Solving bilevel combinatorial optimization as bilinear min-max optimization via a branch-and-cut algorithm

Artur Alves Pessoa

Production Engineering Department - Fluminense Federal University, Rua Passo da Pátria 156, 24210-240, Niterói, RJ, Brasil

Michael Poss

UMR CNRS 7253 Heudiasyc, Université de Technologie de Compiègne, Centre de Recherches de Royallieu, 60200 Compiègne, France

Marcos Costa Roboredo

Production Engineering Department - Fluminense Federal University, Rua Passo da Pátria 156, 24210-240, Niterói, RJ, Brasil

Abstract

In this paper, we propose a generic branch-and-cut algorithm for a special class of bi-level combinatorial optimization problems. Namely, we study such problems that can be reformulated as bilinear min-max combinatorial optimization problems. We show that the reformulation can be efficiently solved by a branch-and-cut algorithm whose cuts represent the inner maximization feasibility set. The algorithm generalizes a method developed recently by Roboredo and Pessoa for two particular bi-level problems. In addition, we apply the algorithm on the r-Interdiction Median Problem with Fortification (RIMF). The RIMF considers sets of facilities and customers where each cus-

Email addresses: artur@producao.uff.br (Artur Alves Pessoa), michael.poss@hds.utc.fr (Michael Poss), mcr.marcos@yahoo.com.br (Marcos Costa Roboredo)

tomer is served by the nearest facility unless the facility is interdicted and not fortified. The objective is to minimize the total weighted distance by fortifying q facilities knowing that r facilities will be interdicted. Our numerical results show that our method is more suitable on large instances than the best exact method found in literature.
Keywords: Integer programming, Min-max problems, Bilevel Problems, (r, p)-centroid problem

---

1. Introduction

Bilevel programming can model optimization problems where two non-cooperative decison makers (competitors) choose their decisions (strategies) in a sequential way. Each competitor aims at optimizing her own objective function taking into account the strategy of the other. The competitor that chooses the strategy first is called leader while the second one is called follower. Each decision maker makes her choices knowing that the other decision maker will react optimally, choosing among a set of predetermined strategies. The objective function of the leader and follower are called, respectively, first level objective function and second level objective function. The decision variables of the leader and the follower are called, respectively, decision variables of the first level and decision variables of the second level. Bilevel programming is a large and active field of research and we refer the interested reader to the surveys presented in [1, 2, 3, 4], and the references therein.

We deal in this paper with a class of Bilevel Combinatorial Optimization Problems (BCOP) that can be reformulated as bilinear min-max optimization

problems. The min-max optimization are further reformulated as combinatorial optimization problems with a very large number of constraints. In our approach, the constraints are generated on demand within a branch-and-cut algorithm.

The contributions of this paper are two-fold. First, we formalize a generic branch-and-cut algorithm that can be applied to a large class of BCOP. Our approach encompasses the algorithms that had been used previously in [5] for the (r, p)-centroid problem proposed by [6]. While [5] could optimally solve many open instances for the first time, the general framework behind the algorithms was not fully understood in [5]. Second, we apply the framework to the r-Interdiction Median Problem with Fortification [7] and our results show that our algorithm is more efficient than the best from the literature [8] on large instances.

This paper is divided as follows. In section 2 we present our framework. In Section 3, we describe two applications of our methodology. We describe in Sections 3.1 and 3.2 the (r, p)-centroid problem and the r-Interdiction Median Problem with Fortification (RIMF), respectively. We provide natural bilevel formulations for these problems and show how they can be reformulated as bilinear min-max optimization problems. In section 4 we show a comparison between our technique and the best exact one found for the RIMF and present statistics of our approach for large instances. Finally in section 5 we present the conclusions and some possibilities for future research.

2. The Framework

In this section we describe formally our approach to solve BCOP as bilinear min-max combinatorial optimization problems.

First, we define a BCOP as follows. Let $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$ be first-level and second-level vectors of 0-1 variables. A BCOP can be generally formulated as:

$$\min_{x \in \mathcal{X}(y)} \quad f(x,y) \tag{1}$$

$$\text{s.t.} \quad \max_{y \in \mathcal{Y}(x)} \quad g(x,y), \tag{2}$$

where $f(\cdot)$ and $g(\cdot)$ are the leader and follower objective functions, respectively, and $\mathcal{X}(y) \subseteq \{0,1\}^n$ and $\mathcal{Y}(x) \subseteq \{0,1\}^m$ are the set of leader's feasible solutions given a follower's solution y and the set of follower's feasible solutions given a leader's solution x, respectively.

In this paper, we focus on solving through a branch-and-cut approach BCOP that can be reformulated as follows. Let P and P' be two polyhedrons and define the feasibility sets of the leader and of the follower as $X = P \cap \{0,1\}^n$ and $Y = P' \cap \{0,1\}^m$, respectively. Let $C \in \mathbb{R}^{m \times n}$ be a cost matrix, $D \in \mathbb{R}^n$ and $E \in \mathbb{R}^m$ be cost vectors. We study herein BCOP that can be reformulated as:

$$\min_{x \in X} \max_{y \in Y} y^T C x + Dx + Ey. \tag{3}$$

Note that we need to eliminate both the dependences of $\mathcal{X}$ on y and of $\mathcal{Y}$ on x, modeling all interaction between leader and follower solutions through

the quadratic terms of $y^T Cx$. Although this reformulation seems to be restrictive, the two examples of BCOP shown in the next subsection indicate that our approach is quite general. The first one because it is a $\sum_2^{\mathrm{p}}$-hard problem, and the second one because the reformulation requires the insertion of several auxiliary variables, illustrating how to lead the mathematical model to the desired form.

