SOLVING BILEVEL COMBINATORIAL OPTIMIZATION AS BILINEAR MIN-MAX OPTIMIZATION VIA A BRANCH-AND-CUT ALGORITHM

Artur Alves Pessoa
Universidade Federal Fluminense
artur@producao.uff.br

Michael Poss
University of Technology of Compiègne
michael.poss@hds.utc.fr

Marcos Costa Roboredo
Universidade Federal Fluminense
mcr.marcos@yahoo.com.br

Luiz Aizemberg
Universidade Federal Fluminense
luizaizemberg@gmail.com

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 customer 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.

KEY WORDS. Integer programming, bi-level programming, Min-max problems, r-Interdiction median problem with fortification.

Main area: Optimization

1 Introduction

Bilevel programming can model optimization problems where two noncooperative decison makers (competitors) choose their decisions (strategies) in a sequential way. Each competitor aims at optimizing his 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 his 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. Some good references on the topic are Dempe (2003), Colson *et al.* (2005) and Moore and Bard (1990), among others.

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 Roboredo and Pessoa (2013) and Roboredo and Pessoa (2012) for the (r, p)-centroid problem (Hakimi, 1983) and a competitive location problem called Budget Constrained Centroid Problem proposed by Plastria (2001), respectively. While Roboredo and Pessoa (2012, 2013) could optimally solve many open instances for the first time, the general framework behind the algorithms was not fully understood in Roboredo and Pessoa (2012, 2013). Second, we apply the framework to the r-Interdiction Median Problem with Fortification (Church and Scaparra, 2007) and our results show that our algorithm is more efficient than the best from the literature (Scaparra and Church, 2008a) 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 researches.

2 The Framework

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

$$\min_{x \in X} \max_{y \in Y} y^T C x + Dx + Ey \qquad (1)$$

We solve min-max problem (1) by replacing the inner maximization with a (finite) set of linear inequalities:

$$\min_{x \in X} z \tag{2}$$

$$s.t.\ z \geq y^T Cx + Dx + Ey,\ \forall y \in Y \tag{3}$$

Because $Y \subset \{0,1\}^m$, the number of constraints (3) 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 $\overline{x}$, the separation problem associated to (3) can be cast as:

$$\max_{y \in Y} y^T C\overline{x} + D\overline{x} + Ey \tag{4}$$

Our exact algorithm is built on the top of the branch-and-cut algorithm using the model given by (2)-(3). Constraints (3) are added through a cut callback by applying the exact model (4). Heuristic can also be used for the separation problem in order to efficiently find some violated cuts and thus avoiding solving exactly problem (4).

3 Applications

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. The (r,p)-centroid problem was proposed by Hakimi (1983) and has been widely researched in literature, see Roboredo and Pessoa (2013) and the references therein.

Now we describe a natural bilevel formulation for the (r,p)-centroid problem. This formulation uses four sets of variables x, y, s and t. 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. Then, for each $i \in I$ and $j \in J$, binary variable s_{ij} is equal to 1 if and only if customer j is served by the leader's facility i. Similarly, for each $i \in I$ and $j \in J$, binary variable t_{ij} is equal to 1 if and only if customer j is served by the follower's facility i. The formulation follows.

$$\max_{x,s} \quad \sum_{j \in J} \sum_{i \in I} w_j s_{ij} \tag{5}$$

$$\text{s.t.} \quad \sum_{i \in I} x_i = p \tag{6}$$

$$\sum_{i \in I} s_{ij} \leq 1, \quad \forall j \in J \tag{7}$$

$$s_{ij} \leq x_i, \quad \forall i \in I, \forall j \in J \tag{8}$$

$$s_{ij} \leq 1 - \sum_{k \in I | d_{kj} < d_{ij}} t_{kj}, \quad \forall i \in I, \forall j \in J \tag{9}$$

$$\max_{y,t} \quad \sum_{j \in J} \sum_{i \in I} w_j t_{ij} \tag{10}$$

$$\text{s.t.} \quad \sum_{i \in I} y_i = r \tag{11}$$

$$\sum_{i \in I} t_{ij} \leq 1, \quad \forall j \in J \tag{12}$$

$$t_{ij} \leq y_i, \quad \forall i \in I, \forall j \in J \tag{13}$$

$$t_{ij} \leq 1 - \sum_{k \in I | d_{kj} \leq d_{ij}} s_{kj}, \quad \forall i \in I, \forall j \in J \tag{14}$$

$$x_i, y_i, s_{ij} \in \{0, 1\}, \quad \forall i \in I, \forall j \in J \tag{15}$$

