTERVAL UNCERTAINTY

Iago A. Carvalho[1], Thiago F. Noronha[2,*] and Christophe Duhamel[3]

Abstract. The Binary Integer Programming problem (BIP) is a mathematical optimization problem, with linear objective function and constraints, on which the domain of all variables is $\{0,1\}$. In BIP, every variable is associated with a determined cost coefficient. The Minmax regret Binary Integer Programming problem under interval uncertainty (M-BIP) is a generalization of BIP in which the cost coefficient associated to the variables is not known in advance, but are assumed to be bounded by an interval. The objective of M-BIP is to find a solution that possesses the minimum maximum regret among all possible solutions for the problem. In this paper, we show that the decision version of M-BIP is Σ_2^p-complete. Furthermore, we tackle M-BIP by both exact and heuristic algorithms. We extend three exact algorithms from the literature to M-BIP and propose two fix-and-optimize heuristic algorithms. Computational experiments, performed on the Minmax regret Weighted Set Covering problem under Interval Uncertainties (M-WSCP) as a test case, indicates that one of the exact algorithms outperforms the others. Furthermore, it shows that the proposed fix-and-optimize heuristics, that can be easily employed to solve any minmax regret optimization problem under interval uncertainty, are competitive with ad-hoc algorithms for the M-WSCP.

Mathematics Subject Classification. 68R01, 90C11, 90C17, 90C47, 90C59.

Received April 7, 2022. Accepted November 6, 2022.

1. Introduction

The Binary Integer Programming problem (BIP) is a mathematical optimization problem, with linear objective function and constraints, on which the domain of all variables is $\{0,1\}$. BIP can be formulated by the objective function (1) and the constraints (2) and (3), where b and c are n-dimensional vectors of coefficients,

Keywords. Minmax regret, interval uncertainty, binary integer programming, fix-and-optimize, metaheuristics.

[1] Computer Science Department, Universidade Federal de Alfenas, Av. Jovino Fernandes de Sales 2600, Alfenas, MG 37133-840, Brazil.

[2] Computer Science Department, Universidade Federal de Minas Gerais, Av. Pres. Antônio Carlos 6627, Belo Horizonte, MG 31270-901, Brazil.

[3] Laboratoire d'Informatique, du Traitement de l'Information et des Systèmes, Universitè Le Havre Normandie, 25 Rue Philippe Lebon, 76600 Le Havre, France.

*Corresponding author: tfn@dcc.ufmg.br

A is a $m \times n$-dimensional matrix of coefficients, and x is a n-dimensional vector of binary variables. BIP is NP-Hard [40]. However, for small- to medium-sized instances, it can be solved in many cases by branch-and-bound based algorithms available in state-of-the-art BIP solvers, such as CPLEX[1] and GUROBI[2] [4].

$$\text{(BIP)} \min c^T x \tag{1}$$

$$Ax \leq b \tag{2}$$

$$x \in \{0,1\}^n. \tag{3}$$

In this paper, we focus on a generalization of BIP whose parameters A and b are known, but the coefficients in c are uncertain. Despite not being known in advance, the cost of each coefficient c_i is assumed to be bounded by an interval $[l_i, u_i]$, with $0 \leq l_i \leq u_i$. Such uncertainty representation is known as interval uncertainty [44]. We refer to this problem as the Minmax regret Binary Integer Programming problem under interval uncertainty (M-BIP), which is formally defined in the following.

Definition 1.1. A scenario S is an assignment of a single value $c_i^S \in [l_i, u_i]$ for each uncertain coefficient c_i, $i = 1 \ldots n$.

Let Γ be the set of all scenarios, and Φ be the set of all feasible solutions to the constraints (2)–(3).

Definition 1.2. The cost of a solution $X \in \Phi$ in a specific scenario $S \in \Gamma$ is given by

$$F(X,S) = \sum_{i=1\ldots n} c_i^S x_i.$$

Definition 1.3. The best possible solution $Y^S \in \Phi$ for the specific scenario $S \in \Gamma$ is the optimal BIP solution for the scenario S

$$Y^S = \arg\min_{Y\in\Phi} F(Y,S) = \arg\min_{Y\in\Phi} \sum_{i=1\ldots n} c_i^S y_i.$$

Definition 1.4. The regret of a solution $X \in \Phi$ in a specific scenario $S \in \Gamma$ is the difference between the cost of X and the cost of Y^S in S [44]: $F(X,S) - F(Y^S,S)$.

Definition 1.5. The robustness cost $Z(X)$ of a solution $X \in \Phi$ is defined as the maximum regret of X among all scenarios in Γ:

$$Z(X) = \max_{S\in\Gamma} \left\{ F(X,S) - F(Y^S,S) \right\}.$$

Even if $|\Gamma| = \infty$, it is proven by [3] and [10] that the scenario on which the regret of $X \in \Phi$ is the maximum is the scenario $S^X \in \Gamma$, such that

$$c_i^{S^x} = l_i + (u_i - l_i)x_i$$

Thus, $c_i^{S^X} = u_i$ if $x_i = 1$, and $c_i^{S^X} = l_i$ otherwise. From this result, we have

$$F(X,S^X) = \sum_{i=1\ldots n} u_i x_i$$

$$F(Y^{S^X}, S^X) = \min_{Y\in\Phi} \sum_{i=1\ldots n} \big(l_i + (u_i - l_i)x_i\big) y_i$$

and

$$Z(X) = F(X,S^X) - F(Y^{S^X}, S^X).$$

[1]https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer
[2]http://www.gurobi.com

We refer to this scenario as the *worst-case scenario* S^X induced by solution X.

It is worth noticing that $F(Y^{S^X}, S^X)$ is a BIP, as in this case x_i is constant. Therefore, the robust cost of a solution X is computed by solving a single BIP problem in the scenario S^X. M-BIP aims at finding a solution with minimum robustness cost. It is defined as

$$\text{(M-BIP)} \quad \min_{X \in \Phi} Z(X) = \min_{X \in \Phi} \left\{ F\left(X, S^X\right) - F\left(Y^{S^X}, S^X\right) \right\}.$$

It can be reformulated as a 0–1 Bilevel Binary Program in (4) by substituting $F(X, S^X)$ and $F(Y^{S^X}, S^X)$.

$$\text{(M-BIP)} \qquad \min_{X \in \Phi} \left\{ \sum_{i=1\ldots n} u_i x_i - \min_{Y \in \Phi} \sum_{i=1\ldots n} \left(l_i + (u_i - l_i) x_i\right) y_i \right\}. \tag{4}$$

This formulation can be linearized by replacing $F(Y^{S^X}, S^S)$ with a free variable θ and adding a new set of linear constraints that bounds the value of θ to the value of $F(Y^{S^X}, S^X)$. The resulting M-BIP formulation (5)–(9) has an exponentially large number of constraints (6). The constraints (6) are linear since y_i is constant.

$$\min_{X \in \Phi} \sum_{i=1\ldots n} u_i x_i - \theta \tag{5}$$

$$\theta \leq \sum_{i=1\ldots n} \left(l_i + (u_i - l_i) x_i\right) y_i, \quad \forall\, Y \in \Phi \tag{6}$$

$$\sum_{i=1\ldots n} a_{ji} x_i \leq b_j, \quad \forall j = 1 \ldots m \tag{7}$$

$$x_i \in \{0, 1\}, \quad \forall i = 1 \ldots n \tag{8}$$

$$\theta \geq 0. \tag{9}$$

Proposition 1.6. *The decision version of M-BIP is Σ_2^p-complete*

Proof. M-BIP is a generalization of the decision version of the Minmax regret Binary Knapsack problem under interval uncertainty, which is also known to be Σ_2^p-complete [29]. □

Therefore, M-BIP is harder to solve than classical NP-Hard problems as its decision version does not belong to NP, unless $\Sigma_2^p = NP$ [50, 53]. This also means one cannot build a compact ILP formulation for M-BIP. Finally, it is also NP-Hard to compute the cost of a single solution of M-BIP.

In this paper, we propose two Fix-and-Optimize heuristics (FO) for M-BIP that also can be applied to solve any minmax regret optimization problem under interval uncertainty. These are the first FO heuristics in the literature applied to a Σ_2^p-hard problem under interval uncertainty. They are inspired by the Fix-and-Optimize metaheuristic framework proposed by [37] for a variant of the Vehicle Routing Problem and successfully adapted to several other combinatorial optimization problems [20,27,30,38,48,54]. We evaluate the performance of these heuristics using the Minmax regret Weighted Set Covering problem under interval uncertainty (M-WSCP) [51] as a test case. It is one of the most studied minmax regret optimization problems under interval uncertainty [7,22,23,28].

