RAIRO-Oper. Res. 53 (2019) 505–516
https://doi.org/10.1051/ro/2017060

RAIRO Operations Research
www.rairo-ro.org

AN EXACT APPROACH FOR THE R-INTERDICTION MEDIAN PROBLEM WITH FORTIFICATION

Marcos Costa Roboredo[1,*], Artur Alves Pessoa[1] and Luiz Aizemberg[2]

Abstract. We study the r-interdiction median problem with fortification (RIMF), which considers demand points that are served by facilities. If a facility is interdicted, it can not serve any demand point, increasing the total system cost. To avoid an interdiction, a facility can be fortified. The problem consists of fortifying facilities knowing that some facilities will be interdicted. We propose a branch-and-cut algorithm for the RIMF and several experiments attest that our method outperforms the best exact algorithm found.

Mathematics Subject Classification. 90C10, 90C11, 90C27, 90C47, 90C57, 90B06, 90B80.

Received June 16, 2016. Accepted July 23, 2017.

1. Introduction

Critical infrastructures are systems composed of facilities, networks and assets whose a minor destruction or disruption of some component causes a huge negative impact on the whole efficiency. In the United States, for example, the department of homeland security considers the following sixteen critical infrastructure sectors: Chemical, Commercial Facilities, Communications, Critical Manufacturing, Dams, Defense Industry, Emergency Services, Energy, Financial Services, Food and Agriculture, Government Facilities, Nuclear Reactors, Materials, Waste, Transportation Systems, Water and Wastewater Systems [8].

The History has shown that critical infrastructures are subject to natural disasters, failures or intentional attacks. These events are called interdictions in the literature. In general, the system is represented by a network where the infrastructures and the customers are located on the nodes or on the links. The customers are served by the infrastructures according to some rule, generating a serving cost of the system. In this context, the interdictor problem consists of maximizing the increase on this cost identifying a subset of infrastructures to be interdicted. The literature about interdiction problems is vast. [7] proposed integer linear programming (ILP) formulations for the node interdiction problem on median and covering networks. [11] dealt with the interdiction problem on hub-and-spoke networks. Recently, [23] reviewed several interdiction models on supply chain systems. In response to the interdictions, several models have been proposed in the literature aiming to mitigate the damage caused by those events. [6] proposed an ILP formulation for the protection of median

Keywords. Bilevel Programming, Branch-and-cut, The r-interdiction median problem with fortification.

[1] Production Engineering Department, Federal Fluminense University, R. Passo da Pátria, 156, 24210-240, São Domingos, Niterói, RJ, Brazil.

[2] BNDES, Brazilian Development Bank, Avenida República do Chile, 100, 20031-917, Rio de Janeiro, RJ, Brazil
*Corresponding author: mcr.marcos23@gmail.com

Article published by EDP Sciences

networks. To protect power grids, [25] proposed a decomposition approach. [20] proposed an algorithm to identify the best way to protect a rail intermodal terminal network. [16] proposed a model to defend organizations from cyber attacks. For a more complete overview of the literature on definitions, models and algorithms to protection and interdiction problems, we refer to [2, 5].

We focus in this paper on the r-interdiction median problem with fortification (RIMF) which was proposed by [6]. In the RIMF model, we are given n demand points and p facilities where each demand point is served by the closest facility, generating a serving cost proportional to the distance between them. The total serving cost of the system is the sum of the serving cost of each demand point. If a facility is interdicted then the demand points served by this facility skip to the closest facility not interdicted, increasing the total serving cost. A way to mitigate this damage is fortifying the facilities. If a facility is fortified then it can not be interdicted. 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 increase the cost of the damaged system as much as possible.

The literature on RIMF is rather recent. The RIMF was proposed by [6], 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 $r = 4$[3]. [21] proposed a reformulation for the RIMF and new techniques that significantly reduce the resulting model size. The main weakness of proposed reformulation 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 $p = 30$, $q = 7$ and $r = 7$. In [22], 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. [12] considered uncertainty on the number of facilities to be interdicted. [1] studied 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 demand point is interdicted based on the assumption that, in this case, it is necessary to expand the capacity of another facility. In the RIMF variant studied by [13], a facility can be interdicted just partially and the impact of an interdiction can propagate across the network. In the variant proposed by [14], the role of facility recovery time on system performance and the possibility of multiple disruptions over time is taken into account.

The RIMF can be seen as a bilevel integer problem (BIP) 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. In the literature, there are a few papers about generic exact methods for BIP. Branch-and-bound algorithms for linear BIPs were proposed by [4, 15], but it is able to solve only instances with up to 10 general integer and 35 binary variables for the first level problem. Some techniques are based on specific characteristics of the problem, such as a decomposition method based on super valid inequalities [10, 17] and cutting plane algorithms [24]. Recently, [19] derived a MILP formulation for a competitive location problem named discrete $(r|p)$-centroid. That formulation allowed the use of commercial integer programming solvers which opened the room for the resolution of several open instances.