Then, we solve min-max problem (3) by replacing the inner maximization with a (finite) set of linear inequalities:

$$\min_{x \in X} \; z \tag{4}$$

$$\text{s.t. } z \geq y^T Cx + Dx + Ey, \; \forall y \in Y. \tag{5}$$

Because $Y \subset \{0,1\}^m$, the number of constraints (5) is finite. Nevertheless, this number is likely to be exponential in n and m so that efficient approaches for the above problem should rather generate the constraints on demand in a branch-and-cut algorithm. Namely, given a relaxed leader solution $\overline{x} \in X$, the separation problem associated to (5) can be cast as:

$$\max_{y \in Y} \; y^T C\overline{x} + D\overline{x} + Ey. \tag{6}$$

Our exact algorithm is built on the top of the branch-and-cut algorithm that solves the model given by (4)-(5) using commercial solvers. Constraints (5) are added through a cut callback by solving the exact problem (6). We point out that the bilinear form of constraints (5) results in a relatively strong continuous relaxation for the outer minimization of (3), which an important

advantage of our approach when compared to other techniques for solving bilevel problems. As a result it is usually worth separating cuts over fractional solutions along the branch-and-bound tree until the gap between the current relaxation and the best integer solution found so far drops below a given threshold. Heuristics can also be used for the separation problem in order to efficiently find some violated cuts and thus avoiding solving exactly problem (6).

3. Applications

We present in this section two examples of BCOP studied in the literature and show how they can be reformulated as particular cases of problem (3).

3.1. The discrete (r, p)-centroid problem

The discrete (r, p)-centroid problem is formally defined as follows: Consider two nooncooperative firms (leader and follower). The leader has to place p facilities on an arena knowing that the follower will react by placing r facilities. The arena is a complete bipartite graph $G = (V, E)$ where each vertex $v \in V$ is either a customer or an applicant facility of the leader or the follower. As a result, V can be partitioned into two disjoint subsets I and J, where I is the set of applicant facilities, and J is the set of customers. The edge set E of G has an edge $e = (i, j)$ for each $i \in I$ and $j \in J$, with an associated distance d_{ij}. Each customer's demand w_j is totally served by the firm which places the nearest facility. Ties are broken in favor of the leader's facilities, and ties between facilities of the same firm are broken arbitrarily. Each firm aims at serving the maximum demand as possible. The

discrete (r, p)-centroid problem consists of deciding where the leader places its p facilities.

It has been shown in [9] that the problem is $\sum_2^{\mathrm{p}}$-hard. Hence, that problem turns out to be harder than any optimization problem whose decision version is in NP. The hardness of that problem comes from the fact that evaluating a single leader's strategy requires to solve an NP-hard problem to optimize the follower's strategy. The Hardness of the (r, p)-centroid problem has spurred some heuristics for this problem, see [10], [11] and the references therein. Exact methods have also been proposed [12, 13, 5].

The authors of [5] proposed the first MIP formulation for the discrete (r, p)-centroid problem having a polynomial number of variables although an exponential number of constraints are required. Based on that formulation, the authors proposed a branch-and-cut algorithm where cuts are separated using an auxiliary MIP formulation. Their paper also reported experiments that show that the new algorithm clearly outperformed the previous known exact methods for the problem. Next, we derive a suitable bilinear min-max formulation for the problem such that the corresponding generic branch-and-cut algorithm described in Section 2 turns out to be the basic algorithm proposed by [5].

The reformulation below is such that the variables controlled by the leader do not appear in the follower's constraints and vice-versa. For that, we use variables x, y, u and v. For each $i \in I$, binary variable x_i is equal to 1 if and only if the leader places the facility i and binary variable y_i is equal to 1 if and only if the follower places the facility i. Moreover, for each $i \in I$ and $j \in J$, the binary variable u_{ij} is equal to 1 if and only if i is the facility

placed by the leader closest to the customer j, and the binary variable v_{ij} is equal to 1 if and only if i is the facility placed by the follower closest to the customer j. The complete bilinear min-max formulation for the discrete (r, p)-centroid problem is below.

$$\min_{x,s} \quad \sum_{j\in J}\sum_{k\in I}\left(w_j \sum_{i\in I|d_{ij}>d_{kj}} u_{ij}\right) v_{kj} \tag{7}$$

$$\text{s.t.} \quad \sum_{i\in I} x_i = p \tag{8}$$

$$u_{ij} \le x_i, \quad \forall j \in J, \forall i \in I \tag{9}$$

$$\sum_{i\in I} u_{ij} = 1, \quad \forall j \in J \tag{10}$$

$$x_i, u_{ij} \in \{0,1\}, \quad \forall i \in I, \forall j \in J \tag{11}$$

$$\max_{y,t} \quad \sum_{j\in J}\sum_{k\in I}\left(w_j \sum_{i\in I|d_{ij}>d_{kj}} u_{ij}\right) v_{kj} \tag{12}$$

$$\text{s.t.} \quad \sum_{i\in I} y_i = r \tag{13}$$

$$v_{ij} \le y_i, \quad \forall j \in J, \forall i \in I \tag{14}$$

$$\sum_{i\in I} v_{ij} = 1, \quad \forall j \in J \tag{15}$$

$$y_i, v_{ij} \in \{0,1\}, \quad \forall i \in I, \forall j \in J \tag{16}$$

The first level objective function (7) aims at minimizing the total demand served by the follower. Constraint (8) ensures that p facilities must be placed by the leader. Constraints (9) ensure the consistency between the variables x and u. Constraints (10) ensure that for each customer j, there is exactly one facility closest to j placed by the leader. The second level constraint

