A Branch and Price Algorithm for a Stackelberg Security Game

Felipe Lagos[a], Fernando Ordóñez[b], Martine Labbé[c]

[a] *Georgia Institute of Technology*
[b] *Universidad de Chile, fordon@dii.uchile.cl*
[c] *Université Libre de Bruxelles, mlabbe@ulb.ac.be*

Abstract

Mixed integer optimization formulations are an attractive alternative to solve Stackelberg Game problems thanks to the efficiency of state-of-the-art mixed integer algorithms. In particular, decomposition algorithms, such as branch and price methods, make it possible to tackle instances large enough to represent games inspired in real world domians.

In this work we focus on Stackelberg Games that arise from a security application and investigate the use of a new branch and price method to solve its mixed integer optimization formulation. We prove that the algorithm provides upper and lower bounds on the optimal solution at every iteration and investigate the use of stabilization heuristics. Our preliminary computational results compare this solution approach with previous decomposition methods obtained from alternative integer programming formulations of Stackelberg games.

1. Introduction

Stackelberg games model the strategic interaction between players, where one participant – the leader – is able to commit to a strategy first, knowing that the remaining players – the followers – will take this strategy into account and respond in an optimal manner. These games have been used to represent markets in which a participant has significant market share and can commit to a strategy [19], where government decides tolls or capacities in a transportation network [11], and of late have been used to represent the attacker-defender interaction in security domains [9]. These games are examples of bilevel optimization problems, which are in general non convex optimization problems that are difficult to solve.

In this work we focus on a specific class of Stackelberg games which we refer to as Stackelberg Security Games (SSG) that arise in security domains and have a particular payoff structure [21]. In a SSG, the security (or defender) behaves as the leader selecting a patrolling strategy first and the, possibly many attackers act as the follower, observing the defender's patrolling strategy and deciding where to attack. Such Stackelberg Security Game models have been used in the deployment of decision support systems with specialized algorithms in real security domain applications [9, 16, 17].

Recent work has developed efficient integer optimization solution algorithms for different variants of the SSGs [10, 6, 7, 8, 5]. In general terms these optimization problems are formulated with the defender committing to a mixed (randomized) strategy, whereas the attacker(s) conduct(s) surveillance of the defender mixed strategy and respond(s) with a pure strategy corresponding to an attack on a target. In addition, the number of actions of the defender can be exponential in size, with respect to the targets and defense resources, due to the combinatorics of using N resources to patrol m targets. This illustrates that to solve SSGs we have to address mixed integer optimization problems with exponential number of variables. Addressing the combinatorial size of defender strategies has led to both development of branch and price methods [10] and constraint generation methods [20]. There are, however, problem instances that arise from real security applications that still challenge existing solution methods. Here we investigate a new branch and price method developed for a novel formulation of Stackelberg games (MIPSG), introduced in [3]. This new formulation has been shown to provide tighter linear relaxations than other existing mixed integer formulations and to give the convex hull of the feasible integer solutions when there is only one follower.

We begin by introducing notation and describing the integer optimization formulations that have been considered previously in the next section. We also introduce the equivalent MIPSG formulation. In section 3 we present the column generation algorithm for the solution of the linear relaxation of MIPSG, along with a speed up that can be obtained by aggregating subproblems, and the existence of upper and lower bounds at every iteration. We also describe the branching strategies used in adapting this column generation to a Branch and Price method and how to apply dual stabilization techniques. We present our preliminary computational results in section 4 and provide concluding remarks in section 5.

2. Integer Optimization Formulations of SSG

In a Stackelberg security game we consider that the leader is the defender and the attacker (of possibly many types) is the follower. We let Θ be the set of possible attacker types and assume that p^θ corresponds to a known a-priori probability distribution that the defender is facing an attacker of type $\theta \in \Theta$. The attacker may decide to attack any one of a set of targets Q. The mixed strategy for the θ-th attacker is the vector of probabilities over this set of targets, which we denote as $\mathbf{q}^\theta = (q_j^\theta)_{j \in Q}$. The defender allocates up to N resources to protect targets, with $N < |Q|$. Each resource can be assigned to a patrol that protects multiple targets, $s \subseteq Q$, so the set of feasible patrols for one resource is a set $S \subseteq P(Q)$, where $P(Q)$ represents the power set of Q. The defender's pure strategies, or joint patrols, are combinations of up to N such patrols, one for each available resource. In addition we assume that in a joint patrol a target is covered by at most one resource. Let X denote the set of joint patrols, or defender strategies. A joint patrol $i \in X$, can be represented by the vector $\mathbf{a_i} = [a_{i1}, a_{i2}, ..., a_{i|Q|}] \in \{0,1\}^{|Q|}$ where a_{ij} represents whether or not target j is covered in strategy i. The defender's mixed strategy $\mathbf{x} = (x_i)_{i \in X}$ specifies the probabilities of selecting each joint patrol $i \in X$.