The contribution of this paper is to propose an exact approach for the RIMF. First, we present a bilevel formulation for the problem that can be translated into an ILP with a polynomial number of variables and an exponential number of constraints. To solve the formulation, the constraints are generated on demand in a branch-and-cut algorithm. Our approach was compared with the best exact one proposed by [22], being faster mainly for median and large values of p, q, and r. Besides, optimal solutions for several open instances are reported in this paper.

The remainder of this paper is divided as follows. Section 2 presents formally the problem and a bilevel formulation for it. Section 3 presents the proposed bilevel formulation. Section 4 derives an ILP formulation for the RIMF and shows how to solve this formulation *via* branch-and-cut algorithm. Section 5 presents our computational experiments. Finally, Section 6 summarizes our conclusions.

[3]It is usual to measure the complexity of instances in the RIMF literature by the values of p, q, and r, and not by n as usual.

2. The problem

Consider n demand points and p facilities where the demand w_j of each demand point j ($j = 1, \ldots, n$) is completely served by the closest facility. The distance between a facility i ($i = 1, \ldots, p$) and a demand point j is denoted by d_{ij}, yielding a serving cost of $c_{ij} = w_j d_{ij}$. The total serving cost of the system is given by the sum of the serving costs of all demand points. If a facility is interdicted, then the demand points served by this facility skip to the non-interdicted facility that causes the smallest increase in the serving cost of the system. A way to avoid part of this cost increase is to fortify a set of facilities. Fortified facilities cannot be interdicted. The problem consists of choosing a group of q facilities to fortify knowing that r facilities will be interdicted. When choosing which facilities to fortify, one aims to minimize the final serving cost of the system, assuming that such a cost is maximized when the interdicted facilities are chosen.

The formulation proposed by [22] to solve this bilevel optimization problem is present in the following. Let $V_{ij} = \{k \in I | c_{kj} > c_{ij}\}$ be the set of all facilities that are farther from the customer j than facility i. Define also the following decision variables:

$$x_i = \begin{cases} 1 & \text{if the facility } i \text{ is fortified} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ij} = \begin{cases} 1 & \text{if the facility } i \text{ is the cheapest not interdicted facility to the demand point } j \\ 0 & \text{otherwise} \end{cases}$$

$$s_i = \begin{cases} 1 & \text{if the facility } i \text{ is interdicted} \\ 0 & \text{otherwise} \end{cases}$$

The formulation follows.

$$\min_{x} \sum_{i \in I} \sum_{j \in J} c_{ij} z_{ij} \tag{2.1}$$

$$\text{s.t.} \sum_{i \in I} x_i = q \tag{2.2}$$

$$x_i \in \{0, 1\}, \quad \forall i \in I \tag{2.3}$$

$$\max_{s,z} \sum_{i \in I} \sum_{j \in J} c_{ij} z_{ij} \tag{2.4}$$

$$\text{s.t.} \sum_{i \in I} z_{ij} = 1, \quad \forall j \in J \tag{2.5}$$

$$\sum_{i \in I} s_i = r \tag{2.6}$$

$$\sum_{h \in V_{ij}} z_{hj} \leq s_i, \quad \forall j \in J, \forall i \in I \tag{2.7}$$

$$s_i + x_i \leq 1, \quad \forall i \in I \tag{2.8}$$

$$s_i \in \{0, 1\}, \quad \forall i \in I \tag{2.9}$$

$$z_{ij} \in \{0, 1\}, \quad \forall j \in J, \forall i \in I \tag{2.10}$$

The first level objective function (2.1) shows that the system planner's goal is to minimize the total serving cost after the fortifications and interdictions. The constraint (2.2) indicates that q facilities are fortified. The second level objective function (2.4) shows that the interdictor aims to maximize the total serving cost. The set

of constraints (2.5) ensures that just one facility serves a customer j. The constraint (2.6) shows that r facilities are interdicted. The constraints (2.7) ensure that each customer j must be served by the cheapest non-interdicted facility. Finally, the constraints (2.8) indicate that a facility i cannot be fortified and interdicted at the same time.

For fixed x satisfying (2.2) and (2.3), let $H(x)$ be the optimum value for (2.4) - (2.10). In this case, the previous formulation can be rewritten as $\min_x H(x)$ subject to (2.2) and (2.3), as presented in [22].

3. Proposed bilevel formulation

In this section, we present a new bilevel formulation for the problem. The main advantage of our formulation is to model the interaction between the fortification and interdiction strategies only through the objective functions. This means that no constraint in one of the levels uses a decision variable of the other level. Moreover, the objective functions (which are represented by the same expression with opposite signs) are bilinear, *i.e.* it becomes linear if we fix all variables of either the system planner or the interdictor. As a result, we reduced the problem to a min-max bilinear optimization problem. This structure will permit to derive an equivalent formulation with a unique level, a polynomial number of variables and an exponential number of constraints.