(13) ensures that r facilities must be placed by the follower. Constraints (14) ensure the consistency between the variables v and y. Constraints (15) ensure that for each customer j, there is exactly one facility closest to j placed by the follower.

We obtain problem (3) by defining X and Y through (8)–(11) and (13)–(16), respectively, setting $D = E = 0$, and choosing the coefficients of C according to (7).

3.2. The r-Interdiction Median Problem with Fortification

The environment of the RIMF is composed of n customers and p facilities where the demand w_j of each customer j is served by the closest facility. The distance between a facility $i \in I$ and a customer $j \in J$ is denoted by d_{ij}, yielding a serving cost of $c_{ij} = w_j d_{ij}$. If a facility is interdicted due to for example an intentional attack or natural disaster then the customers served by this facility skip to the cheapest facility not interdicted. When it happens the system's performance decreases. A way to avoid part of this decrease in the performance is to fortify the facilities. If a facility is fortified and interdicted at the same time then customers can be served by this facility. The problem consists of choosing a group of q facilities to fortify knowing that r facilities will be interdicted. The r facilities to be interdicted are chosen in order to damage the cost of the system performance as much as possible. The RIMF can be seen as a BCOP where the first level decision chooses the group of q facilities to fortify and the second one chooses the group of r facilities to interdict.

The literature on RIMF is rather recent. The RIMF was proposed by [7], where the authors proposed a mixed-integer formulation with an exponential

number of constraints and variables. That formulation could optimally solve instances with up to $p = 20$, $q = 10$ and $r = 4$. Scaparra and Church [14] proposed an approach where the size of the model is significantly reduced. The main weakness of that approach is that it requires a complete enumeration of all possible ways of interdicting r of the p facilities. The authors optimally solved instances with up to $p = 30$, $q = 7$ e $r = 7$. In [8], the authors proposed a bilevel formulation and a specialized tree search algorithm where it is necessary to solve at most $\frac{r^{q+1}-1}{(r-1)}$ second level subproblems. That tree search algorithm could optimally solve instances with up to $p = 60$, $q = 12$, and $r = 5$.

Variants of the RIMF have also been considered. Liberatore et al. [15] considered uncertainty on the number of facilities to be interdicted. Aksen et al. [16] dealt with a variant of the RIMF where the number of facilities to be fortified is under a budget constraint. Besides, the authors consider an unitary cost when the closest facility of a customer is interdicted because in this case it is necessary to expand the capacity of another facility.

We describe next a natural formulation for the RIMF. This formulation uses three sets of variables x, y, and s. For each $i \in I$, binary variable x_i is equal to 1 if and only if the leader chooses to fortify facility i and binary variable y_i is equal to 1 if and only if the follower decides to interdict facility i. Then, for each $i \in I$ and $j \in J$, variable s_{ij} is equal to 1 if and only if customer j uses the leader's facility i. The formulation follows.

$$\min_{x,s} \quad \sum_{j\in J}\sum_{i\in I} c_{ij}s_{ij} \tag{17}$$

$$\text{s.t.} \quad \sum_{i\in I} x_i = q \tag{18}$$

$$\sum_{i\in I} s_{ij} = 1, \quad \forall j \in J \tag{19}$$

$$s_{ij} \leq 1 - y_i + x_i, \quad \forall i \in I, \forall j \in J \tag{20}$$

$$\max_{y} \quad \sum_{j\in J}\sum_{i\in I} c_{ij}s_{ij} \tag{21}$$

$$\text{s.t.} \quad \sum_{i\in I} y_i = r \tag{22}$$

$$x_i, y_i, s_{ij} \in \{0,1\}, \quad \forall i \in I, \forall j \in J \tag{23}$$

The leader's objective function (17) aims at minimizing the system's cost after the fortifications and interdictions. Constraint (18) ensures that the leader fortifies exactly q facilities. Constraints (19) ensure that each customer uses exactly one facility. Constraints (20) ensure that a customer can not use an interdicted facility unless it is fortified. The follower's objective function (21) is the opposite of (17). Constraint (22) ensures that the follower interdicts exactly r facilities.