The remainder of this paper is organized as follows. Related work is reviewed in Section 2. Algorithms for M-BIP are presented in sections 3 and 4. The former shows exact algorithms for this problem while the latter presents the Fix-and-Optimize heuristics. In the latter, we also show how a lower bound can be obtained by using the proposed heuristics. Then, Section 5 formally defines the M-WSCP and reports the computational experiments. Finally, the concluding remarks are drawn in the last session.

Table 1. Complexity results for min-max regret optimization problems under interval uncertainty.

Problem	Complexity
Assignment	Strongly NP-Hard [1]
Binary Knapsack	Σ_2^p-Complete [29]
Binary Programming	Σ_2^p-Complete
Linear Programming	Strongly NP-Hard [9]
Min Cut	Polynomial [2]
Minimum Spanning Tree	Strongly NP-Hard [8]
Min s-t Cut	Strongly NP-Hard [2]
Representatives Selection	Polynomial [31]
Scheduling $1\|\| \Sigma C_j$	NP-Hard [45]
Shortest Path	Strongly NP-Hard [8]
Shortest Path Arborescence	NP-Hard [20]
Selection	Polynomial [10]
Weighted Independent Set	NP-Hard [41]
1-median (edge uncertainty)	Strongly NP-Hard [11]
1-median (node uncertainty)	Polynomial [55]
1-center (edge uncertainty)	Strongly NP-Hard [11]
1-center (node uncertainty)	Polynomial [55]

2. Related work

Minmax regret optimization problems under interval uncertainty were widely studied in the literature. They were studied from the theoretical point-of-view, where one is interested in determining the problem's complexity, and also from the algorithmic point-of-view, where one is interested in proposing algorithms for solving the problem. A general result for minmax regret optimization problems under interval uncertainty is that the uncertain variant is, at least, as hard to solve as the deterministic version of the problem [43]. This is due to the fact that, for computing the minmax regret of a solution, one must solve the deterministic problem on its worst-case induced scenario, as presented in Section 1. Table 1 summarizes the complexity studies for several minmax regret optimization problems under interval uncertainty in the literature displaying, for each problem, its known complexity. Several classic problems that are solvable in polynomial time, such as the minimum spanning tree problem and the shortest path problem, turn out to be NP-hard when considering the minmax regret objective function and the interval uncertainty. Furthermore, classic problems that were known to be NP-hard remain NP-hard in their minmax regret optimization variant under interval uncertainty. Others become harder to solve, like the Binary Programming problem and the Binary Knapsack problem.

The Minmax regret Weighted Set Covering problem under interval uncertainty was first studied by [51]. The authors proposed an integer linear programming formulation for this problem, which was solved by three algorithms based on Benders' decomposition. Several heuristics were also proposed for this problem. A Genetic Algorithm and a Hybrid Genetic Algorithm were proposed by [51], a scenario-based heuristic with a path-relinking strategy was proposed by [22], while a hybrid heuristic based on the Benders' decomposition algorithm and a scenario-based heuristic were developed by [23]. The latter heuristic was demonstrated to be the most efficient of them, see [23].

A fix-and-optimize heuristic was first used to solve minmax regret optimization problems by [20]. In this paper, the authors considered the Minmax regret Shortest Path Arborescence problem under interval uncertainty and proposed an ad-hoc fix-and-optimize heuristic. The proposed heuristic consists in solving several Minmax regret Shortest Path problems under interval uncertainty and uses their solutions to reduce the number of variables in the shortest path arborescence problem. Despite the encouraging results obtained by this heuristic, it cannot be extended to other minmax regret optimization problems under interval uncertainty.

In this research, we propose fix-and-optimize heuristics that can be easily employed to solve any minmax regret optimization problem under interval uncertainty. Unlike [20], they are based on (i) the linear programming relaxation of the problems; or (ii) the Scenario-based Heuristic (SBA) [23], which is a well-known heuristic that can be used to solve any problem within this class.

3. Exact algorithms

The M-BIP formulation (5)–(9) has an exponential number of the constraints (6). Enumerating all these constraints is a #P-Complete problem [26], since there is one constraint for each feasible solution of the problem. Therefore, this formulation cannot be explicitly solved by commercial BIP solvers, and more sophisticated methods are needed. Thus, we present below three exact algorithms for solving M-BIP.

3.1. Benders-like Decomposition (BLD)

The Benders-like Decomposition algorithm (BLD) relies on a cutting plane algorithm inspired by the Benders' Decomposition [14]. It differs from the classic algorithm because the BLD' subproblem is not a linear program. BLD decomposes M-BIP into two problems: (i) the Master Problem (MP); and (ii) the subproblem (SP). The MP consists in solving the formulation defined by objective function (5) and the constraints in (7)–(10), *i.e.* using only a subset of the constraints in (6). The SP consists in solving a BIP where the cost coefficients of the objective function are given by the worst-case scenario induced by the optimal solution of the MP. BLD iteratively solves the MP, then the SP. The solution Y of the SP is added to Φ such that $\Phi^{h+1} \leftarrow \Phi^h \cup \{Y\}$ where h is the current iteration. Initially $\Phi^0 \leftarrow \emptyset$. BLD stops when the robustness cost of the solution given by the MP is equal to the smallest robustness cost of a solution computed by the SP. This algorithm is guaranteed to obtain the optimal solution for M-BIP [6].

BLD was successfully applied to solve the Minmax regret Traveling Salesman Problem under Interval Uncertainty [46], the Minmax regret Knapsack problem under Interval Uncertainty [35], the Minmax regret Restricted Shortest Path problem under Interval Uncertainty [7], and the Minmax regret Set Covering problem under Interval Uncertainty [51], among other problems.

$$\theta \leq \sum_{i=1\dots n} (l_i + (u_i - l_i)x_i)\, y_i, \quad \forall\, Y \in \Phi^h \tag{10}$$

Algorithm 1 gives a pseudo-code of BLD. It receives as input the matrix A, the vector x, and the vector b of BIP. Additionally, it receives the vectors l and u of lower and upper bounds for the cost coefficients, such that $c_i = [l_i, u_i], i = 1 \dots n$. Initially, as suggested by [46] to avoid an unbounded MP, Φ^0 is initialized lines 1 to 3 with the solutions $z^{=}$ and z^{+} of two constructive heuristics for minmax regret optimization problems under interval uncertainty: the Algorithm Mean (AM) and the Algorithm Upper (AU) [42]. The best solution and its robustness cost are kept lines 4 and 5. Furthermore, the iteration count is set line 6. The loop lines 7–14 corresponds to an iteration of BLD. It is run until an optimal solution is found. First, the MP is solved line 8 using constraints (10), giving solution X. Then, the SP corresponding to X is solved in line 9, giving solution Y. Next, the primal bound ϑ of BLD is updated with the cost of Y line 10 and the solution with the smallest robustness cost is kept line 11. At the end of the loop, the iteration number is updated line 12 and the solution Y is inserted into Φ^h line 13. Finally, the best feasible solution X^* computed is returned line 15.

3.2. Extended Benders-like Decomposition (EB)

The Extended Benders-like Decomposition (EB) uses the methodology introduced by [33] for selecting Benders' cuts. It consists in solving one SP for each incumbent solution found by MP, instead of solving a single SP for the optimal solution of MP. Thus, the number of iterations in EB is expected to be smaller than in BLD.

Algorithm 1: Benders-like Decomposition (BLD)

```
input  : $\langle A, x, b, l, u\rangle$
output: $X^* \in \Phi$
1  $X^= \leftarrow AM(A, x, b, l, u)$
2  $X^+ \leftarrow AU(A, x, b, l, u)$
3  $\Phi^0 \leftarrow \{X^=, X^+\}$
4  $X^* \leftarrow \arg\min_{X \in \{X^= X^+\}} Z(X)$
5  $\vartheta \leftarrow Z(X^*)$
6  $h \leftarrow 0$
7  do
8      $X \leftarrow MP(A, x, b, \Phi^h, l, u)$
9      $Y \leftarrow SP(A, x, b, S^X)$
10     $\vartheta \leftarrow \min\left(\vartheta, F\left(Y^{(S^X)}, S^X\right)\right)$
11     $X^* \leftarrow \arg\min_{X \in \{X^*, Y\}} Z(X)$
12     $h \leftarrow h + 1$
13     $\Phi^h \leftarrow \Phi^{h-1} \cup \{Y\}$
14 while $(Z(X) < \vartheta)$
15 return $X^*$
```

The pseudo-code for EB is very similar to Algorithm 1 (BLD). The main difference is that, instead of solving an unique SP on line 10, it solves a SP for each integer solution found by the MP solved in line 8.