The proposed bilevel formulation for the RIMF uses four sets of binary variables: x, y, y', s and t. The variables x and s receive the same definition that in the formulation (2.1)–(2.10). The other variables receive the following definition:

$$y'_j = \begin{cases} 1 & \text{if none of the } r \text{ closest facilities to the demand point } j \text{ are fortified} \\ 0 & \text{otherwise} \end{cases}$$

$$s_i = \begin{cases} 1 & \text{if the facility } i \text{ is interdicted} \\ 0 & \text{otherwise} \end{cases}$$

$$t_{ij} = \begin{cases} 1 & \text{if the facility } i \text{ is interdicted and any closer facility to the demand point } j \text{ is also} \\ & \text{interdicted} \\ 0 & \text{otherwise} \end{cases}$$

We only create the variable y_{ij} when the facility i is one of the r closest facilities with respect to the demand point j. It is not necessary to create all other y variables because, since only r facilities are interdicted, fortifying a facility farther from the demand point j than its rth closest facility does not change the cost of serving j. To represent these cases (where the fortified facilities cannot change the cost of serving j), we use the binary variables y'_j. Similarly, we also only create the variable t_{ij} when the facility i is one of the r closest facilities with respect to the demand point j. It is easy to see that the remaining variables would be fixed to zero by definition.

Now, let $\varphi(i, j)$ indicate the ith closest facility to the demand point j. The formulation follows.

$$\min_{x,y,y'} \quad C(y, y', t) = \sum_{j\in J} c_{\varphi(1,j),j} + \sum_{j\in J}\sum_{i=1}^{r} t_{\varphi(i,j)j} \left(y'_j + \sum_{k=i+1}^{r} y_{\varphi(k,j)j} \right) \left(c_{\varphi(i+1,j),j} - c_{\varphi(i,j)j} \right) \tag{3.1}$$

$$\text{s.t.} \quad \sum_{i\in I} x_i = q \tag{3.2}$$

$$y_{\varphi(i,j),j} \le x_{\varphi(i,j)}, \quad \forall j \in J, \forall i = 1, 2, \ldots, r \tag{3.3}$$

$$y'_j + \sum_{i=1}^{r} y_{\varphi(i,j),j} = 1, \quad \forall j \in J \tag{3.4}$$

$$x_i \in \{0,1\}, \quad \forall i \in I \tag{3.5}$$

$$y'_j, y_{\varphi(i,j),j} \in \{0,1\}, \quad \forall j \in J, \forall i = 1,2,\ldots,r \tag{3.6}$$

$$\max_{s,t} \quad C(y,y',t) \tag{3.7}$$

$$\text{s.t.} \quad \sum_{i\in I} s_i = r \tag{3.8}$$

$$t_{\varphi(i,j),j} \le s_{\varphi(i,j)}, \quad \forall j \in J, \forall i = 2,3,\ldots,r \tag{3.9}$$

$$t_{\varphi(1,j)j} = s_{\varphi(1,j)}, \quad \forall j \in J \tag{3.10}$$

$$t_{\varphi(i,j)j} \ge t_{\varphi(i+1,j)j}, \quad \forall j \in J, \forall i = 1,2,\ldots,r-1 \tag{3.11}$$

$$s_i \in \{0,1\}, \quad \forall i \in I \tag{3.12}$$

$$t_{\varphi(i,j),j} \in \{0,1\}, \quad \forall j \in J, \forall i = 1,2,\ldots,r \tag{3.13}$$

The first level objective function (3.1) indicates the total serving cost of the system after the fortifications and the interdictions. The first part of this function is a constant, representing the total cost if each demand point is served by the closest facility, while the second part indicates the cost increase when each closest facility to each demand point is interdicted. Constraint (3.2) ensures that q facilities are fortified. Constraints (3.3) ensure that $y_{\varphi(i,j),j} \forall i = 1,\ldots,r$ can only be 1 if i is fortified. Constraints (3.4) ensure that exactly one facility can be considered as the closest fortified for each given demand point j. The second level objective function (3.7) indicates that the interdictor aims to maximize the total serving cost. Constraint (3.8) ensures that r facilities are interdicted. Constraints (3.9) and (3.10) ensure that $t_{i,j}$ can only be 1 if i is interdicted. Constraints (3.11) ensure that $t_{i,j}$ can only be 1 if all closer facilities of j are also interdicted. Note that the model does not avoid fortification and interdiction at the same time. In this case, any customer can be covered by this facility normally.