Similarly to the (r,p)-centroid problem, we obtain a min-max formulation for the RIMF by reformulating the problem in such a way that the variables controlled by the leader (follower) appear only in the leader's (follower's) constraints. The formulation uses five sets of variables. Variables x and y are defined in the above formulation, and we describe next variables t, u and v together with the new objective function. For each $i \in I$ and $j \in J$,

the binary variable u_{ij} is equal to 1 if and only if i is the cheapest fortified facility for customer j, the binary variable v_{ij} is equal to 1 if and only if i is the cheapest non-interdicted facility for customer j, and the binary variable t_{ij} is equal to 1 if and only if i is interdicted and any facility cheaper for the customer j than the facility i is also interdicted. Note that the cost of serving a given customer $j \in J$ is equal to the minimum between the serving cost to the cheapest fortified facility and the serving cost to the cheapest non-interdicted facility. This cost can be calculated by the following expression.

$$\sum_{i \in I} c_{ij} u_{ij} t_{ij} + \sum_{i \in I} c_{ij} v_{ij} \sum_{k \in I : c_{kj} \geq c_{ij}} u_{kj}, \tag{24}$$

Exchanging the order of the sums in the second term of (24), we obtain that the cost of serving customer $j \in J$ can be computed as

$$\sum_{i \in I} u_{ij} \left(c_{ij} t_{ij} + \sum_{k \in I : c_{kj} \leq c_{ij}} c_{kj} v_{kj} \right).$$

Summing over all customers, the objective function of our reformulation is

$$C(u, t, v) = \sum_{j \in J} \sum_{i \in I} u_{ij} \left(c_{ij} t_{ij} + \sum_{k \in I : c_{kj} \leq c_{ij}} c_{kj} v_{kj} \right). \tag{25}$$

For each customer $j \in J$ we define the constant $\varphi(k, j)$ denoting the kth cheapest facility for customer j. Ties are broken arbitrarily. The bilinear

min-max formulation for the RIMF follows.

$$\min_{x,u} \quad C(u,t,v) \tag{26}$$

$$\text{s.t.} \quad \sum_{i\in I} x_i = q \tag{27}$$

$$u_{ij} \le x_i, \quad \forall j \in J, \forall i \in I \tag{28}$$

$$\sum_{i\in I} u_{ij} = 1, \quad \forall j \in J \tag{29}$$

$$x_i, u_{ij} \in \{0,1\}, \quad \forall i \in I, \forall j \in J \tag{30}$$

$$\max_{y,t',t} \quad C(u,t,v) \tag{31}$$

$$\text{s.t.} \quad \sum_{i\in I} y_i = r \tag{32}$$

$$t_{ij} \le y_i, \quad \forall j \in J, \forall i \in I \tag{33}$$

$$t_{\varphi(1,j)j} = y_{\varphi(1,j)}, \quad \forall j \in J \tag{34}$$

$$t_{\varphi(i,j)j} \ge t_{\varphi(i+1,j)j}, \quad \forall j \in J, \forall i = 1, ..., |I| - 1 \tag{35}$$

$$v_{\varphi(1,j)j} = 1 - t_{\varphi(1,j)j}, \quad \forall j \in J \tag{36}$$

$$v_{\varphi(i,j)j} = t_{\varphi(i-1,j)j} - t_{\varphi(i,j)j}, \quad \forall j \in J, \forall i = 2, ..., |I| \tag{37}$$

$$y_i, t_{ij}, v_{ij} \in \{0,1\}, \quad \forall i \in I, \forall j \in J \tag{38}$$

Constraint (27) ensures that q facilities must be fortified. Constraints (28) ensure the consistency between the variables x and u. Constraints (29) ensure that for each customer j, there is exactly one cheapest fortified facility. The second level constraint (32) ensures that r facilities must be interdicted. Constraints (33) ensure the consistency between the variables t and y. Constraints (34) ensure that, if the customer j's cheapest facility is interdicted, then the corresponding t variable associated to this facility is equal to 1.

Constraints (35) ensure that, for each customer j and facility i, if $t_{ij} = 1$ then the t variables associated to customer j and any facility cheaper than i are also equal to 1. This ensures the consistency of the t variables with their definitions. Constraints (36) and (37) ensure that $v_{\varphi(i,j)j} = 1$ if and only if i is the cheapest facility for customer j such that $t_{ij} = 0$.

We see easily that problem (25) - (38) is a particular case of min-max bilinear problem (3). We solve the min-max bilinear problem by applying the framework described in Section 2. In order to speed up our method, the cuts are separated in the following way. We propose a greedy heuristic to efficiently find some violated cuts avoiding some IP optimizations. To describe this heuristic, we recall that given a fortification described by $\bar{x}$ and the corresponding $\bar{s}$, the separation problem looks for an interdiction described by y and the corresponding t and v that maximizes $C(u, t, v)$. The heuristic greedily constructs the strategy y by choosing r facilities. At each iteration, it chooses the facility that causes the maximum increase in $C(u, t, v)$. We always try to separate cuts first by the greedy heuristic avoiding some IP optimizations. We also define a threshold parameter ϵ to reduce the total number of separated cuts. While the gap is greater than ϵ, we separate cuts for any solution of the linear relaxation found during the branch-and-bound search. When the gap becomes smaller than or equal to ϵ, the cuts are separated only for integer solutions.