3.3. Branch-and-Cut (BC)

The Branch-and-Cut algorithm (BC) was initially proposed by [46] for solving the Minmax regret Traveling Salesman Problem under Interval Uncertainty. The authors noticed that BLD may be computationally inefficient, as each iteration runs a branch-and-bound algorithm from scratch to solve the MP. In BC, each incumbent solution found by the algorithm is processed by a SP (which is identical to SP in BLD and EB) to generate a new cut on the original branch-and-bound algorithm. BC differs from EB in the fact that the computed cuts are inserted globally and propagated to all active nodes in the branch-and-bound tree. The MP in BLD and EB is always reset after each iteration.

Algorithm 2 gives a pseudo-code of BC. It receives as input the matrix A, the vector x, and the vector b of M-BIP. Additionally, it receives the vectors l and u of lower and upper bounds for the cost coefficients, such that $c_i = [l_i, u_i], i = 1 \dots n$. Initially, as suggested by [46] and similar to BLD and EB, BC solves M-BIP using the AM and AU constructive heuristics lines 1 and 2, respectively. Then, it initializes $\Phi' \subseteq \Phi$ with solutions obtained by these heuristics line 3. Next, a branch-and-cut framework is performed line 4, which stops when an optimal solution is found or when a predefined time limit is met. Finally, the best feasible solution X^* computed is returned line 5.

4. Fix-and-Optimize heuristics

This section proposes two fix-and-optimize heuristics for M-BIP. These heuristics work in two steps. The first one is the *preprocessing step*, which uses an ad-hoc algorithm to fix the value of some variables in the problem's formulation. This leads to a reduced M-BIP, which is solved in the second step (*solving step*). One expects Reduced M-BIP to be solved significantly faster than M-BIP as the number of variables in the former will be much smaller.

The heuristics differ on the ad-hoc algorithm used in the preprocessing step. The first one uses the Scenario-based Heuristic (SBA) [22] to identify variables that can be fixed to zero and removed from the M-BIP

Algorithm 2: Branch-and-Cut (BC)

```
input  : $\langle A, x, b, l, u\rangle$
output: $X^* \in \Phi$
1 $X^= \leftarrow AM(A, x, b, l, u)$
2 $X^+ \leftarrow AU(A, x, b, l, u)$
3 $\Phi' \leftarrow \{X^=, X^+\}$
4 $X^* \leftarrow$ Branch-and-Cut$(A, x, b, \Phi', l, u)$
5 return $X$
```

formulation, while the second one uses the linear relaxation of the M-BIP formulation to achieve this goal. Despite being proposed for M-BIP, both heuristics are very general and they can be applied to solve any minmax regret optimization problem under interval uncertainty. They are described below.

4.1. Fix-and-Optimize through Scenario-based Algorithm (FO-SBA)

The Fix-and-Optimize through Scenario-based Algorithm (FO-SBA) relies on the SBA heuristic [22]. The intuition behind this heuristic is that the variables whose value is zero in the optimal solution of all scenarios solved by SBA are less likely to appear in the optimal solution of M-BIP. Therefore, they will be set to zero and removed from the set of variables, leading to a reduced M-BIP. In this subsection, we first introduce SBA in Section 4.1.1. Then, we state FO-SBA in Section 4.1.2.

4.1.1. The Scenario-based Algorithm

SBA was first proposed for the Minmax regret Weighted Set Covering problem under Interval Uncertainties [22, 23] and later extended to M-BIP [18] and other minmax regret optimization problems under interval uncertainties [19, 20, 24]. It consists in investigating a set $Q = \{q_1, q_2, \ldots, q_p\} \subseteq \Gamma$ of so-called target scenarios. Each one is a linear combination between the lower scenario $s^l \in \Gamma$ (a scenario where the cost of the arcs are set to their respective lower, *i.e.* $c_i^{s^l} = l_i$) and the upper scenario $s^u \in \Gamma$ (a scenario where the cost of the arcs are set to their respective upper, *i.e.* $c_i^{s^u} = u_i$).

The set Q of investigated scenarios is computed using three parameters: (i) the initial scenario α; (ii) the final scenario β; and (iii) the step size γ. All parameters are real-valued in the interval $[0, 1]$. The cost of the uncertain coefficients for each target scenario $q_j \in Q$ is computed as $c_i^{q_j} = l_i + (\alpha + \delta \cdot \gamma) \cdot (u_i - l_i)$, $\delta \in \mathbb{N}$, for all values of δ such that $\alpha + \delta\gamma \leq \beta$.

Algorithm 3 presents the pseudocode of SBA. It receives as input the matrix A, the vector x, and the vector b of BIP. Additionally, it receives the vectors l and u of lower and upper bounds for the cost coefficients, such that $c_i = [l_i, u_i], i = 1 \ldots n$, and parameters α, β, and γ. Initially, γ is set to α line 1 and the primal bound *vartheta* is set to infinity line 2. The algorithm consists in the main loop lines 3–8. In each iteration, a BIP is solved on scenario q_j (which is defined by the current value of δ) line 4. If the regret of the computed solution is smaller than that of the previous best solution, it updates the value of the best solution found line 6 and stores the solution found in line 7. The value of γ is incremented line 8. This loop runs until γ is greater than β. The best solution found X is then returned line 9. It is worth noting that SBA is an exponential-time algorithm, since the BIP solved in line 4 is NP-Hard.

4.1.2. FO-SBA for the M-BIP

Let V be the set of variables that appear in the optimal solution of at least one of the $k = |Q|$ scenarios investigated by SBA. Reduced M-BIP consists in the objective function (5) and the constraints in (6)–(9) and (11). One may observe that constraint (11) fix all variables that are not in V to zero.

$$x_i = 0 \quad \forall i \notin V. \tag{11}$$

Algorithm 3: Scenario-based Algorithm (SBA)

```
input  : $\langle A, x, b, \alpha, \beta, \gamma, l, u \rangle$
output: $X^* \in \Phi$
1 $\delta \leftarrow \alpha$
2 $\vartheta \leftarrow \infty$
3 while $\delta \leq \beta$ do
4     $X \leftarrow \mathrm{BIP}(A, x, b, \delta, l, u)$            // Solve BIP in the scenario defined by $\delta$
5     if $(Z(X) < Z(X^*))$ then
6         $\vartheta \leftarrow Z(X)$
7         $X^* \leftarrow X$
8     $\delta \leftarrow \delta + \gamma$
9 return $X^*$
```

Algorithm 4 shows the pseudo-code of FO-SBA. It receives as input the matrix A, the vector x, and the vector b of BIP. Additionally, it receives the vectors l and u of lower and upper bounds for the cost coefficients, such that $c_i = [l_i, u_i], i = 1 \ldots n$. Besides, is also receives as input the parameters of the SBA heuristic, *i.e.* α, β, and γ. Furthermore, it uses set V to maintain all variables that should be kept in the reduced M-BIP. Initially, γ is set to α in line 1. Then, line 2 initializes the set V as an empty set. The preprocessing step consists of the loop in lines 3–6. At each iteration of this loop, it solves a BIP on scenario q_j on line 4. Then, it stores the variables which were in the optimal solution of the previously solved problem on line 5 and increases the value of γ in line 6. The solving step is performed in line 7, which consists in solving the resulting reduced M-BIP formulation. One may observe that this formulation can be solved with any exact algorithm for M-BIP. Finally, FO-SBA returns the computed solution X^* on line 8.

Algorithm 4: FO-SBA

```
input  : $\langle A, x, b, \alpha, \beta, \gamma, l, u \rangle$
output: $X^* \in \Phi$
1 $\delta \leftarrow \alpha$
2 $V \leftarrow \emptyset$
3 while $\delta \leq \beta$ do
4     $X \leftarrow \mathrm{BIP}(A, x, b, \delta, l, u)$            // Solve BIP in the scenario defined by $\delta$
5     $V \leftarrow V \cup \{x_i : x_i = 1\}$
6     $\delta \leftarrow \delta + \gamma$
7 $X^* \leftarrow$ Reduced M-BIP$(A, x, b, l, u, V)$
8 return $X^*$
```

4.2. Fix-and-Optimize through Linear Relaxation (FO-LR)

The Fix-and-Optimize through Linear Relaxation (FO-LR) relies on the linear relaxation of the M-BIP formulation defined by Equations (5), (7)–(10). In this case, the integrality of the constraints in (8) is relaxed, and $\Phi^h = \Phi^0 = \{X^=, X^+\}$. This approach is the same used in the first iteration of BLD, EB, and BC, where $X^=$ and X^+ are the solutions of Algorithm Mean (AM) and Algorithm Upper (AU) [42], respectively.

The intuition behind this heuristic is that the variables whose value is zero in the optimal solution of the LP relaxation are less likely to appear in the optimal solution of M-BIP than the others. Therefore, one can build a reduced M-BIP without the null variables in the solution of the linear relaxation.

FO-LR builds a reduced M-BIP by setting some of the variables of M-BIP to zero. Let V be the set of null variables in the optimal solution of the linear relaxation of M-BIP. The reduced M-BIP formulation consists in solving the formulation (5)–(9) and (11).