4. The branch-and-cut algorithm

In this section, we present the newly proposed algorithm, referred to as B&C-RIMF. As previosly mentioned, the formulation (3.1)−(3.13) has the following three characteristics: the first level variables x, y and y' do not appear in the second level constraints, the second level variables s and t do not appear in the first level constraints, and the objective functions are opposite and bilinear. As a min-max bilinear formulation with linear constraints, it leads to the following ILP formulation. Let F be the set of values that can be assigned to the variable vectors s and t satisfying the constraints (3.8)−(3.13). Note that each element of F represents a feasible interdiction strategy. We replace the second level problem (3.7)−(3.13) by a set of linear inequalities over an auxiliary variable z that represents the objective function. The proposed reformulation follows.

$$\min_{x,y,y',z} \quad z \tag{4.1}$$

$$\text{s.t.} \quad (3.2)-(3.6) \tag{4.2}$$

$$z \ge C(y,y',t), \quad \forall (s,t) \in F \tag{4.3}$$

Consider a relaxed model (referred to as the master problem) that contains all constraints (3.2)−(3.4), and a initially empty subset of the constraints (4.3). The missing constraints (4.3) may be inserted on demand by

a cut separation routine. Given a feasible solution $(\bar{x}, \bar{y}, \bar{y'}, \bar{z}) \in \{0,1\}^{p+(r+1)n} \times \mathbb{R}$ to the master problem, a violated constraint (4.3) can be found by solving the ILP (3.7)−(3.13), where $\bar{y}$, and $\bar{y'}$ are treated as constants, taking advantage of the fact that the objective function is bilinear. This problem is referred to as the separation subproblem. Let s^* and t^* be the optimal values of the variable vectors s and t, respectively, for the separation subproblem solved in a given iteration. if the constraint $z \geq C(y, y', t^*)$ is violated in the master problem, then it should be inserted on it. Otherwise, the solution represented by $(\bar{x}, \bar{y}, \bar{y'}, \bar{z})$ satisfies all constraints (4.3).

However, it may be inefficient to wait for the full resolution of the master problem to add new constraints to it. The proposed algorithm overcomes this limitation by allowing to solve the separation subproblems associated to intermediary solutions found through the execution of the ILP solver. As a result, the B&C-RIMF is a branch-and-cut method built on the top of the algorithm implemented by any general purpose ILP solver. In this case, given a feasible solution to a linear relaxation $(\bar{x}, \bar{y}, \bar{y'}, \bar{z}) \in [0,1]^{p+(r+1)n} \times \mathbb{R}$ of the master problem, a violated constraint to be inserted on it can be found in the same way as before. Note that the fact that $\bar{x}$, $\bar{y}$, and $\bar{y'}$ are not necessarily integer does not impact the separation algorithm. The constraints (4.3) are added in a cut callback provided by the solver, which is responsible to manage all the branch-and-bound tree.

The separation subproblem can be executed for both integer and fractional relaxed solutions. Separating cuts for fractional relaxed solutions can strengthen the bound, reducing the number of nodes of the branch-and-bound tree. On the other hand, it increases the number of executions of the separation subproblems. The following cut separation strategy is then used to speed up the method. In each node of the tree, we consider the current optimality gap, which is provided by the solver. If this gap is greater than a given parameter ϵ, we solve the separation subproblem associated to (4.3) regardless of whether $\bar{x}$, $\bar{y}$, and $\bar{y'}$ are integer or not. Otherwise, if this gap is smaller than or equal to ϵ, the separation is only solved if all those variables have integer values. Note that skipping the separation subproblem when $\bar{x}$, $\bar{y}$, and $\bar{y'}$ are all integer is not possible since it may let the ILP solver accept infeasible solutions as valid. We set the value of ϵ according to the instance size.

Algorithm 1 describes the steps of our method, where ϵ is a given parameter. The algorithm starts in the line 10 where the ILP (4.1)−(4.2) is built. To solve the problem (line 11), we use a commercial ILP solver where we set the procedure SEPARATE as a cut callback. The solver calls this procedure at each node of the classical branch-and-bound algorithm or whenever it finds a new integer solution. First the procedure SEPARATE, tries to find a violated cut (4.3) by solving the ILP (3.7)−(3.13) fixing the current solution. To avoid some ILP optimizations, we do not solve (3.7)−(3.13) when the current solution is fractional and the relative difference between the current cost $\bar{z}$ and the best feasible cost does not exceed ϵ.