The leader's objective function (5) aims at maximizing the total demand served by the leader. Constraint (6) ensures that the leader places exactly p facilities. Constraints (7) ensure that each customer is served by at most one leader's facility. Constraints (8) ensure the consistency between the variables x and s. Constraints (9) ensure that the leader can only use the facility i to serve the customer j if there is no facility placed by the follower closer to j than i. The follower's objective function (10) aims at maximizing the total demand served by the follower. Constraints (11) ensures that the follower places exactly r facilities. Constraints (12) ensure that each customer is served by at most one follower's facility. Constraints (13) ensure the consistency between the variables y and t. Finally, constraints (14) ensure that the follower can not use the facility i to serve the customer j if there is a facility placed by the leader at least as close to j as i.

Roboredo and Pessoa (2013) 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 Roboredo and Pessoa (2013). To obtain this formulation we first follow the steps of Roboredo and Pessoa (2013), and observe that the competition is a zero-sum game. As a result, the leader's objective function (5) is equivalent to minimize the total demand served by the follower. Next, we reformulate the problem in such a way that the variables controlled by the leader do not appear in the follower's constraints and vice-versa. For that, we use the set of variables x, y, s and t. Variables x and y are as above. For each $i \in I$ and $j \in J$, the binary variable s_{ij} is equal to 1 if and only if i is the facility placed by the leader closest to the customer j. For each $i \in I$ and $j \in J$, the binary variable t_{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.

$$\min_{x,s} \sum_{j\in J}\sum_{k\in I}\left(w_j \sum_{i\in I|d_{ij}>d_{kj}} s_{ij}\right) t_{kj} \tag{16}$$

$$\text{s.t.} \quad \sum_{i\in I} x_i = p \tag{17}$$

$$s_{ij} \le x_i, \quad \forall j\in J, \forall i \in I \tag{18}$$

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

$$x_i, s_{ij} \in \{0,1\}, \quad \forall i\in I, \forall j \in J \tag{20}$$

$$\max_{y,t} \sum_{j\in J}\sum_{k\in I}\left(w_j \sum_{i\in I|d_{ij}>d_{kj}} s_{ij}\right) t_{kj} \tag{21}$$

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

$$t_{ij} \le y_i, \quad \forall j\in J, \forall i \in I \tag{23}$$

$$\sum_{i\in I} t_{ij} = 1, \quad \forall j \in J \tag{24}$$

$$y_i, t_{ij} \in \{0,1\}, \quad \forall i\in I, \forall j \in J \tag{25}$$

The first level objective function (16) aims at minimizing the total demand served by the follower. Constraint (17) ensures that p facilities must be placed by the leader. Constraints (18) ensure the consistency between the variables x and s. Constraints (19) ensure that for each customer j, there is exactly one facility closest to j placed by the leader. The second level constraint (22) ensures that r facilities must be placed by the follower. Constraints (23) ensure the consistency between the variables t and y. Constraints (24) ensure that for each customer j, there is exactly one facility closest to j placed by the follower.

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 is to choose the group of q facilities to fortify and the second one is to choose the group of r facilities to interdict.

The researches on RIMF are recent. The RIMF was proposed by Church and Scaparra (2007), 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 (2008b) proposed an approach where the size of the model is significantly reduced. The main weakness of that approach is to require 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 $r = 7$. Scaparra and Church (2008a) 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 $r = 5$.