Algorithm 5 shows the pseudo-code of FO-LR. The preprocessing step consists in lines 1 and 2. The linear relaxation of M-ILP is solved line 1 using a linear programming solver. V receives the strictly positive variables in the optimal solution of the linear relaxation of M-BIP line 2. The solving step is performed in line 3. It consists in solving the reduced M-BIP formulation, which can be done using any exact algorithm for M-BIP. Finally, the resulting solution X^* is returned on line 4.

Algorithm 5: FO-LR

```
  input  : $\langle A, x, b, l, u \rangle$
  output: $X^* \in \Phi$
1 $X \leftarrow$ Linear relaxation of M-BIP$(A, x, b, l, u)$
2 $V \leftarrow \{x_i : x_i > 0\}$
3 $X^* \leftarrow$ Reduced M-BIP$(A, x, b, l, u, V)$
4 return $X^*$
```

5. Computational experiments

The computational experiments were performed on a 2.4 GHz Intel Xeon E5645 CPU with 32 GB of RAM, running under Linux Ubuntu. The BIP and the linear programming formulations were solved by the ILOG CPLEX solver, version 12.6, with default parameter settings and a single thread. All algorithms were implemented in `C++`, using ILOG Concert Technology, and compiled with GNU `g++` 8.2.0. The parameters of SBA were set accordingly to [22]. In addition, the maximum running time of all algorithms in any of the experiments was set to 3600 seconds.

5.1. The test case

The Weighted Set Covering Problem (WSCP) [32] is one of the most studied combinatorial optimization problems. Its decision version was one of the original 21 problems proved to be NP-Complete by [40]. WSCP applications range from data partitioning [49] to vehicle routing problems [17]. For the reader interested in a deeper knowledge on this problem, we refer to the annotated bibliography [21].

WSCP is defined by a set $\mathcal{N}$ of objects and a set $\mathcal{S}$ of non-empty subsets of $\mathcal{N}$. Furthermore, a weight w_s is associated to each subset $s \in \mathcal{S}$. The objective of WSCP is to find $\mathcal{S}^* \subseteq \mathcal{S}$ such that all objects are covered by at least one subset $s \in \mathcal{S}^*$ and such that the sum of the weights of the sets in S^* is minimum.

Figure 1 illustrates an WSCP instance with $|\mathcal{N}| = 16$ and $|\mathcal{S}| = 5$. The optimal solution $\mathcal{S}^*$ for this instance is $\mathcal{S}^* = \{s_1, s_3, s_5\}$ of cost $w^* = 4 + 7 + 2 = 13$. One can note that subsets s_2 and s_4 are not needed, since the other subsets cover all of their elements.

WSCP can be formulated as follows. The decision variable $x_s \in \{0, 1\}$ states if subset $s \in \mathcal{S}$ belongs to the optimal WSCP solution ($x_s = 1$) or not ($x_s = 0$). Furthermore, let $A_{is} = 1$ if item $i \in \mathcal{N}$ is covered by subset s and $c_{is} = 0$ otherwise. The WSCP formulation consists of the objective function (12) and constraints (13)–(14). The objective function (12) aims at minimizing the weight of subsets $s \in \mathcal{S}$. Inequalities (13) ensure that all items $i \in \mathcal{N}$ are covered by at least one subset $s \in \mathcal{S}$. The domain of variables x_s is defined by constraints (13).

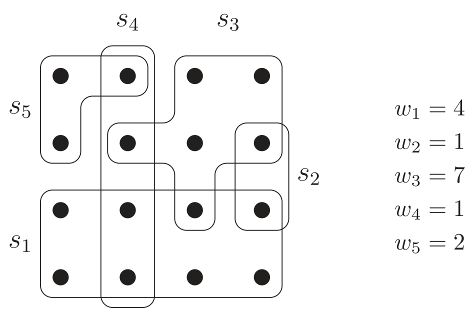

FIGURE 1. An WSCP instance.

$$\min \sum_{s \in \mathcal{S}} w_s x_s \tag{12}$$

$$\text{s.t.} \sum_{s \in \mathcal{S}} A_{is} x_s \geq 1, \qquad \forall i \in \mathcal{N} \tag{13}$$

$$x_s \in \{0,1\}, \qquad \forall s \in \mathcal{S}. \tag{14}$$

5.1.1. The Minmax regret Weighted Set Covering problem under Interval Uncertainties

The Minmax regret Weighted Set Covering problem under Interval Uncertainties (M-WSCP), as defined by [51], is a variant of WSCP under uncertainties. In this problem, the weight associated to each subset $s \in \mathcal{S}$ is uncertain and it is assumed to be in the interval $[l_s, u_s]$, with $0 \leq l_s \leq u_s$. The objective of M-WSCP is to find a solution X^* with the smallest robustness cost, according to the definitions presented in Section 1.

A BIP formulation for M-WSCP was presented in [51]. It is obtained by linearizing $F(Y^{(S^X)}, S^X)$, as explained by [3]. It uses binary decision variables $\{\, x_s \in \{\, 0, 1\,\}\,\} : s \in \mathcal{S}$ and parameter $\{\, A_{is} : i \in \mathcal{N}, s \in \mathcal{S}\,\}$ as for the WSCP. Besides, it also uses an auxiliary variables $\theta \in \mathbb{R}^+$ to compute the cost of the solution in the worst-case scenario, as in the formulation (5)–(9) for M-BIP. The resulting formulation is defined by (15)–(19).

$$\min \sum_{s \in \mathcal{S}} (u_s x_s) - \theta \tag{15}$$

$$\text{s.t.} \sum_{s \in \mathcal{S}} A_{is} x_s \geq 1, \qquad \forall i \in \mathcal{N} \tag{16}$$

$$\theta \leq \sum_{s \in \mathcal{S}} (l_s + (u_s - l_s) x_s)\, y_s, \qquad \forall\, Y \in \Phi \tag{17}$$

$$x_s \in \{0,1\}, \qquad \forall s \in \mathcal{S} \tag{18}$$

$$\theta \geq 0. \tag{19}$$

The objective function (15) aims at minimizing the maximum regret. The constraints (16) ensure that all items $i \in \mathcal{N}$ are covered by at least one subset $s \in \mathcal{S}$. The inequalities (17) enforce the correct value of θ. The constraints (18)–(19) define the domain of the variables x_s and θ, respectively.

5.1.2. Benchmark instances for the M-WSCP

We used two set of benchmark instances in our computational experiments, namely *OR-Library* and *Shunji*. In both sets, the interval $[l_s, u_s]$, for all $s \in \mathcal{S}$, was computed as suggested by [39]. Initially, a random value c_s was obtained from the original instance. Then, we set $l_s = \mathcal{U}\big((1-b)c_s, (1+b)c_s\big)$ and $u_s = \mathcal{U}\big(l_s + 1, (1+b)c_s\big)$, where $b \in \{0.3, 0.6, 0.9\}$ is a parameter that defines the degree of uncertainty of the instance, *i.e.*, the greater is the value of b, the greater is the difference between u_s and l_s, on average.

The set *OR-Library* of instances was retrieved from the OR-Library[3], a collection of test data sets for a variety of Operations Research (OR) problems. It was originally described by [13] and encompass instances for several combinatorial optimization problems. We retrieved the instance sets 4, 5, and 6, which were first used for experiments for the WSCP by [12] and later adapted for the M-WSCP [7, 51]. Set 4 contains 10 instances with $|\mathcal{N}| = 200$ and $|\mathcal{S}| = 1000$, while set 5 contains 10 instances with $|\mathcal{N}| = 200$ and $|\mathcal{S}| = 2000$ and set 6 contains 5 instances with $|\mathcal{N}| = 200$ and $|\mathcal{S}| = 1000$. The weight values range between 1 and 100 for all instances. For each instance obtained from the OR-Library, we generated three instances for the M-WSCP, such that each generated instance use a different value of b. Therefore, there are 75 instances in this set.

The set *Shunji* of instances was generated using the instances generator for WSCP developed by Shunji Umetani[4]. This generator implements the scheme for constructing instances for WSCP proposed by [12] and guarantees that every elements in $\mathcal{N}$ is covered by, at least, two sets in $\mathcal{S}$, and guarantees that every set $s \in \mathcal{S}$ covers at least one element in $\mathcal{N}$. The instances within this set have $|\mathcal{N}| = 1000$ and $|\mathcal{S}| = 2000$. The weight values for all instances range between 1 and 100. Furthermore, it has a parameter $\mathcal{D} \in [0, 1]$ which controls the density of the covering matrix. Given the parameters $\mathcal{D} \in \{0.02, 0.05, 0.10\}$ and $b \in \{0.3, 0.6, 0.9\}$, we generated 5 random instances for each combination of these parameters. Therefore, there are 45 instances in this set.