Algorithm 1. B&C-RIMF$(I, J, c, q, r, \epsilon)$

```
1: procedure SEPARATE(x̄, ȳ, ȳ', z̄, ε)
2:     if (current gap > ε) or ((x̄, ȳ, ȳ') are integer) then
3:         Solve the ILP (3.7) - (3.13) for fixed y = ȳ and y' = ȳ'.
4:         Let (s*, t*) be the obtained optimal solution.
5:         if C(ȳ, ȳ', t*) > z̄ then
6:             add z ≥ C(y, y', t*) to the ILP as a cut.
7:         end if
8:     end if
9: end procedure

10: Build the Master ILP (4.1)−(4.2).
11: Call the ILP solver with the Procedure SEPARATE as a cut callback.
```

5. Computational experiments

In this section, we report computational experiments with the instances used by [22]. 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 [9] and 316-node Alberta [3] benchmark data sets. The London and Alberta data sets are composed of respectively 150 nodes ($n = 150$) and 316 nodes ($n = 316$). To generate the instances, we varies the parameters 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 facilities are initially located at the optimal p-median sites of the data set. We set $\epsilon = 0.05$ because this value gives better results for the set of instances tested. We used a PC with Intel Core i7-4790 3.60GHz CPU and 3 GB of memory running the Windows 8 operating system, CPLEX 12.5.1 and the C++ language. A single thread was used for the tests. Sections 5.1 and 5.2 show respectively the comparison between the proposed method B&C-RIMF and the Implicit Enumeration proposed by [22], and statistics of our method for several instances.

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

In this subsection, we present a comparison of the computational performance of B&C-RIMF against the Implicit Enumeration (IE) proposed by [22] for London instances with $p \geq 40$. The IE results for Alberta instances with $p \geq 40$ were not available. Table 1 shows the comparison for instances with $p \in \{40, 50, 60\}$ and small values of $r (r \leq 5)$, while Table 2 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 identify the instance, *Time(s)* indicates the computational time in seconds consumed by each method, and $\frac{IE}{B\&C\text{-}RIMF}$ indicates the ratio between the computational time consumed by the IE and our method. For each instance, we marked in bold the smallest computational time. The results for the IE were obtained directly from the authors of [22]. These results were carried out in a HP 2500 workstation, with an Intel(R) Xeon(R) CPU E5630 @ 2.53 GHz processor and 6GB RAM using CPLEX 12.5. To overcome the difference between the machines specifications, we took a Passmark scores for the two processors at the website of the PassMark Software [18] and calculated a ratio between them. The Passmark scores obtained by our processor and the one used by [22] was respectively 9997 and 5166, indicating that our processor is about 1.935 times faster. Therefore we divided each time obtained by [22] by that previous ratio.

To complement the comparative results presented in Tables 1 and 2, we present the Table 3 with the average ratios for each pair of values for p and r for instances with $p \in \{40, 50, 60\}$ and small values of r.

Table 3 shows that the average ratio increases as the value of r also increases. Besides, Tables 1 and 2 show the B&C-RIMF is slower than the IE for almost instances with $r \leq 3$. On the other hand, for instances $r \geq 4$, B&C-RIMF is faster for the most instances. More specifically, our method was faster for all the instances with $r \geq 6$. This can be explained by the fact that our method performs expensive ILP optimizations to separate cuts in order to obtain good lower bounds in the nodes that are close to the root. On the one hand, it reduces the asymptotic increase of running time as a function of the value of r, on the other hand it increases the absolute running time for small instances. As a result, on instances with small values of r, where the number of enumerated solutions is not so big, the reduction on the number of alternatives tested does not pay the increase in the time required to process each alternative. The opposite is observed for large values of r. In short, the comparison indicated that our method is clearly more suitable for instances with $r \geq 4$.

5.2. Statistics of our method for several instances.

In this section we present statistics of B&C-RIMF for several instances. The Table 4 shows the optimal cost and the total computational time in seconds spent by our method for Alberta instances with $p = 40$ or $p = 60$ and $r > 5$. In this Table, we can note that the hardest instance spent just 51.14 seconds.

The London instances are significantly harder. So we present more detailed results of B&C-RIMF for them. Tables 5 and 6 these results for instances with $r > 5$ and $p = 40$ and $p = 60$ respectively. It is valid to emphasize that the optimal solutions for those instances are being presented by the first time in this paper. The following new headers are used for the columns: *Root time* shows the total CPU time in seconds spent at the root node. *Root gap(%)* indicates the gap between the root node relaxation lower bound and the value in the column *Optimal value*, *#Nodes* indicates the total number of nodes created by the branch-and-cut tree,

TABLE 1. Comparing runtime between our method and [22] (IE) for London instances with $p \in \{40, 50, 60\}$ and small values of r.