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.

$$\min_{x,s} \quad \sum_{j\in J}\sum_{i\in I} c_{ij}s_{ij} \tag{26}$$

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

$$\sum_{i\in I} s_{ij} = 1, \quad \forall j \in J \tag{28}$$

$$s_{ij} \leq 1 - y_i + x_i, \quad \forall i \in I, \forall j \in J \tag{29}$$

$$\max_{y} \quad \sum_{j\in J}\sum_{i\in I} c_{ij}s_{ij} \tag{30}$$

$$\text{s.t.} \quad \sum_{i\in I} y_i = r \tag{31}$$

$$x_i, y_i, s_{ij} \in \{0,1\}, \quad \forall i \in I, \forall j \in J \tag{32}$$

The leader's objective function (26) aims at minimizing the system's cost after the fortifications and interdictions. Constraint (27) ensures that the leader fortifies exactly q facilities. Constraints (28) ensure that each customer uses exactly one facility. Constraints (29) ensure that a customer can not use an interdicted facility unless it is fortified. The follower's objective function (30) is the opposite of (26). Constraint (31) 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 s', t and t' together with the new objective function. For each $i \in I$ and $j \in J$, the binary variable s'_{ij} is equal to 1 if and only if i is the cheapest fortified facility for customer j, the binary variable t'_{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.

$$\sum_{i\in I} c_{ij}s'_{ij}t_{ij} + \sum_{i\in I} c_{ij}t'_{ij} \sum_{k\in I: c_{kj}\geq c_{ij}} s'_{kj}, \tag{33}$$

Exchanging the order of the sums in the second term of (33), we obtain that the cost of serving customer $j \in J$ can be computed as $\sum_{i\in I} s'_{ij}\left(c_{ij}t_{ij} + \sum_{k\in I: c_{kj}\leq c_{ij}} c_{kj}t'_{kj}\right)$. Summing over all customers, the objective function of our reformulation is

$$C(s', t, t') = \sum_{j\in J}\sum_{i\in I} s'_{ij}\left(c_{ij}t_{ij} + \sum_{k\in I: c_{kj}\leq c_{ij}} c_{kj}t'_{kj}\right). \tag{34}$$

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.

$$\min_{x,s'} \quad C(s',t,t') \tag{35}$$

$$\text{s.t.} \quad \sum_{i \in I} x_i = q \tag{36}$$

$$s'_{ij} \le x_i, \quad \forall j \in J, \forall i \in I \tag{37}$$

$$\sum_{i \in I} s'_{ij} = 1, \quad \forall j \in J \tag{38}$$

$$x_i, s'_{ij} \in \{0,1\}, \quad \forall i \in I, \forall j \in J \tag{39}$$

$$\max_{y,t',t} \quad C(s',t,t') \tag{40}$$

$$\text{s.t.} \quad \sum_{i \in I} y_i = r \tag{41}$$

$$t_{ij} \le y_i, \quad \forall j \in J, \forall i \in I \tag{42}$$

$$t_{\varphi(1,j)j} = y_{\varphi(1,j)}, \quad \forall j \in J \tag{43}$$

$$t_{\varphi(i,j)j} \ge t_{\varphi(i+1,j)j}, \quad \forall j \in J, \forall i = 1,...,|I|-1 \tag{44}$$

$$t'_{\varphi(1,j)j} = 1 - t_{\varphi(1,j)j}, \quad \forall j \in J \tag{45}$$

$$t'_{\varphi(i,j)j} = t_{\varphi(i-1,j)j} - t_{\varphi(i,j)j}, \quad \forall j \in J, \forall i = 2,...,|I| \tag{46}$$

$$y_i, t_{ij}, t'_{ij} \in \{0,1\}, \quad \forall i \in I, \forall j \in J \tag{47}$$

Constraint (36) ensures that q facilities must be fortified. Constraints (37) ensure the consistency between the variables x and s'. Constraints (38) ensure that for each customer j, there is exactly one cheapest fortified facility. The second level constraint (41) ensures that r facilities must be interdicted. Constraints (42) ensure the consistency between the variables t and y. Constraints (43) 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 (44) 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 (45) and (46) ensure that $t'_{\varphi(i,j)j} = 1$ if and only if i is the cheapest facility for customer j such that $t_{ij} = 0$.