5.2. Results for the exact algorithms

The first set of experiments aims at evaluating the efficiency of BLD, EB, and BC to solve M-WSCP using the instances in sets *OR-Library* and *Shunji*. The results are reported in Tables 2 and 3. The first and second column respectively report the value of b and the set that the instance belongs. For BLD, the third, fourth, fifth, and sixth columns show respectively (*i*) the number of instances solved to optimality; (*ii*) the average number of cuts inserted by BLD and its standard deviation; (*iii*) the average relative optimality gap for the instances whose optimal solution was not found and the standard deviation of this same value; and (*iv*) the average running time (in seconds) for the instances whose optimal solutions were not found and the standard deviation of this same value. This same data is reported for EB in the seventh, eight, ninth, and tenth columns and for BC in the eleventh, twelfth, thirteenth, and fourteenth columns. When optimal solutions were not found for any of the five instances in a group, the fifth column is filled with a '$-$'. We recall that, for each value of b, sets 4 and 5 have 10 instances, whereas set 6, as so as the instances in *Shunji*, only have 5 instances each.

Regarding the set *OR-Library* of instances, it can be seen from Table 2 that all algorithms found optimal solutions for all evaluated instances. Furthermore, as greater was the value of b, then greater was the average running time of the algorithms. BLD inserted the smallest average number of cuts for all evaluated instance sets, whereas EB inserted the greatest number of cuts, on average. One can see that BC was the fastest among the evaluated algorithms, being able to solve all instance sets within 65 seconds, on average. This smaller running time may be due to the fact that BC do not reinitialize its branching tree, as presented in Section 3.3, while BLD and EB restarts their branching tree at each iteration. Regarding the set *Shunji* of instances, it can be seen from Table 3 that, as denser is the instance, less time the algorithms need to found the optimal solution. EB found a smaller number of optimal solutions than BLD and BC. Furthermore, BLD found a greater number of optimal solutions for instances with $b = 0.3$, while BC found a greater number of optimal solutions for instances with $b = 0.9$. Despite that, the average relative gap achieved by BLD, EB, and BC were up to 0.5%, on average, being very close to the optimal solution. We choose to use BLD for solving the remaining formulation in FO-SBA and FO-LR, as it was able to found a greater number of optimal solutions than EB and BC.

[3]http://people.brunel.ac.uk/~mastjjb/jeb/info.html
[4]https://sites.google.com/site/shunjiumetani/benchmark

TABLE 2. Results for the exact algorithms on *OR-Library* instances.

		BLD				EB				BC			
b	set	opt	cuts	gap (%)	t (s)	opt	cuts	gap (%)	t (s)	opt	cuts	gap (%)	t (s)
	4	10	13.8(90)	0.0(0)	0.1(1)	10	103.4(1495)	0.0(0)	0.1(1)	10	15.8(90)	0.0(0)	1.0(12)
0.3	5	10	13.5(82)	0.0(0)	4.2(47)	10	84.2(898)	0.0(0)	11.1(147)	10	16.3(90)	0.0(0)	2.0(19)
	6	5	6.8(29)	0.0(0)	6.6(30)	5	27.8(145)	0.0(0)	15.2(95)	5	8.2(34)	0.0(0)	6.6(34)
	4	10	41.2(163)	0.0(0)	19.9(125)	10	417.7(4095)	0.0(0)	43.5(469)	10	48.4(195)	0.0(0)	6.7(67)
0.6	5	10	57.3(297)	0.0(0)	49.8(480)	10	1493.4(18648)	0.0(0)	263.9(3657)	10	74.5(423)	0.0(0)	11.1(73)
	6	5	10.6(50)	0.0(0)	10.6(87)	5	51.2(487)	0.0(0)	25.2(333)	5	13.8(40)	0.0(0)	9.2(57)
	4	10	101.7(498)	0.0(0)	230.6(3787)	10	1107.4(14298)	0.0(0)	163.0(2573)	10	128.9(640)	0.0(0)	16.8(180)
0.9	5	10	158.7(792)	0.0(0)	804.9(11254)	10	3501.2(26108)	0.0(0)	1171.1(12005)	10	203.3(1032)	0.0(0)	64.3(628)
	6	5	37.2(94)	0.0(0)	27.8(137)	5	380.8(1717)	0.0(0)	74.4(340)	5	59.8(303)	0.0(0)	13.4(93)

TABLE 3. Results for the exact algorithms on *Shunji* instances.

		BLD				EB				BC			
b	density	opt	cuts	gap (%)	t (s)	opt	cuts	gap (%)	t (s)	opt	cuts	gap (%)	t (s)
	0.02	4	15.4(62)	0.0(0)	2613.2(12183)	1	37.0(156)	0.1(00)	3071.0(0)	1	19.4(68)	0.3(01)	1194.0(0)
0.3	0.05	5	15.4(78)	0.0(0)	1369.4(13391)	3	52.4(286)	0.1(01)	1130.7(7505)	4	20.4(120)	0.3(0)	1174.5(9487)
	0.10	5	15.8(74)	0.0(0)	344.2(2186)	5	121.4(775)	0.0(0)	1195.4(8397)	5	28.8(234)	0.0(0)	430.8(3627)
	0.02	0	16.6(75)	0.1(01)	–	0	19.8(136)	0.2(01)	–	0	16.4(90)	0.5(01)	–
0.6	0.05	1	40.0(236)	0.1(00)	3358.0(0)	0	76.4(523)	0.1(00)	–	1	63.4(467)	0.3(02)	3474.0(0)
	0.10	5	51.0(358)	0.0(0)	791.4(8306)	3	461.6(2866)	0.1(00)	1477.0(11750)	5	75.6(455)	0.0(0)	462.8(4017)
	0.02	0	23.4(84)	0.1(00)	–	0	72.0(559)	0.2(01)	–	0	65.4(402)	0.3(01)	–
0.9	0.05	1	40.6(184)	0.1(00)	816.0(0)	1	323.6(1942)	0.1(00)	2092.0(0)	2	239.0(1144)	0.1(01)	2143.5(19240)
	0.10	3	77.2(192)	0.0(00)	1138.0(6475)	0	706.8(533)	0.1(00)	–	5	138.0(549)	0.0(0)	989.0(6051)

TABLE 4. Results for the FAO heuristics on *OR-Library* instances.

		FO-SBA				FO-LR			
b	set	$\lvert\mathcal{S}'\rvert$	$\frac{\lvert\mathcal{S}'\rvert}{\lvert\mathcal{S}\rvert}$ (%)	t_p (s)	t_t (s)	$\lvert\mathcal{S}'\rvert$	$\frac{\lvert\mathcal{S}'\rvert}{\lvert\mathcal{S}\rvert}$ (%)	t_p (s)	t_t (s)
0.3	4	73.7(85)	7.4	0.6(11)	0.1(03)	68.6(95)	6.9	0.0(00)	0.0(00)
	5	76.4(142)	3.8	1.3(18)	0.3(05)	72.0(107)	3.6	0.1(03)	0.1(03)
	6	50.0(110)	5.0	8.2(33)	0.8(08)	56.2(77)	5.6	1.4(08)	1.5(16)
0.6	4	86.0(82)	8.6	0.7(08)	0.1(03)	75.0(80)	7.5	0.0(00)	0.1(03)
	5	89.0(123)	4.4	1.1(11)	0.2(04)	69.5(70)	3.5	0.0(00)	0.1(03)
	6	52.6(95)	5.3	5.8(22)	0.4(05)	47.8(98)	4.8	0.8(08)	0.6(05)
0.9	4	91.6(84)	9.2	0.0(00)	0.0(00)	71.5(57)	7.1	0.0(00)	0.0(00)
	5	96.2(137)	4.8	0.6(07)	0.2(04)	70.9(80)	3.5	0.0(00)	0.0(00)
	6	53.6(74)	5.4	2.2(16)	0.4(05)	45.6(74)	4.6	0.4(05)	0.0(00)

5.3. Results on the preprocessing step of the fix-and-optimize heuristics

The second set of experiments evaluates how efficient are FO-SBA and FO-LR to fix the value of variables in the M-WSCP formulation defined by the objective function (15) and the constraints in (16)–(19). As the number of binary variables is the same as the cardinality of the set $\mathcal{S}$, we measure as $|\mathcal{S}'|$ how many subsets of $\mathcal{S}$ remain in the instance after the preprocessing step of each heuristic and how long it takes to solve the resulting formulation.