Instance			Optimal	Time		
p	q	r	Value	This Paper	IE	$\frac{\text{IE}}{B\&C-RIMF}$
40	4	2	75676.41	0.89	**0.14**	0.16
40	6	2	75418.05	2.27	**0.43**	0.19
40	8	2	74847.58	4.09	**1.08**	0.26
50	5	2	60168.5	1.81	**0.22**	0.12
50	8	2	59225.56	6.20	**0.89**	0.14
50	10	2	58553.08	7.94	**1.70**	0.21
60	6	2	46563.64	3.47	**0.59**	0.17
60	9	2	45889.14	9.98	**1.22**	0.12
60	12	2	45310.07	12.16	**2.03**	0.17
40	4	3	81766.35	3.25	**1.19**	0.37
40	6	3	81424.85	11.23	**4.55**	0.40
40	8	3	80370.63	31.13	**23.95**	0.77
50	5	3	65160.41	6.69	**2.40**	0.36
50	8	3	63552.59	21.02	**14.93**	0.71
50	10	3	62261.17	**29.77**	48.36	1.62
60	6	3	50809.54	14.13	**5.46**	0.39
60	9	3	49697.61	36.34	**35.99**	0.99
60	12	3	48814.47	**64.84**	170.24	2.63
40	4	4	88495.16	5.84	**4.05**	0.69
40	6	4	87178.67	**17.19**	28.73	1.67
40	8	4	86182.77	**111.25**	225.56	2.03
50	5	4	69918.29	15.44	**8.31**	0.54
50	8	4	68302.73	**54.16**	84.48	1.56
50	10	4	67026.53	**109.05**	324.57	2.98
60	6	4	54621.16	26.91	**17.12**	0.64
60	9	4	53509.22	**106.92**	172.29	1.61
60	12	4	52011.79	**202.81**	894.47	4.41
40	4	5	94687.71	17.27	**11.01**	0.64
40	6	5	93286.72	**58.61**	106.61	1.82
40	8	5	91664.38	**204.16**	870.06	4.26
50	5	5	74694.85	30.81	**30.04**	0.98
50	8	5	73055.22	152.61	500.08	3.28
50	10	5	71140.4	**308.39**	2831.10	9.18
60	6	5	58615.76	**61.05**	63.48	1.04
60	9	5	56932.06	**259.27**	814.02	3.14
60	12	5	55469.28	**768.69**	5740.46	7.47

#Sep and *#Cuts* indicate respectively the total number of separation problems solved and the total number of cuts generated during the algorithm. *Sep Time(s)* and *Total time(s)* indicate the total CPU time in seconds consumed by respectively the ILP separations and the complete branch-and-cut algorithm.

In Tables 5 and 6 we can note that the B&C-RIMF is able to solve large instances in reasonable computational time. Just for three instances, it was necessary to spend more than one hour to find the optimal solution. Another interesting observation is that the total time spent with the separation problems is not so high even for large instances.

TABLE 2. Comparing runtime between our method and [22] (IE) for London instances with $p = 40$ and large values of r.

Instance			Optimal	Time(s)		
p	q	r	Value	This Paper	IE	$\frac{\text{IE}}{B\&C-RIMF}$
40	4	6	101 598.45	**18.13**	30.51	1.68
40	4	7	108 225.05	**31.98**	56.39	1.76
40	4	8	115 080.07	**26.91**	109.18	4.06
40	4	9	122 170.27	**96.14**	199.77	2.08
40	4	10	130 408.32	**80.32**	326.57	4.07
40	6	6	100 078.89	**124.73**	357.07	2.86
40	6	7	106 352.95	**180.59**	773.16	4.28
40	6	8	113 960.94	**337.52**	1825.01	5.41
40	6	9	120 606.98	**528.11**	3669.96	6.95
40	6	10	126 403.82	**454.11**	6223.10	13.70
40	8	6	97 508.17	**529.31**	3405.67	6.43
40	8	7	102 380.26	**410.33**	8705.57	21.22
40	8	8	108 230.84	**727.22**	22 817.15	31.38
40	8	9	113 464.87	**832.94**	52 033.38	62.47
40	8	10	118 595.86	**1131.55**	75 924.43	67.10
40	10	6	94 526.17	**564.86**	24 629.68	43.60
40	10	7	99 124.14	**854.09**	72 302.93	84.65
40	10	8	103 738.66	**1422.69**	191 542.72	134.63
40	10	9	108 677.52	**1897.73**	432 327.93	227.81

TABLE 3. Average ratios for London instances with $p \in \{40, 50, 60\}$ and small values of r .

(p,r)	2	3	4	5
40	0.21	0.51	1.46	2.24
50	0.16	0.90	1.69	4.48
60	0.15	1.33	2.22	3.88

The main weakness observed in the algorithm are the root gaps. Consequently, the number of branch-and-bound tree nodes necessary to guarantee the optimal solution is also high. For the largest two instances, for example, this number exceeded one million nodes.

6. CONCLUSIONS

In this paper we proposed an exact approach for the RIMF named B&C-RIMF. The main advantage of that algorithm is the use of an ILP formulation with a polynomial number of variables and exponential number of constraints, which allows the embedding into efficient commercial ILP solvers. The method proposed is compared with the best exact one found in the literature, presenting good comparative results, mainly for the large instances.

Although the results are good, some improvements can be made in future researches, like the use of strengthened valid inequalities, which may reduce the root gaps and consequently the total execution time.

TABLE 4. Statistics of our method for Alberta instances with $p = 40$ or $p = 60$ and $r > 5$.