We solve the min-max bilinear problem (34) - (47) 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 t' that maximizes $C(s',t,t')$. The heuristic greedily constructs the strategy y by choosing r facilities. At each iteration, it chooses the facility that causes the maximum increase in $C(s',t,t')$. 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 present report computational experiments made to test the method proposed in Subsection 3.2. First we compare the computational performance of our method and the best previous exact one, proposed by Scaparra and Church (2008a). We test our method on all

Table 1: Comparing runtime between our method and Scaparra and Church (2008a) (IE) for instances with $p = 25$ and $p = 30$

Instance Characteristics			Optimal Value	Time(s)	
p	q	r		This Paper	IE
25	3	4	153638.54	6.72	**1.86**
25	3	5	164458.35	25.42	**4.99**
25	3	6	174942.60	19.91	**9.91**
25	3	7	188282.97	**25.48**	26.89
25	3	8	205611.14	69.97	**41.53**
25	5	4	143058.40	8.92	**7.13**
25	5	5	151559.19	23.92	**22.61**
25	5	6	162485.24	**31.28**	55.71
25	5	7	171987.27	**68.97**	170.76
25	5	8	181881.35	**82.28**	246.03
25	7	4	137307.81	**17.61**	19.11
25	7	5	147589.13	**60.78**	80.89
25	7	6	156685.61	**105.25**	209.96
25	7	7	164595.84	**176.45**	616.69
25	7	8	172623.60	**265.75**	980.67
30	3	4	121378.81	6.08	**2.69**
30	3	5	132032.99	13.10	**9.36**
30	3	6	140618.50	**22.47**	24.34
30	3	7	152969.85	49.92	**43.97**
30	3	8	164159.70	**57.82**	72.31
30	5	4	118060.47	**13.44**	16.31
30	5	5	128667.35	75.54	**70.70**
30	5	6	137061.54	**114.55**	195.29
30	5	7	146299.89	**133.36**	436.55
30	5	8	155709.62	**239.34**	693.39
30	7	4	114789.52	**34.28**	61.00
30	7	5	121953.59	**106.62**	357.04
30	7	6	130678.87	**222.33**	994.23
30	7	7	136730.79	**273.20**	2832.11
30	7	8	144073.46	**346.03**	4255.58

instances tested in that paper. 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 (Goodchild and Noronha, 1983). 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 use the CPLEX 12.1 and all tests are carried out in a 2.13 GHz PC Pentium Intel Core 2 duo with 2 GB of RAM. Section 4.1 and 4.2 show respectively the comparison between our method and the exact one proposed by Scaparra and Church (2008a), and the computational results for larger 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 Scaparra and Church (2008a). For instances with $r \leq 4$ we separate the cuts by a pure enumerative method instead of executing the exact MIP separation given by (40) - (47). For all the instances in this subsection we use $\epsilon = 0.03$ except for instances with $p = 40$, where we use $\epsilon = 0.05$. Table 1 shows the comparison for instances with $p \in \{25, 30\}$, while Table 2 shows the comparison for instances with $p \in \{40, 50, 60\}$.

Table 2: Comparing runtime between our method and Scaparra and Church (2008a) (IE) for instances with $p = 40$, $p = 50$ and $p = 60$

Instance Characteristics			Optimal Value	Time(s)	
p	q	r		This Paper	IE
40	4	2	75676.41	2.08	**0.20**
40	4	3	81766.35	6.55	**2.67**
40	4	4	88495.16	39.92	**15.30**
40	4	5	94687.71	76.44	**58.71**
40	6	2	75418.05	7.52	**0.52**
40	6	3	81424.85	26.13	**11.10**
40	6	4	87170.59	142.83	**110.76**
40	6	5	93286.72	**267.44**	476.28
40	8	2	74847.58	13.59	**1.30**
40	8	3	80370.63	84.47	**53.23**
40	8	4	86182.77	**505.20**	797.86
40	8	5	91664.38	**1313.22**	3467.25
50	5	2	60168.50	3.91	**0.25**
50	5	3	65160.41	30.39	**4.42**
50	5	4	69918.29	86.86	**22.86**
50	5	5	74694.85	205.31	**117.28**
50	8	2	59225.56	9.25	**1.14**
50	8	3	63552.59	71.50	**26.51**
50	8	4	68302.73	323.86	**197.98**
50	8	5	73055.22	**645.91**	1665.66
50	10	2	58553.08	39.70	**2.11**
50	10	3	62261.17	**47.34**	77.23
50	10	4	67026.53	**368.69**	664.77
50	10	5	71140.40	**986.72**	7441.07
60	6	2	46563.64	18.33	**0.70**
60	6	3	50809.54	119.27	**9.53**
60	6	4	54621.16	349.42	**45.11**
60	6	5	58615.76	551.05	**204.12**
60	9	2	45889.14	13.31	**1.44**
60	9	3	49697.61	103.25	**65.67**
60	9	4	53509.22	689.90	**351.38**
60	9	5	56932.06	2516.75	**2229.49**
60	12	2	45310.07	36.95	**2.55**
60	12	3	48814.47	**182.99**	254.38
60	12	4	52011.79	**1185.30**	1568.08
60	12	5	55469.28	**8664.08**	13088.10