The results of this experiment are presented in Tables 4 and 5 for the *OR-Library* and *Shunji* instances, respectively. The first column reports the value of b, while the second shows the set of the instance (for *OR-Library* instances) or the density value $\mathcal{D}$ (for *Shunji* instances). The third column gives the average and the standard deviation of the number $|\mathcal{S}'|$ of subsets of $\mathcal{S}$ that remained in the instance after the preprocessing step of FO-SBA, while the fourth column presents the average ratio $\mathcal{S}'/\mathcal{S}$, *i.e.*, the proportion of subsets of $\mathcal{S}$ that remained in the instance. The average running times, along with their standard deviations, for the preprocessing and solving steps of FO-SBA are presented in the fifth and sixth columns, respectively. The same data is reported for FO-LR in the last four columns. As pointed out in Section 5.2, the solving step of both FO-SBA and FO-LR employs BLD.

Regarding the *OR-Library* instances, one can see from Table 4 that both heuristics obtained similar results. For both FO-SBA and FO-LR, the greatest average reduction was achieved in instances from set 5, while the smallest average reduction was achieved in instances from set 4. FO-SBA average preprocessing running time never exceeded 9 seconds and was able to fix more than 90% of the variables of the formulation, on average, for all evaluated instances. FO-LR was faster than FO-SBA, such that the maximum running time of its preprocessing step was 1.4 seconds for instances from set 6 with $b = 0.3$. Using this reduced variable set, the BLD average running time never exceeded 1 second for FO-SBA and 1.5 second for FO-LR. In addition, one can also see that the accumulated running times of the preprocessing step and the solving step of FO-SBA and FO-LR, computed as $t_p + t_t$, were significantly smaller than those of the exact algorithms, as reported in Table 2.

Regarding the *Shunji* instances, one can see from Table 5 that, the denser was the instance, then smaller was the running time of FO-SBA and FO-LR, on average. Besides, the heuristics also fixed a greater number of variables in denser instances than on the sparser instances, such that only 2.7% and 2.5% of the original variables were not fixed for the instances with $b = 0.3$ and $\mathcal{D} = 0.10$ using FO-SBA and FO-LR, respectively. The average running time of FO-SBA preprocessing step was up to 1592 seconds on intances with $b = 0.6$ and $\mathcal{D} = 0.05$, while that of FO-LR was up to only 430 seconds on instances with $b = 0.6$ and $\mathcal{D} = 0.05$. However, the solving step of FO-SBA was faster than that of FO-LR for all evaluated instance subsets, being up to 206 seconds on instances with $b = 0.6$ and density of 0.05.

TABLE 5. Results for the FAO heuristics on *Shunji* instances.

		FO-SBA				FO-LR			
b	density	$\vert\mathcal{S}'\vert$	$\frac{\vert\mathcal{S}'\vert}{\vert\mathcal{S}\vert}$ (%)	t_p (s)	t_t (s)	$\vert\mathcal{S}'\vert$	$\frac{\vert\mathcal{S}'\vert}{\vert\mathcal{S}\vert}$ (%)	t_p (s)	t_t (s)
	0.02	150.4(236)	7.5	1235.8(5851)	81.6(428)	174.2(340)	8.7	241.4(1466)	395.0(3652)
0.3	0.05	90.2(68)	4.5	518.2(3993)	38.8(261)	112.2(41)	5.6	72.8(357)	815.6(7061)
	0.10	52.4(84)	2.6	222.0(2129)	12.6(151)	69.4(39)	3.5	42.6(356)	36.0(338)
	0.02	143.6(233)	7.2	1592.2(23451)	33.0(258)	150.0(326)	7.5	430.8(7109)	1160.2(23615)
0.6	0.05	98.6(153)	4.9	453.0(3891)	206.4(3911)	107.8(49)	5.4	80.6(470)	1203.0(15034)
	0.10	53.0(73)	2.6	64.2(407)	3.0(07)	61.6(32)	3.1	8.6(48)	15.8(137)
	0.02	184.8(175)	9.2	434.4(3845)	81.6(1130)	166.0(130)	8.3	109.6(1314)	275.2(3688)
0.9	0.05	99.8(117)	5.0	154.2(1025)	6.4(22)	88.0(114)	4.4	26.8(146)	13.0(64)
	0.10	53.2(51)	2.7	29.8(104)	1.6(13)	50.8(68)	2.5	5.0(23)	1.4(05)

TABLE 6. Results for the heuristics on *OR-Library* instances.

		SBA		LPH		FO-SBA		FO-LR	
b	set	dev (%)	t (s)	dev (%)	t (s)	dev (%)	t (s)	dev (%)	t (s)
0.3	4	4.9(92)	0.6(11)	8.6(97)	0.2(04)	4.6(82)	0.7(11)	6.5(100)	0.0(00)
	5	9.0(116)	1.3(18)	13.7(126)	0.1(03)	9.0(116)	1.6(16)	10.1(108)	0.2(04)
	6	11.9(76)	8.2(33)	54.4(290)	0.4(05)	9.4(68)	9.0(38)	9.4(68)	3.0(23)
0.6	4	10.4(170)	0.7(08)	10.2(148)	0.3(05)	9.9(172)	0.8(10)	10.7(169)	0.1(03)
	5	7.0(63)	1.1(11)	6.6(68)	0.8(06)	7.0(63)	1.3(09)	7.6(78)	0.1(03)
	6	2.8(62)	5.8(22)	15.7(24)	0.6(05)	2.8(62)	6.2(22)	5.1(57)	1.4(11)
0.9	4	9.0(113)	0.1(00)	7.2(116)	0.3(05)	9.0(114)	0.0(00)	9.7(111)	0.0(00)
	5	2.9(28)	0.6(07)	1.7(26)	0.7(07)	2.9(28)	0.8(06)	3.8(32)	0.0(00)
	6	2.7(19)	2.2(16)	2.8(17)	1.0(00)	2.4(22)	2.6(15)	3.5(29)	0.4(05)

5.4. Comparison of the heuristics

The last set of experiments compares the results of the proposed heuristics with those of the best known heuristics for M-WSCP. Our heuristics are constrated with each other and with two other heuristics for the M-WSCP: the Scenario-based Algorithm (SBA) [22] and the Linear Programming Heuristic (LPH) [7]. The results reported in Tables 6 and 7 for the *OR-Library* and *Shunji* instances, respectively. The first column reports the value of b, while the second shows the set of the instance (for *OR-Library* instances) or the density value $\mathcal{D}$ (for *Shunji* instances). The third and fourth columns report the results of SBA. Let $\bar{X}$ the best known solution for each instance (found by any of the exact or heuristic algorithms used in our experiments), the third column gives the *average relative deviation* $\frac{Z(SBA)-Z(\bar{X})}{Z(\bar{X})}$, where $Z(SBA)$ is the robustness cost of the solution obtained by SBA, along with the standard deviation of this same metric. The fourth column presents the average and the standard deviation of the running time of SBA. The same data is reported for LPH in the fifth and sixth columns, for FO-SBA in the seventh and eighth columns, and for FO-LR in the ninth and tenth columns.

Regarding the *OR-Library* instances, one can see from Table 6 that the average running time of the heuristics never exceeded 9 seconds, which was the case of FO-SBA for instances from set 6 with $b = 0.3$. LPH was the fastest among the evaluated heuristics. However, it also found the greater average relative deviations among the evaluated heuristics, being up to 54.4% for instances from set 6 with $b = 0.3$. The best results for all instances were obtained by FO-SBA, such that it achieved a maximum average relative deviation of 9.9% for instances from set 4 with $b = 0.6$. Besides, one can observe that FO-SBA average running time never exceeds 9 seconds, and that it was able to improve the average relative deviations of SBA for 4 out of the 9 subset of instances evaluated. These results indicate that FO-SBA was able to fastly compute near-optimal solutions, since the optimal solution for all *OR-Library* instances were given by BLD, EB, and BC, as shown in Table 2.

Regarding the *Shunji* instances, one can see from Table 7 that the results were very similar to those obtained for the *OR-Library* instances. LPH obtained the greatest average relative deviations for all subset of instances. On the other hand, the FO-SBA solutions had the smallest average relative deviation among the evaluated heuristics for all instance subsets, except for instances with $b = 0.3$ and $\mathcal{D} = 0.05$. One can observe that FO-SBA was able to improve the results of SBA for instances with $b = 0.3$ and $\mathcal{D} = 0.10$ and for instances with $b = 9$ and $\mathcal{D} = 0.10$. Furthermore, FO-LR obtained competitive results, being faster than FO-SBA for 7 out of the 9 subset of instances evaluated and computing solutions with an average relative deviation at least as good as those of FO-SBA for 3 out of the 9 subset of instances evaluated.

The results from Tables 6 and 7 point out to the fact that FO-SBA obtained the best results among the evaluated heuristics. To test this observation for *OR-Library* and *Shunji* instances, we analyzed our experimental data following the statistical procedure of [36], which is composed of three steps. The first step verifies the non-normality of the relative deviations of SBA, LPH, FO-SBA, and FO-LR. Next, the second step evaluates whether

TABLE 7. Results for the heuristics on *Shunji* instances.