Instance		Alberta			
		$p = 40$		$p = 60$	
q	r	Optimal Value	time total(s)	Optimal Value	time total(s)
4	7	39 621 861	7.44	25 459 477	4.95
4	8	44 466 470	12.09	28 815 123	11.45
4	9	46 681 810	13.27	33 497 305	15.83
4	10	49 028 457	11.81	35 935 342	18.08
6	7	28 282 324	9.67	16 910 414	6.28
6	8	30 456 785	13.59	18 309 306	6.98
6	9	32 255 407	17.42	19 841 887	10.01
6	10	34 429 868	14.17	22 139 538	20.95
8	7	23 120 651	7.88	14 847 753	15.09
8	8	24 872 232	12.92	16 281 149	23.67
8	9	26 578 611	12.84	17 551 630	27.63
8	10	27 965 427	15.14	18 751 848	32.95
10	7	21 050 525	18.28	14 041 243	44.66
10	8	22 258 377	28.14	14 808 043	37.92
10	9	23 449 581	30.52	15 641 030	51.14
10	10	24 618 777	38.12	16 466 064	48.52

TABLE 5. Statistics of our method for London instances with $p = 40$ and $r > 5$.

Instance			Optimal	Root		Branch and bound tree				
p	q	r	value	time(s)	gap(%)	#Nodes	#Sep	#Cuts	Sep time(s)	Total time(s)
40	4	6	101 598.45	5.48	8.42	1762	17	15	8.69	18.13
40	4	7	108 225.05	14.58	8.71	1519	23	20	22.72	31.98
40	4	8	115 080.07	9.5	8.60	1318	21	18	17.33	26.91
40	4	9	122 170.27	18.38	8.08	1627	57	52	77.22	96.14
40	4	10	130 408.32	13.44	8.72	1159	57	50	60.94	80.30
40	6	6	100 078.89	8.84	10.31	26 706	82	78	30.52	124.73
40	6	7	106 352.95	14.73	11.37	20 670	108	103	90.73	180.59
40	6	8	113 960.94	10.78	11.99	30 628	256	250	174.98	337.52
40	6	9	120 606.98	17.03	13.29	41 550	366	345	267.91	528.11
40	6	10	126 403.82	16.38	13.43	32 718	314	308	222.20	454.11
40	8	6	97 508.17	7.22	11.74	113 316	311	305	106.56	529.31
40	8	7	102 380.26	8.16	12.68	53 051	290	284	153.64	410.30
40	8	8	108 230.84	12.42	12.56	106 010	412	405	181.39	727.22
40	8	9	113 464.87	20.27	13.03	83 658	586	578	273.30	832.94
40	8	10	118 595.86	9.48	14.26	84 590	812	806	363.09	1131.50
40	10	6	94 526.17	8.42	12.74	109 972	402	392	98.09	564.80
40	10	7	99 124.14	21.59	11.61	100 325	811	800	246.95	854.09
40	10	8	103 738.66	6.95	13.58	199 585	864	856	209.08	1422.69
40	10	9	108 677.52	6.63	14.71	267 033	900	890	223.88	1897.73
40	10	10	113 149.70	9.38	14.88	213 973	1452	1438	363.55	2482.63

TABLE 6. Statistics of our method for London instances with $p = 60$ and $r > 5$.

Instance			Optimal	Root		Branch and bound tree				
p	q	r	value	time(s)	gap(%)	#Nodes	#Sep	#Cuts	Time sep(s)	Time total(s)
60	4	6	63 761.97	10.38	8.18	1937	31	29	15.09	28.77
60	4	7	69 139.79	13.36	10.52	4441	44	41	40.44	63.39
60	4	8	74 730.36	10.09	12.48	3969	92	87	80.08	116.39
60	4	9	79 351.34	19.13	11.63	8093	83	78	129.08	174.72
60	4	10	84 507.98	18.88	12.05	6990	100	91	142.63	192.98
60	6	6	62915.66	10.19	11.44	7940	148	143	60.63	117.09
60	6	7	66681.65	16.91	11.97	10 551	138	131	112.00	181.48
60	6	8	70 687.26	12.38	14.29	23 342	181	175	153.41	285.25
60	6	9	74 504.22	24.92	10.68	27 052	142	139	178.66	315.14
60	6	10	79 387.74	17.66	11.85	42 603	192	188	233.11	453.06
60	8	6	61 129.04	14.09	11.82	36 721	264	258	105.09	282.30
60	8	7	64 769.70	19.28	11.84	82781	274	268	169.94	534.81
60	8	8	69 195.09	16.77	12.64	164 055	564	559	376.61	1181.86
60	8	9	73 254.84	15.89	14.80	248 572	826	821	806.45	2158.44
60	8	10	77 280.88	11.33	16.11	293 539	1085	1073	1026.42	2916.61
60	10	6	60 201.28	10.21	15.25	262 097	795	786	337.28	1469.36
60	10	7	64 273.21	17.87	13.99	651 907	1134	1130	536.98	3413.61
60	10	8	67 807.04	10.69	15.50	810 884	1792	1786	1024.16	5719.19
60	10	9	71 453.90	24.28	15.25	1 155 956	2133	2123	1299.33	9137.80
60	10	10	75 006.42	13.50	16.31	1 275 396	3022	3017	2020.66	14204.44

REFERENCES

[1] D. Aksen, N. Piyade and N. Aras, The budget constrained r-interdiction median problem with capacity expansion. *Central Europ. J. Oper. Res.* **18** (2010) 269–291.