The following headers are used for the columns: p, q and r indicate the instance characteristics, *Time(s)* indicates the computational time in seconds. For each instance, we mark in bold the smallest required time. The runs performed by Scaparra and Church (2008a) were carried out in a Pentium 4, 2.8 Ghz processor and 1GB of RAM.

Table 1 shows that our method can be several times slower than the IE for the smallest instances where both running times are very small. On the largest instances we observe the opposite: our method is several times faster. This can be explained by the fact that our method performs expensive MIP optimizations to separate cuts in order to obtain a 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. Our method was faster in 14 out of 24 instances.

The results shown in Table 2 have a similar pattern except that the method of Scaparra and Church (2008a) 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 4 to 8 for the instances of Table 1 and only from 2 to 5 for the instances of Table 2. We believe that our method would be several times faster for instances with r greater than 5.

4.2 Computational results for large instances.

In this section we present statistics of our method for large instances ($p \geq 40$ and $r \geq 6$). For all the instances in this subsection we use $\epsilon = 0.03$ except for instances with $p = 40$, where we use $\epsilon = 0.1$. Table 3 shows the results for $p \in \{40, 50, 60\}$. 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 Table 3, 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 Scaparra and Church (2008a) 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 proposed 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 to large instances.

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

References

Church, R. and Scaparra, M. (2007), Protecting critical assets: The r-interdiction median problem with fortification. *Geographical Analysis*, v. 39, n. 2, p. 129–146.

Colson, B., Marcotte, P. and Savard, G. (2005), Bilevel programming: A survey. *4OR: A Quarterly Journal of Operations Research*, v. 3, n. 2, p. 87–107.

Dempe, S. (2003), Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. *Annals of the Association of American Geographers*, v. 94, n. 3, p. 491–502.

Goodchild, M. F. and Noronha, V. T. *Location-allocation for small computers*. Department of Geography, University of Iowa, 1983.

Hakimi, S. (1983), On locating new facilities in a competitive environment. *European Journal of Operational Research*, v. 12, n. 1, p. 29 – 35.

Moore, J. and Bard, J. (1990), The mixed integer linear bilevel programming problem. *Operations Research*, v. 38, p. 911–921.

Plastria, F. (2001), Static competitive facility location: an overview of optimisation approaches. *European Journal of Operational Research*, v. 129, n. 3, p. 461–470.

Roboredo, M. C. and Pessoa, A. A. A branch-and-cut algorithm for a budget constrained centroid problem. *Anais do XLIV Simposio Brasileiro de Pesquisa Operacional (SBPO)*. Sobrapo. In Portuguese, 2012.

Roboredo, M. C. and Pessoa, A. A. (2013), A branch-and-cut algorithm for the discrete (r—p)-centroid problem. *European Journal of Operational Research*, v. 224, n. 1, p. 101 – 109.

Scaparra, M. and Church, R. (2008a), A bilevel mixed-integer program for critical infrastructure protection planning. *Computers & Operations Research*, v. 35, n. 6, p. 1905–1923.

Scaparra, M. and Church, R. (2008b), An exact solution approach for the interdiction median problem with fortification. *European Journal of Operational Research*, v. 189, n. 1, p. 76–92.