4. Computational experiments

In this section, we report computational experiments to evaluate the method proposed in Subsection 3.2. First, we compare the computational

performance of our method and the best previous exact one, proposed by [8]. We test our method on most instances used in [8] and present new results. In order to show the robustness of our method, we also test it on larger instances. The experiments include tests on the 150-node London benchmark data set [17]. That data set is composed of 150 nodes ($n = 150$) and is frequently used as a benchmark for the p-median problem. The other parameters of the problem vary as follows: $p \in \{25, 30, 40, 50, 60\}$, $q \in \{3, 4, 5, 6, 7, 8, 9, 10, 12\}$ and $r \in \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$. In all tests the p existing facilities are initially located at the optimal p-median sites of the data set. We tested several values for ϵ without a significant difference in the total computational time. Thus, we set $\epsilon = 0.03$. We use the MIP solver CPLEX 12.1 and all tests are carried out in a 2.31 GHz PC Pentium Intel Core 2 duo with 3 GB of RAM. Sections 4.1 and 4.2 show respectively the comparison between our method and the exact one proposed by [8], and statistics of our method for several instances.

4.1. Comparison between our method and the best exact one.

In this subsection we present a comparison of the computational performance of our method and the Implicit Enumeration (IE) proposed by [8]. For instances with $r \leq 4$ we separate the cuts by a pure enumerative method instead of executing the exact MIP separation given by (31) - (38). Tables 1 and 2 show the comparison for instances with $p \in \{40, 50, 60\}$ and small values of $r(r \leq 5)$ while Table 3 shows this comparison for instances with $p = 40$ and large values of $r(r > 5)$. The following headers are used for the columns: p, q and r indicate the instance characteristics, *Time(s)* indicates the computational time in seconds consumed by each method, and $\frac{IE}{This\ Paper}$

indicates the ratio between the computational time consumed by the IE and our method. For each instance, we marked in bold the ratio when our method was faster. The runs performed by [8] were carried out in an HP 2500 workstation, with an Intel(R) Xeon(R) CPU E5630 @ 2.53 GHz processor and 6GB RAM. That configuration is faster than the one used for our tests.

Tables 1, 2 and 3 show that our method can be slower than the IE for the smallest instances where both running times are very small. For the largest instances we observe the opposite: our method is usually faster than the IE. This can be explained by the fact that our method performs expensive MIP optimizations to separate cuts in order to obtain good lower bounds in the nodes that are close to the root. If, on one hand it reduces the asymptotic increase of running time as a function of the instance size, on the other hand it increases the absolute running time for small instances.

In Tables 1 and 2, the method of [8] was faster than ours in more than 78% of the instances. This can be explained by the fact that the values of r ranges from 2 to 5 for the instances of Tables 1and 2. On the other hand, the results presented in Table 3 for large instances ($r \geq 5$) show that our method was significantly faster for 79% of the instances.

In short, the comparison indicated that our method is more suitable for large instances.

4.2. Statistics of our method for several instances.

In this section we present statistics of our method for several instances. Tables 4, 5 and 6 show results for instances with $p = 40$, $p = 50$ and $p = 60$ respectively. The following headers are used for the columns: p, q, and r indicate the instance characteristics, $BestUB$ indicates the best upper bound

obtained for the problem, *Final gap(%)* indicates the gap between the best upper bound and the best lower bound, *Root gap(%)* indicates the gap between the root node relaxation lower bound and the value in the column *Best UB*, *#Nodes* indicates the total number of nodes created by the branch-and-cut tree, and *Total Time* indicates the total CPU time in seconds consumed by the complete branch-and-cut algorithm. For the instances that our method is not able to optimally solve, we report the results when the running time reached 36000 seconds (10 hours).

In Tables 4, 5 and 6 we note that the method optimally solved 50 out of the 60 instances tested in reasonable computational times where for 28 instances the total time consumed was smaller than 4000 seconds. Another interesting observation is that as the value of p, q and r increase, the instances also become more difficult. For a rough comparison against the method proposed in [8] on large instances, the authors state that their method requires about 6 hours (21600 seconds) to solve the instance with $p = 50$, $q = 8$ and $r = 7$, while our execution time is about one hour. Finally, a promising direction of improvement for our method is tightening the root node relaxation bounds. Observe that these gaps are smaller than 10% only for 9 out of 60 instances.

5. Conclusions

In this paper we presented a framework to model certain bilevel combinatorial optimization problems as bilinear min-max problems, and we derived a generic branch-and-cut algorithm that could be applied to any problem modeled in that way. Then, we showed that the best known algorithm pro-

posed to solve the discrete (r, p)-centroid problem is indeed a particular case of our framework. We further applied the framework to the RIMF and compare the computational results with the best previous exact approach for the problem. The results showed that our approach is more suitable for solving large instances.

Acknowledgements. MCR received support from CAPES. AAP received support from FAPERJ grant E-26/110.550/2010.

References

[1] Dempe S. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. Annals of the Association of American Geographers 2003;94(3):491–502.

[2] Colson B, Marcotte P, Savard G. Bilevel programming: A survey. 4OR: A Quarterly Journal of Operations Research 2005;3(2):87–107.

[3] Labbé M, Violin A. Bilevel programming and price setting problems. 4OR 2013;11(1):1–30.

[4] Moore J, Bard J. The mixed integer linear bilevel programming problem. Operations Research 1990;38:911–21.

[5] Roboredo MC, Pessoa AA. A branch-and-cut algorithm for the discrete (r—p)-centroid problem. European Journal of Operational Research 2013;224(1):101 –9. URL: http://www.sciencedirect.com/science/article/pii/S0377221712005991. doi:10.1016/j.ejor.2012.07.042.