[2] D.L. Alderson, G.G. Brown and W.M. Carlyle, Assessing and improving operational resilience of critical infrastructures and other systems. INFORMS Tutorials in Operations Research. Bridging Data and Decisions (2014) 180–215.

[3] O. Alp, E. Erkut and Z. Drezner, An efficient genetic algorithm for the p-median problem. *Ann. Oper. Res.* **122** (2003) 21–42.

[4] J.F. Bard and J.T. Moore, An algorithm for the discrete bilevel programming problem. *Naval Research Logistics* **39** (1992) 419–435.

[5] G. Brown, M. Carlyle, J. Salmerón and K. Wood, Defending critical infrastructure. *Interfaces* **36** (2006) 530–544.

[6] R.L. Church and M.P. Scaparra, Protecting critical assets: The r-interdiction median problem with fortification. *Geographical Anal.* **39** (2007) 129–146.

[7] R.L. Church, M.P. Scaparra and R.S. Middleton, Identifying critical infrastructure: the median and covering facility interdiction problems. *Ann. Association Amer. Geographers* **94** (2004) 491–502.

[8] Department of Homeland Security of the United States. Available at: https://www.dhs.gov/critical-infrastructure-sectors (2019).

[9] M.F. Goodchild and V.T. Noronha, Location-allocation for small computers, Department of Geography, University of Iowa (1983).

[10] E. Israeli and R.K. Wood, Shortest-path network interdiction. *Networks* **40** (2002) 97–111.

[11] T.L. Lei, Identifying critical facilities in hub-and-spoke networks: A hub interdiction median problem. *Geographical Anal.* **45** (2013) 105–122.

[12] F. Liberatore, M.P. Scaparra and M.S. Daskin, Analysis of facility protection strategies against an uncertain number of attacks: the stochastic r-interdiction median problem with fortification. *Comput. Oper. Res.* **38** (2011) 357–366.

[13] F. Liberatore, M.P. Scaparra and M.S. Daskin, Hedging against disruptions with ripple effects in location analysis. *Omega* **40** (2012) 21–30.

[14] C. Losada, M.P. Scaparra and J.R. O'Hanley, Optimizing system resilience: a facility protection model with recovery time. *Europ. J. Oper. Res.* **217** (2012) 519–530.

[15] J.T. Moore and J.F. Bard, The mixed integer linear bilevel programming problem. *Oper. Res.* **38** (1990) 911–921.

[16] A.K. Nandi, H.R. Medal and S. Vadlamani, Interdicting attack graphs to protect organizations from cyber attacks: A bi-level defender–attacker model. *Comput. Oper. Res.* **75** (2016) 118–131.

[17] J.R. O'Hanley and R.L. Church, Designing robust coverage networks to hedge against worst-case facility losses. *Europ. J. Oper. Res.* **209** (2011) 23–36.

[18] Passmark software. Available at: https://www.cpubenchmark.net (2019).
[19] M.C. Roboredo and A.A. Pessoa, A branch-and-cut algorithm for the discrete (r|p)-centroid problem. *Europ. J. Operat. Res.* **224** (2013) 101–109.
[20] H. Sarhadi, D.M. Tulett and M. Verma, A defender-attacker-defender approach to the optimal fortification of a rail intermodal terminal network. *J. Trans. Security* **8** (2015) 17–32.
[21] M.P. Scaparra and R.L. Church, An exact solution approach for the interdiction median problem with fortification. *Europ. J. Oper. Res.* **189** (2008) 76–92.
[22] M.P. Scaparra and R.L. Church, A bilevel mixed-integer program for critical infrastructure protection planning. *Comput. Oper. Res.* **35** (2008) 1905–1923.
[23] L.V. Snyder, Z. Atan, P. Peng, Y. Rong, A.J. Schmitt and B. Sinsoysal, Or/ms models for supply chain disruptions: A review. *IIE Trans.* **48** (2016) 89–109.
[24] Z.C. Taşkın, J.C. Smith, S. Ahmed and A.J. Schaefer, Cutting plane algorithms for solving a stochastic edge-partition problem. *Discrete Optim.* **6** (2009) 420–435.
[25] W. Yuan, L. Zhao and B. Zeng, Optimal power grid protection through a defender–attacker–defender model. *Reliability Eng. Syst. Safety* **121** (2014) 83–89.