Both the leader and followers aim to maximize a linear utility function that averages the rewards of every combination of pure strategies weighted by the mixed strategies. If we let R_{ij}^{θ} and C_{ij}^{θ} denote the utility received by the defender (and the θ-th attacker) for having the defender conduct patrol i while the θ-th attacker strikes target j, then the defender and θ-th attacker utilities are given by

$$\begin{aligned} u_D(\mathbf{x}, (\mathbf{q}^{\theta})_{\theta \in \Theta}) &= \sum_{\theta \in \Theta} \sum_{i \in X} \sum_{j \in Q} p^{\theta} x_i q_j^{\theta} R_{ij}^{\theta} \\ u_A^{\theta}(\mathbf{x}, \mathbf{q}^{\theta}) &= \sum_{i \in X} \sum_{j \in Q} x_i q_j^{\theta} C_{ij}^{\theta} \ . \end{aligned}$$

The goal is to find the optimal mixed strategy for the leader, given the follower may know this mixed strategy when choosing its strategy. Stackelberg equilibria can be of two types: strong and weak, as described by [2]. We use the notion of Strong Stackelberg Equilibrium (SSE), in which the leader selects an optimal mixed strategy based on the assumption that the follower will choose an optimal response and will break ties in favor of the leader. In other words, following the formal definition of a SSE in [10], a pair of strategies form a SSE if they satisfy: **FO: MAKE PRECISE**

1. The leader (defender) plays a best-response
2. The follower (attacker) plays a best response
3. The follower breaks ties optimally for the leader

This can be formulated as the following bilevel optimization problem, where $\mathbf{e}$ is the vector of all ones of appropriate dimension:

$$\begin{aligned} \max \quad & u_D(\mathbf{x}, (\mathbf{q}^{\theta})_{\theta \in \Theta}) \\ \text{s.t.} \quad & \mathbf{e}^T \mathbf{x} = 1, \ \mathbf{x} \geq 0 \\ & \mathbf{q}^{\theta} = \text{argmax}_{\mathbf{g}} \{ u_A^{\theta}(\mathbf{x}, \mathbf{g}) \mid \mathbf{e}^T \mathbf{g} = 1, \ \mathbf{g} \geq 0 \} \quad \theta \in \Theta \ . \end{aligned}$$

Given that the inner optimization problem is a linear optimization problem over the $|Q|$ dimensional simplex, there always exists an optimal pure-strategy response for the attacker, so in the integer optimization formulations we present now we restrict our attention to the set of pure strategies for the attacker. As we see below, the optimality condition of the inner optimization problem can be expressed with linear constraints and integer variables when we make use of the fact that the followers respond with an optimal pure strategy. Although this leads to being able to use efficient mixed integer optimization machinery, the problem remains theoretically difficult as the problem of choosing the optimal strategy for the leader to commit to in a Bayesian Stackelberg game is NP-hard [4].

The payoffs for agents depend only on the target attacked, the adversary type and whether or not a defender resource is covering the target. Let the parameter Rd_j^{θ} denote the defender's utility, or reward, if $j \in Q$ is attacked by adversary $\theta \in \Theta$ when it is covered by a defender resource. If $j \in Q$ is not covered, the

defender receives a penalty Pd_j^θ. Likewise, the attacker's utilities are denoted by a reward Ra_j^θ when target j is attacked and not covered and penalty Pa_j^θ, when j is attacked while protected. Therefore if we let $j \in i$ denote when target $j \in Q$ is protected by patrol $i \in X$, then we consider the following reward structure

$$R_{ij}^\theta = \begin{cases} Rd_j^\theta & j \in i \\ Pd_j^\theta & j \notin i \end{cases} \qquad C_{ij}^\theta = \begin{cases} Pa_j^\theta & j \in i \\ Ra_j^\theta & j \notin i \end{cases} .$$