[6] Hakimi S. On locating new facilities in a competitive environment. European Journal of Operational Research 1983;12(1):29 – 35. URL: http://www.sciencedirect.com/science/article/pii/0377221783901807.

[7] Church R, Scaparra M. Protecting critical assets: The r-interdiction median problem with fortification. Geographical Analysis 2007;39(2):129–46.

[8] Scaparra M, Church R. A bilevel mixed-integer program for critical infrastructure protection planning. Computers & Operations Research 2008;35(6):1905–23.

[9] Noltemeier H, Spoerhase J, Wirth H. Multiple voting location and single voting location on trees. European Journal of Operational Research 2007;181(2):654–67.

[10] Alekseeva E, Kochetov Y. Matheuristics and exact methods for the discrete (r— p)-centroid problem. In: Metaheuristics for Bi-level Optimization. Springer; 2013, p. 189–219.

[11] Davydov I, Kochetov Y, Carrizosa E. A local search heuristic for the (r—p)-centroid problem in the plane. Computers and Operations Research 2013;.

[12] Rodríguez C, Peñate D, Pérez J. An exact procedure and lp formulations for the leader–follower location problem. TOP: An Official Journal of the Spanish Society of Statistics and Operations Research 2010;18(1):97–121.

[13] Alekseeva E, Kochetova N, Kochetov Y, Plyasunov A. Heuristic and exact methods for the discrete (r —p)-centroid problem. In: Cowling P, Merz P, editors. Evolutionary Computation in Combinatorial Optimization; vol. 6022 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. ISBN 978-3-642-12138-8; 2010, p. 11–22.

[14] Scaparra M, Church R. An exact solution approach for the interdiction median problem with fortification. European Journal of Operational Research 2008;189(1):76–92.

[15] Liberatore F, Scaparra M, Daskin M. Analysis of facility protection strategies against an uncertain number of attacks: The stochastic r-interdiction median problem with fortification. Computers & Operations Research 2011;38(1):357–66.

[16] Aksen D, Piyade N, Aras N. The budget constrained r-interdiction median problem with capacity expansion. Central European Journal of Operations Research 2010;18(3):269–91.

[17] Goodchild MF, Noronha VT. Location-allocation for small computers. Department of Geography, University of Iowa; 1983.

Table 1: Comparing runtime between our method and [8] (IE) for instances for $p \in \{40, 50\}$ and small values of r.

Instance Characteristics			Optimal Value	Time		
p	q	r		This Paper	IE	$\frac{\text{IE}}{\text{This Paper}}$
40	4	2	75676.41	2.08	0.28	0.13
40	4	3	81766.35	6.55	2.31	0.35
40	4	4	88495.16	39.92	7.83	0.20
40	4	5	94687.71	76.44	21.31	0.28
40	6	2	75418.05	7.52	0.83	0.11
40	6	3	81424.85	26.13	8.80	0.34
40	6	4	87170.59	142.83	55.6	0.39
40	6	5	93286.72	267.44	206.3	0.77
40	8	2	74847.58	13.59	2.09	0.15
40	8	3	80370.63	84.47	46.35	0.55
40	8	4	86182.77	505.20	436.46	0.86
40	8	5	91664.38	1313.22	1683.56	**1.28**
50	5	2	60168.50	3.91	0.42	0.11
50	5	3	65160.41	30.39	4.65	0.15
50	5	4	69918.29	86.86	16.08	0.19
50	5	5	74694.85	205.31	58.13	0.28
50	8	2	59225.56	9.25	1.73	0.19
50	8	3	63552.59	71.50	28.88	0.40
50	8	4	68302.73	323.86	163.47	0.50
50	8	5	73055.22	645.91	967.65	**1.50**
50	10	2	58553.08	39.70	3.29	0.08
50	10	3	62261.17	47.34	93.57	**1.98**
50	10	4	67026.53	368.69	628.05	**1.70**
50	10	5	71140.40	986.72	5478.18	**5.55**

Table 2: Comparing runtime between our method and [8] (IE) for instances for $p = 60$ and small values of r.

Instance Characteristics			Optimal Value	Time(s)		
p	q	r		This Paper	IE	$\frac{\text{IE}}{\text{This Paper}}$
60	6	2	46563.64	18.33	1.15	0.06
60	6	3	50809.54	119.27	10.56	0.09
60	6	4	54621.16	349.42	33.12	0.09
60	6	5	58615.76	551.05	122.84	0.22
60	9	2	45889.14	13.31	2.36	0.18
60	9	3	49697.61	103.25	69.64	0.67
60	9	4	53509.22	689.90	333.38	0.48
60	9	5	56932.06	2516.75	1575.12	0.63
60	12	2	45310.07	36.95	3.93	0.11
60	12	3	48814.47	182.98	329.41	**1.80**
60	12	4	52011.79	1185.30	1730.80	**1.46**
60	12	5	55469.28	8664.08	11107.79	**1.28**

Table 3: Comparing runtime between our method and [8] (IE) for instances with $p = 40$ and large values of r.