		SBA		LPH		FO-SBA		FO-LR	
b	density	dev (%)	t (s)	dev (%)	t (s)	dev (%)	t (s)	dev (%)	t (s)
	0.02	4.9(33)	1235.8(5851)	73.1(230)	48.2(532)	4.9(33)	1317.4(5718)	4.9(33)	636.4(4036)
0.3	0.05	6.9(76)	518.2(3393)	79.4(245)	39.8(331)	6.9(76)	557.0(4217)	3.3(75)	888.4(7291)
	0.10	7.2(51)	222.0(2129)	71.1(299)	62.0(582)	6.0(61)	234.6(2097)	6.0(61)	78.6(529)
	0.02	0.9(10)	1592.2(23451)	25.8(56)	90.6(686)	0.9(10)	1625.2(23277)	2.1(25)	1591.0(30722)
0.6	0.05	2.6(28)	453.0(3891)	34.6(125)	335.4(5690)	2.6(28)	659.4(7704)	3.8(23)	1283.6(15341)
	0.10	4.9(35)	64.2(407)	21.7(77)	25.8(236)	4.9(35)	67.2(413)	6.6(57)	24.4(182)
	0.02	0.0(00)	434.4(3845)	6.8(19)	184.8(2024)	0.0(00)	516.0(4934)	2.4(25)	384.8(4998)
0.9	0.05	0.7(10)	154.2(1025)	8.6(35)	166.8(1205)	0.7(10)	160.6(1032)	1.4(09)	39.8(188)
	0.10	2.2(23)	29.8(104)	5.0(21)	29.2(136)	1.7(25)	31.4(105)	2.8(27)	6.4(23)

TABLE 8. p-values obtained by the Nemenyi's test on *OR-Library* and *Shunji* instances.

	OR-Library			Shunji		
	SBA	LPH	FO-SBA	SBA	LPH	FO-SBA
LPH	0.52	–	–	0.01	–	–
FO-SBA	0.93	0.20	–	1.00	0.01	–
FO-LR	0.14	0.86	0.03	0.74	0.01	0.61

there is a statistical significant difference among the four heuristics. Then, in case such a difference exists, the third step verifies whether the relative deviations of the heuristics are significantly different among them. These steps are detailed below. They assume a significance level $\alpha = 0.05$, *i.e.*, the null hypothesis is rejected if the p-value is smaller than 0.05.

In the first step, we applied a Shapiro-Wilk test of normality [52] to verify whether the relative deviations of SBA, LPH, FO-SBA and FO-LR follow a normal distribution. With a p-value of 0.001, the test indicated that the data of the three heuristics does not follow a normal distribution. Thus, a non-parametric test is used in the next step.

In the second step, we applied the Friedman's test [34] to verify whether there is a statistically significant difference between at least two of the evaluated heuristics. The null hypothesis was that SBA, LPH, FO-SBA and FO-LR have the same relative deviation, on average. The data were ranked according to [18]. With a p-value of 0.002, the test rejected the null hypothesis for both *OR-Library* and *Shunji* instances. Therefore, there is indeed a significant difference in the relative deviations of SBA, LPH, FO-SBA and FO-LR.

In the third step, we applied a non-parametric two-tailed Nemenyi's post-hoc test, also known as the Nemenyi-DamicoWolfeDunn post-hoc test [47], which compares the results of multiple algorithms. This test evaluated the pair of hypothesis

$$\begin{cases} H_0 : \mu_i \geq \mu_j \\ H_1 : \mu_i \neq \mu_j \end{cases}, \qquad \forall(\mu_i, \mu_j) \in M,$$

where $M = \{\mu_{\text{sba}}, \mu_{\text{lph}}, \mu_{\text{fo-sba}}, \mu_{\text{fo-lr}}\}$, such that $\mu_{\text{sba}}, \mu_{\text{lph}}, \mu_{\text{fo-sba}}$, and $\mu_{\text{fo-lr}}$ are, respectively, the average of the rankings obtained by SBA, LPH, FO-SBA, and FO-LR in the second step. The null hypothesis (H_0) states that the average ranking of μ_i and μ_j was not significantly different among them, thus implying that the results of one of the evaluated heuristics is not significantly better than those of the other. On the other hand, the alternative hypothesis (H_1) implies that indeed the results of μ_i is significantly different than those of μ_j.

Table 8 presents the results of the Nemenyi's test for *OR-Library* and *Shunji* instances. Each cell of this table displays the p-value obtained by comparing the heuristics displayed on the top of the column and on the beginning of the row.

Regarding the *OR-Library* instances, one can see from Table 8 that only FO-SBA and FO-LR significantly differ among them, with a p-value of 0.03. For the remaining of the comparisons, the test could not reject the null hypothesis and, thus, we cannot assume that the average relative deviation of the heuristics significantly differ among them. Thus, we conclude that SBA, LPH, and FO-SBA results do not significantly differ among them. For this set of instances, the recommended heuristic is LPH, since it had a smaller running time in comparison to SBA and FO-SBA, as shown in Table 6.

Regarding the *Shunji* instances, one can see from Table 8 that LPH was the worst heuristic for these instances, as the Nemenyi's test returned a p-value of 0.01 for the comparison of LPH with all other heuristics. The comparison of SBA and FO-SBA returned a p-value of 1.00, while the comparisons of FO-LR with SBA and FO-SBA returned a p-value of 0.74 nad 0.61, respectively. Thus, the test could not reject the null hypothesis for these comparisons and we cannot assume that the average relative deviations of SBA and FO-SBA significantly differ among them. Additionally, we also cannot assume that the average relative deviations of FO-LR, SBA, and FO-SBA significantly differ among them. Therefore, for this set of instances, the recommended heuristic is

FO-LR, since its average running time was smaller than those of SBA and FO-SBA for 7 out of the 9 subset of instances, as shown in Table 7.

6. Concluding remarks

This paper studied the Minmax regret Binary Integer Programming problem under interval uncertainty (M-BIP). We demonstrated that he decision version of M-BIP is Σ_2^p-complete, as it is a generalization of the decision version of the Minmax regret Binary Knapsack problem under interval uncertainty, which is well-known to be Σ_2^p-complete. Furthermore, we proposed three exact and two fix-and-optimize heuristics to solve M-BIP. Computational experiments evaluated the Minmax regret Weighted Set Covering problem under Interval Uncertainties (M-WSCP) as a test case. The results indicated that the Benders-like Decomposition outperformed the other exact algorithms, while the proposed fix-and-optimize heuristics were competitive with ad-hoc heuristics for the M-WSCP.

The computation results showed that, although FO is a general heuristic for minmax regret optimization problems under interval uncertainty, it is competitive with the best heuristics in the literature of M-WSCP. These results encourage the use of fix-and-optimize heuristics for minmax regret optimization problems under interval uncertainty, as both heuristics can be easily applied to any problem of this class. Therefore, future works may apply FO-SBA or FO-LR to other of these problems, such as the Minmax Knapsack Problem Problem under interval uncertainty [35], the Minmax regret Restricted Shortest Path problem under interval uncertainty [5], the Minmax regret Spanning Arborescence problem under interval uncertainty [25], or the Minmax regret Weighted Independent Set problem under interval uncertainty [41]. Furthermore, future works can evaluate fix-and-optimize heuristics for minmax regret optimization problems under different uncertainty sets, such as the polyhedral uncertainty set [16] or the ellipsoidal uncertainty set [15]. Additionally, one can investigate the use of different algorithms in the preprocessing step of fix-and-optimize heuristics and its applications to other problems.

Acknowledgements. This study was financed in part by the Coordenaçã de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, the Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq), and the Fundação de Amparo à Pesquisa do Estado de Minas Gerais - Brasil (FAPEMIG).

Conflict of interest/Competing interest. The authors declare that they have no conflicts/competing interests.

Availability of data and material. Data and materials can be obtained upon request to the authors.

Code availability. Code can be obtained upon request to the authors.

References

[1] H. Aissi, C. Bazgan and D. Vanderpooten, Complexity of the min–max and min–max regret assignment problems. *Oper. Res. Lett.* **33** (2005) 634–640.

[2] H. Aissi, C. Bazgan and D. Vanderpooten, Complexity of the min-max (regret) versions of cut problems, in *International Symposium on Algorithms and Computation*. Springer (2005) 789–798.

[3] H. Aissi, C. Bazgan and D. Vanderpooten, Min-max and min-max regret versions of combinatorial optimization problems: A survey. *Eur. J. Oper. Res.* **197** (2009) 427–438.

[4] R. Anand, D. Aggarwal and V. Kumar, A comparative analysis of optimization solvers. *J. Stat. Manag. Syst.* **20** (2017) 623–635.