Table 3: Statistics of our method for instances with $p \in \{40, 50, 60\}$. $q \in \{4, 6, 8, 10\}$ and $r \in \{6, 7, 8, 9, 10\}$.

p	q	r	Best UB	Final gap(%)	Root gap(%)	#B&B Nodes	Total Time
40	4	6	101598.45	0.00	9.53	1197	87.73
40	4	7	108225.05	0.00	9.92	2363	189.85
40	4	8	115080.07	0.00	9.31	2341	139.98
40	4	9	122170.27	0.00	9.47	1624	126.25
40	4	10	130408.32	0.00	9.98	1967	253.36
40	6	6	100078.89	0.00	11.06	22276	747.14
40	6	7	106352.95	0.00	12.15	55034	1646.36
40	6	8	113960.94	0.00	12.83	82948	2311.9
40	6	9	120606.98	0.00	14.02	63914	2414.58
40	6	10	126403.82	0.00	13.13	48559	2175.92
40	8	6	97508.17	0.00	12.44	116371	3011.03
40	8	7	102380.26	0.00	12.26	119457	4023.22
40	8	8	108230.84	0.00	12.54	163238	5750.22
40	8	9	113464.87	0.00	13.38	157258	6435.05
40	8	10	118595.86	0.00	13.58	217352	11269.89
40	10	6	94526.17	0.00	11.38	328384	8714.67
40	10	7	99124.14	0.00	12.03	400639	11912.66
40	10	8	103738.66	0.00	12.63	433422	17488.11
40	10	9	108677.52	0.00	14.04	631386	28438.00
40	10	10	113149.70	1.66	13.98	$\geq$ 529543	$\geq$ 36000
50	4	6	80341.38	0.00	9.16	728	98.64
50	4	7	85269.27	0.00	9.11	768	196.27
50	4	8	90305.63	0.00	9.56	1242	286.99
50	4	9	96300.69	0.00	10.53	1086	282.78
50	4	10	105375.7	0.00	14.07	3631	829.31
50	6	6	78583.71	0.00	12.05	4517	418.45
50	6	7	83443.82	0.00	12.45	6441	869.98
50	6	8	89318.99	0.00	13.22	13672	1732.29
50	6	9	95684.25	0.00	14.52	34030	6931.51
50	6	10	100246.16	0.00	14.34	24432	4809.85
50	8	6	77546.05	0.00	13.74	21316	1539.48
50	8	7	82394.56	0.00	13.90	40159	3975.19
50	8	8	87482.62	0.00	14.37	72388	12521.50
50	8	9	91875.63	0.00	14.60	82105	20866.10
50	8	10	96269.41	0.00	15.64	106383	29831.70
50	10	6	75733.38	0.00	13.12	49228	4418.08
50	10	7	80231.98	0.00	14.07	117522	14773.90
50	10	8	84901.59	1.56	14.75	$\geq$143577	$\geq$36000
50	10	9	88692.85	2.85	15.43	$\geq$109012	$\geq$36000
50	10	10	93603.84	4.55	16.72	$\geq$92995	$\geq$36000
60	4	6	63761.97	0.00	9.79	1034	331.83
60	4	7	69139.79	0.00	11.13	2180	782.08
60	4	8	74730.36	0.00	14.43	2943	1369.18
60	4	9	79351.34	0.00	13.12	4946	2652.29
60	4	10	84507.98	0.00	14.51	6370	4250.89
60	6	6	62915.66	0.00	12.83	6440	1526.54
60	6	7	66681.65	0.00	13.26	9375	2637.86
60	6	8	70687.26	0.00	13.51	12994	3794.05
60	6	9	74504.22	0.00	13.10	13684	2700.96
60	6	10	79387.74	0.00	14.49	24530	9668.91
60	8	6	61129.04	0.00	13.18	24720	4062.66
60	8	7	64769.70	0.00	12.77	43907	8566.16
60	8	8	69195.09	0.00	14.59	71217	17285.30
60	8	9	73254.84	1.66	14.62	$\geq$86906	$\geq$36000
60	8	10	77280.88	2.12	15.30	$\geq$93746	$\geq$36000
60	10	6	60201.28	0.00	13.77	103455	16371.60
60	10	7	64273.21	2.33	14.27	$\geq$121532	$\geq$36000
60	10	8	67807.04	4.65	16.42	$\geq$98423	$\geq$36000
60	10	9	71453.90	5.85	15.77	$\geq$97554	$\geq$36000
60	10	10	75006.42	6.95	16.46	$\geq$87895	$\geq$36000