Instance Characteristics			Optimal Value	Time(s)		
p	q	r		This Paper	IE	$\frac{\text{IE}}{\text{This Paper}}$
40	4	6	101598.44	87.73	59.03	0.67
40	4	7	108225.05	189.85	109.12	0.57
40	4	8	115080.07	139.98	211.26	**1.51**
40	4	9	122170.27	126.25	386.55	**3.06**
40	4	10	130408.32	253.36	631.91	**2.49**
40	6	6	100078.89	747.14	690.93	0.92
40	6	7	106352.95	1646.36	1496.06	0.91
40	6	8	113960.94	2311.90	3531.39	**1.53**
40	6	9	120606.98	2414.58	7101.38	**2.94**
40	6	10	126403.82	2175.92	12041.69	**5.53**
40	8	6	97508.17	3011.03	6589.98	**2.19**
40	8	7	102380.26	4023.22	16845.28	**4.19**
40	8	8	108230.84	5750.22	44151.18	**7.68**
40	8	9	113464.87	6435.05	100684.59	**15.65**
40	8	10	118595.86	11269.89	146913.77	**13.04**
40	10	6	94526.17	8714.67	47658.43	**5.47**
40	10	7	99124.14	11912.66	139906.16	**11.74**
40	10	8	103738.66	17488.11	370635.16	**21.19**
40	10	9	108677.52	28438.00	836554.54	**29.42**

Table 4: Statistics of our method for instances with $p = 40$, $q \in \{4, 6, 8, 10\}$ and $r \in \{6, 7, 8, 9, 10\}$.

p	q	r	Best UB	Final gap(%)	Root LB	Root gap(%)	#B&B Nodes	#SEP greedy	#SEP IP	#Cuts greedy	#Cuts IP	#Cuts Total	Greedy Time	IP Time	Total Time
40	4	6	101598.45	0.00	91917.49	9.53	1197	29	9	20	6	26	0.13	14.48	87.73
40	4	7	108225.05	0.00	97489.74	9.92	2363	31	14	17	9	26	0.23	23.78	189.85
40	4	8	115080.07	0.00	104361.53	9.31	2341	32	11	21	7	28	0.25	18.08	139.98
40	4	9	122170.27	0.00	110602.86	9.47	1624	29	11	18	7	25	0.38	17.36	126.25
40	4	10	130408.32	0.00	117395.55	9.98	1967	54	27	27	18	45	0.75	45.86	253.36
40	6	6	100078.89	0.00	89008.28	11.06	22276	52	17	35	13	48	0.17	28.5	747.14
40	6	7	106352.95	0.00	93428.04	12.15	55034	49	19	30	13	43	0.39	32.39	1646.36
40	6	8	113960.94	0.00	99336.79	12.83	82948	91	25	66	20	86	0.75	42.28	2311.9
40	6	9	120606.98	0.00	103694.00	14.02	63914	134	32	102	28	130	1.5	54.47	2414.58
40	6	10	126403.82	0.00	109806.78	13.13	48559	159	60	99	54	153	2.02	104.56	2175.92
40	8	6	97508.17	0.00	85376.13	12.44	116371	104	30	74	25	99	0.47	51.89	3011.03
40	8	7	102380.26	0.00	89826.16	12.26	119457	147	49	98	42	140	1.06	85.84	4023.22
40	8	8	108230.84	0.00	94659.18	12.54	163238	224	73	151	64	215	1.94	126.47	5750.22
40	8	9	113464.87	0.00	98281.47	13.38	157258	314	95	219	85	304	3.08	162.53	6435.05
40	8	10	118595.86	0.00	102493.07	13.58	217352	407	95	312	88	400	5.05	167.06	11269.89
40	10	6	94526.17	0.00	83766.48	11.38	328384	229	44	185	37	222	1.11	76.45	8714.67
40	10	7	99124.14	0.00	87200.23	12.03	400639	409	34	375	26	401	2.7	59.61	11912.66
40	10	8	103738.66	0.00	90633.08	12.63	433422	624	73	551	62	613	5.03	127.81	17488.11
40	10	9	108677.52	0.00	93424.26	14.04	631386	946	83	863	73	936	9.67	140.34	28438.00
40	10	10	113149.70	1.66	97330.25	13.98	$\geq$ 529543	$\geq$ 1313	$\geq$ 87	$\geq$ 1226	$\geq$ 78	$\geq$ 1304	$\geq$ 16.27	$\geq$ 149.41	$\geq$ 36000

Table 5: Statistics of our method for instances with $p = 50$, $q \in \{4, 6, 8, 10\}$ and $r \in \{6, 7, 8, 9, 10\}$.