[5] L. Assunção, T.F. de Noronha, A.C. Santos and R.C. de Andrade, A linear programming based heuristic for robust optimization problems: a case study on solving the restricted robust shortest path problem, in *Matheuristics 2014–5th International Workshop on Model-Based Metaheuristics*. Hambourg, Alemagne (2014) 8.

[6] L. Assunção, A.C. Santos, T.F. Noronha and R. Andrade, On the finite optimal convergence of logic-based benders' decomposition in solving 0–1 min-max regret optimization problems with interval costs, in *International Symposium on Combinatorial Optimization*. Springer (2016) 1–12.

[7] L. Assunção, T.F. Noronha, A.C. Santos and R. Andrade, A linear programming based heuristic framework for min-max regret combinatorial optimization problems with interval costs. *Comput. Oper. Res.* **81** (2017) 51–66.

[8] I. Averbakh and V. Lebedev, Interval data minmax regret network optimization problems. *Discrete Appl. Math.* **138** (2004) 289–301.

[9] I. Averbakh and V. Lebedev, On the complexity of minmax regret linear programming. *Eur. J. Oper. Res.* **160** (2005) 227–231.

[10] I. Averbakh, On the complexity of a class of combinatorial optimization problems with uncertainty. *Math. Program.* **90** (2001) 263–272.

[11] I. Averbakh, Complexity of robust single facility location problems on networks with uncertain edge lengths. *Discrete Appl. Math.* **127** (2003) 505–522.

[12] E. Balas and A. Ho, Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study, in *Combinatorial Optimization*. Springer (1980) 37–60.

[13] J.E. Beasley, Or-library: distributing test problems by electronic mail. *J. Oper. Res. Soc.* **41** (1990) 1069–1072.

[14] J.F. Benders, Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.* **4** (1962) 238–252.

[15] A. Ben-Tal and A. Nemirovski, Robust solutions of linear programming problems contaminated with uncertain data. *Math. Program.* **88** (2000) 411–424.

[16] D. Bertsimas and M. Sim, The price of robustness. *Oper. Res.* **52** (2004) 35–53.

[17] E. Buluc, M. Peker, B.Y. Kara and M. Dora, Covering vehicle routing problem: application for mobile child friendly spaces for refugees. *OR Spectr.* (2021) 1–24.

[18] I.A. Carvalho, On the statistical evaluation of algorithmic's computational experimentation with infeasible solutions. *Inf. Process. Lett.* **143** (2019) 24–27.

[19] I.A. Carvalho, T.F. Noronha, C. Duhamel and L.F.M. Vieira, A scenario based heuristic for the robust shortest path tree problem, in *VIII Conference on Manufacturing, Modelling, Management & Control* (2016) 443–448.

[20] I.A. Carvalho, T.F. Noronha, C. Duhamel, L.F. Vieira and V.F.D. Santos, A fix-and-optimize heuristic for the minmax regret shortest path arborescence problem under interval uncertainty. *Int. Trans. Oper. Res.* (2021).

[21] S. Ceria, P. Nobili and A. Sassano, Set covering problem, in *Annotated Bibliographies in Combinatorial Optimization*, Edited by S. Martello, F. Maffioli. Wiley (1997) 415–428.

[22] A.A. Coco, A.C. Santos and T.F. Noronha, Senario-based heuristics with path-relinking for the robust set covering problem, in *Proceedings of the XI Metaheuristics International Conference (MIC)* (2015) 1–6.

[23] A.A. Coco, A.C. Santos and T.F. Noronha, Coupling scenario-based heuristics to exact methods for the robust weighted set covering problem with interval data, in *VIII Conference on Manufacturing, Modelling, Management & Control* (2016) 455–460.

[24] A.A. Coco, A.C. Santos and T.F. Noronha, Formulation and algorithms for the robust maximal covering location problem. *Electron. Notes Discrete Math.* **64** (2018) 145–154.

[25] E. Conde, A branch and bound algorithm for the minimax regret spanning arborescence. *J. Glob. Optim.* **37** (2007) 467–480.

[26] W. Cook, M. Hartmann, R. Kannan and C. McDiarmid, On integer points in polyhedra. Combinatorica **12** (1992) 27–37.

[27] N.K. Dastjerd and K. Ertogral, A fix-and-optimize heuristic for the integrated fleet sizing and replenishment planning problem with predetermined delivery frequencies. *Comput. Ind. Eng.* **127** (2019) 778–787.

[28] D. Degel and P. Lutter, A Robust Formulation of the Uncertain Set Covering Problem. Working paper, Bochum (2013).

[29] V.G. Deineko and G.J. Woeginger, Pinpointing the complexity of the interval min–max regret knapsack problem. *Discrete Optim.* **7** (2010) 191–196.

[30] Á.P. Dorneles, O.C. de Araújo and L.S. Buriol, A fix-and-optimize heuristic for the high school timetabling problem. *Comput. Oper. Res.* **52** (2014) 29–38.

[31] A. Dolgui and S. Kovalev, Min–max and min–max (relative) regret approaches to representatives selection problem. *4OR* **10** (2012) 181–192.

[32] J. Edmonds, Covers and packings in a family of sets. *Bull. Am. Math. Soc.* **68** (1962) 494–499.

[33] M. Fischetti, D. Salvagnin and A. Zanette, A note on the selection of benders' cuts. *Math. Program.* **124** (2010) 175–182.

[34] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *J. Am. Stat. Assoc.* **32** (1937) 675–701.

[35] F. Furini, M. Iori, S. Martello and M. Yagiura, Heuristic and exact algorithms for the interval min-max regret knapsack problem. *INFORMS J. Comput.* **27** (2015) 392–405.

[36] S. Garcia and F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *J. Mach. Learn. Res.* **9** (2008) 2677–2694.

[37] V. Gintner, N. Kliewer and L. Suhl, Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice. *OR Spectr.* **27** (2005) 507–523.

[38] P. Guo, W. Cheng, Y. Wang and N. Boysen, Gantry crane scheduling in intermodal rail-road container terminals. *Int. J. Prod. Res.* **56** (2018) 5419–5436.

[39] O.E. Karaşan, M.C. Pinar and H. Yaman, *The robust shortest path problem with interval data*, Tech. Rep., Bilkent University, Department of Industrial Engineering (2001).

[40] R.M. Karp, Reducibility among combinatorial problems, in *Complexity of computer computations*. Springer (1972) 85–103.

[41] A. Kasperski and P. Zieliński, Complexity of the robust weighted independent set problems on interval graphs. *Optim. Lett.* **9** (2015) 427–436.

[42] A. Kasperski and P. Zieliński, An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Inf. Process. Lett.* **97** (2006) 177–180.

[43] A. Kasperski and P. Zieliński, Robust discrete optimization under discrete and interval uncertainty: A survey, in *Robustness Analysis in Decision Aiding, Optimization, and Analytics*. Springer (2016) 113–143.

[44] P. Kouvelis and G. Yu, Robust discrete optimization and its applications, In Vol. 14 of *Nonconvex Optimization and its Applications*. Springer (1997).

[45] V. Lebedev and I. Averbakh, Complexity of minimizing the total flow time with interval data and minmax regret criterion. *Discrete Appl. Math.* **154** (2006) 2167–2177.

[46] R. Montemanni, J. Barta, M. Mastrolilli and L.M. Gambardella, The robust traveling salesman problem with interval data. *Transp. Sci.* **41** (2007) 366–381.

[47] P. Nemenyi, Distribution-free multiple comparisons. *Biometrics* **18** (1962) 263.

[48] J.C. Lang and Z.-J.M. Shen, Fix-and-optimize heuristics for capacitated lot-sizing with sequence-dependent setups and substitutions. *Eur. J. Oper. Res.* **214** (2011) 595–605.

[49] X. Li, Q. Lu, Y. Dong and D. Tao, Sce: A manifold regularized set-covering method for data partitioning. *IEEE Trans. Neural Netw. Learn. Syst.* **29** (2017) 1760–1773.

[50] C. Papadimitriou, Computational Complexity, Theoretical computer science. Addison-Wesley (1994).

[51] J. Pereira and I. Averbakh, The robust set covering problem with interval data. *Ann. Oper. Res.* **207** (2013) 1–19.

[52] S.S. Shapiro and M.B. Wilk, An analysis of variance test for normality (complete samples). *Biometrika* **52** (1965) 591–611.

[53] L.J. Stockmeyer, The polynomial-time hierarchy. *Theor. Comput. Sci.* **3** (1976) 1–22.

[54] A.M. Turhan and B. Bilgen, Mixed integer programming based heuristics for the patient admission scheduling problem. *Comput. Oper. Res.* **80** (2017) 38–49.

[55] H.-I. Yu, T.-C. Lin and B.-F. Wang, Improved algorithms for the minmax-regret 1-center and 1-median problems. *ACM Trans. Algorithms (TALG)* **4** (2008) 36.