p	q	r	Best UB	Final gap(%)	Root LB	Root gap(%)	#B&B Nodes	#SEP greedy	#SEP IP	#Cuts greedy	#Cuts IP	#Cuts Total	Greedy Time	IP Time	Total Time
50	4	6	80341.38	0.00	72982.20	9.16	728	51	10	41	8	49	0.31	19.36	98.64
50	4	7	85269.27	0.00	77504.42	9.11	768	69	18	51	14	65	0.61	38.70	196.27
50	4	8	90305.63	0.00	81676.66	9.56	1242	78	30	48	23	71	0.86	65.27	286.99
50	4	9	96300.69	0.00	86160.86	10.53	1086	80	26	54	20	74	1.19	56.08	282.78
50	4	10	105375.7	0.00	90546.05	14.07	3631	172	70	102	66	168	2.69	141.23	829.31
50	6	6	78583.71	0.00	69117.29	12.05	4517	259	24	235	19	254	1.59	50.64	418.45
50	6	7	83443.82	0.00	73054.91	12.45	6441	320	47	273	38	311	2.80	104.11	869.98
50	6	8	89318.99	0.00	77509.56	13.22	13672	618	98	520	91	611	6.77	218.52	1732.29
50	6	9	95684.25	0.00	81789.85	14.52	34030	1057	311	746	272	1018	13.48	684.53	6931.51
50	6	10	100246.16	0.00	85872.92	14.34	24432	911	242	669	230	899	13.95	510.64	4809.85
50	8	6	77546.05	0.00	66888.93	13.74	21316	802	82	720	75	795	5.14	169.55	1539.48
50	8	7	82394.56	0.00	70940.23	13.90	40159	1270	144	1126	136	1262	10.50	323.59	**3975.19**
50	8	8	87482.62	0.00	74908.79	14.37	72388	2429	375	2054	361	2415	25.39	832.95	12521.50
50	8	9	91875.63	0.00	78461.14	14.60	82105	2942	699	2243	684	2927	37.91	1511.41	20866.10
50	8	10	96269.41	0.00	81213.18	15.64	106383	3505	546	2959	532	3491	53.66	1188.28	29831.70
50	10	6	75733.38	0.00	65793.89	13.12	49228	2271	152	2119	135	2254	13.64	319.17	4418.08
50	10	7	80231.98	0.00	68946.77	14.07	117522	3206	107	3099	100	3199	26.78	236.33	14773.90
50	10	8	84901.59	1.56	72380.57	14.75	≥143577	≥6357	≥465	≥5892	≥440	≥6332	≥65.28	≥1036.84	≥36000
50	10	9	88692.85	2.85	75011.42	15.43	≥109012	≥7365	≥435	≥6930	≥426	≥7356	≥92.58	≥925.45	≥36000
50	10	10	93603.84	4.55	77953.41	16.72	≥92995	≥8588	≥506	≥8082	≥493	≥8575	≥130.01	≥1088.56	≥36000

Table 6: Statistics of our method for instances with $p = 60$. $q \in \{4, 6, 8, 10\}$ and $r \in \{6, 7, 8, 9, 10\}$.

p	q	r	Best UB	Final gap(%)	Root LB	Root gap(%)	#B&B Nodes	#SEP greedy	#SEP IP	#Cuts greedy	#Cuts IP	#Cuts Total	Greedy Time	IP Time	Total Time
60	4	6	63761.97	0.00	57517.63	9.79	1034	80	21	59	17	76	0.55	47.95	331.83
60	4	7	69139.79	0.00	61444.24	11.13	2180.00	132	47	85	45	130	1.45	107.81	782.08
60	4	8	74730.36	0.00	63946.34	14.43	2943	204	106	98	93	191	2.64	242.05	1369.18
60	4	9	79351.34	0.00	68942.91	13.12	4946	233	163	70	129	199	3.75	375.94	2652.29
60	4	10	84507.98	0.00	72244.29	14.51	6370	312	242	70	167	237	6.20	567.25	4250.89
60	6	6	62915.66	0.00	54845.76	12.83	6440.00	393	115	278	111	389	3.02	258.16	1526.54
60	6	7	66681.65	0.00	57842.45	13.26	9375	619	122	497	117	614	6.53	287.31	2637.86
60	6	8	70687.26	0.00	61140.79	13.51	12994	621	149	472	143	615	8.34	351.98	3794.05
60	6	9	74504.22	0.00	64747.67	13.10	13684	591	108	483	100	583	9.48	251.63	2700.96
60	6	10	79387.74	0.00	67882.79	14.49	24530	838	419	419	406	825	15.83	981.14	9668.91
60	8	6	61129.04	0.00	53072.93	13.18	24720.00	1094	115	979	107	1086	8.50	260.59	4062.66
60	8	7	64769.70	0.00	56497.20	12.77	43907	1510	153	1357	148	1505	15.58	362.80	8566.16
60	8	8	69195.09	0.00	59097.64	14.59	71217	2432	532	1900	514	2414	31.44	1262.58	17285.30
60	8	9	73254.84	1.66	62542.77	14.62	≥86906	≥3945	≥1350	≥2595	≥1269	≥3864	62.58	≥3144.92	≥36000
60	8	10	77280.88	2.12	65455.16	15.30	≥93746	≥4166	≥1788	≥2378	≥1672	≥4050	≥78.71	≥3898.56	≥36000
60	10	6	60201.28	0.00	51909.63	13.77	103455	2660	342	2318	308	2626	21.11	815.38	16371.60
60	10	7	64273.21	2.33	55099.62	14.27	≥121532	≥6777	≥1095	≥5682	≥959	≥6641	≥69.57	≥2614.67	≥36000
60	10	8	67807.04	4.65	56676.40	16.42	≥98423	≥8284	≥1100	≥7184	≥1063	≥8247	≥107.76	≥2568.33	≥36000
60	10	9	71453.90	5.85	60188.96	15.77	≥97554	≥7995	≥1037	≥6958	≥1025	≥7983	≥126.59	≥2397.22	≥36000
60	10	10	75006.42	6.95	62659.46	16.46	≥87895	≥7337	≥1113	≥6224	≥1105	≥7329	≥139.34	≥2521.84	≥36000