Alternatively the strategy i can be represented by a vector $\mathbf{a_i} \in \{0,1\}^{|Q|}$ such that $a_{ij} = 1$ when $j \in i$ or when $j \in \mathbf{a_i}$. Using this vector $\mathbf{a_i}$ we have

$$R_j^\theta(\mathbf{a_i}_j) := R_{ij}^\theta = Pd_j^\theta + a_{ij}\left(Rd_j^\theta - Pd_j^\theta\right) \qquad C_j^\theta(\mathbf{a_i}_j) := C_{ij}^\theta = Ra_j^\theta - a_{ij}\left(Ra_j^\theta - Pa_j^\theta\right) .$$

We assume adding coverage to target $j \in Q$ is strictly better for the defender and worse for the attacker. That is $Rd_j^\theta > Pd_j^\theta$ and $Ra_j^\theta > Pa_j^\theta$. Note that this does not necessarily mean zero-sum.

2.1. DOBBS and ERASER

Efficient and compact techniques for choosing the optimal strategies for Bayesian Stackelberg games have been a topic of active research from the work of [14, 13]. In particular, the DOBBS problem formulation below, introduced in [13], allows for a Bayesian Stackelberg game to be expressed compactly as a single mixed integer optimization problem.

$$\begin{aligned}
\max \quad & \sum_{i\in X}\sum_{\theta\in\Theta}\sum_{j\in Q} p^\theta z_{ij}^\theta R_{ij}^\theta & \\
& \sum_{i\in X}\sum_{j\in Q} z_{ij}^\theta = 1 & \forall\theta \\
& \sum_{i\in X} z_{ij}^\theta = q_j^\theta & \forall j,\theta \\
& 0 \le v^\theta - \sum_{i\in X} C_{ij}^\theta \sum_{k\in Q} z_{ik}^\theta \le (1-q_j^\theta)M & \forall j,\theta \\
& \sum_{j\in Q} z_{ij}^\theta = x_i & \forall i,\theta \\
& z_{ij}^\theta \in [0,1] & \forall i,j,\theta \\
& q_j^\theta \in \{0,1\} & \forall j,\theta \\
& x_i \in [0,1] & \forall i \\
& & \text{(DOBBS)}
\end{aligned}$$

Algorithms for large-scale SSG, using branch and price and fast upper bound generation framework are introduced in Jain et al. [6]. That work builds these algorithms from a more compact representation of

DOBBS, which has been named as ERASER (Efficient Randomized Allocation of Security Resources). This formulation does not use variable z_{ij}^{θ} obtaining a formulation that uses less variables overall but uses two sets of big M constraints. In the ERASER formulation below we present the notation for the dual variables of constraints (2)-(6) in parenthesis.

$$\max \quad \sum_{\theta \in \Theta} p^{\theta} d^{\theta} \tag{1}$$

$$d^{\theta} - \sum_{i \in X} x_i R_{ij}^{\theta} \leq (1 - q_j^{\theta}) M_1 \qquad \forall j, \theta \qquad (\beta_j^{\theta}) \tag{2}$$

$$a^{\theta} - \sum_{i \in X} x_i C_{ij}^{\theta} \leq (1 - q_j^{\theta}) M_2 \qquad \forall j, \theta \qquad (\alpha_j^{\theta}) \tag{3}$$

$$\sum_{i \in X} x_i C_{ij}^{\theta} \leq a^{\theta} \qquad \forall j, \theta \qquad (\sigma_j^{\theta}) \tag{4}$$

$$\sum_{i \in X} x_i = 1 \qquad (\delta) \tag{5}$$

$$\sum_{j \in Q} q_j^{\theta} = 1 \qquad \forall \theta \qquad (\pi^{\theta}) \tag{6}$$

$$q_j^{\theta} \in \{0, 1\} \qquad \forall j, \theta \tag{7}$$

$$x_i \geq 0 \qquad \forall i \tag{8}$$

$$\text{(ERASER)} \tag{9}$$

The M_1 and M_2 values are important for the ERASER performance, since their value helps determine how tight the linear relaxation is. Thus they must be chosen large enough so that the constraint does not eliminate a feasible solution but as small as possible to give the tightest linear relaxation. The values for M_1 and M_2 are as follows,

$$M_1 = \max_{j,\theta} R d_j^{\theta} - \min_{j,\theta} P d_j^{\theta} \tag{10}$$

$$M_2 = \max_{j,\theta} R a_j^{\theta} - \min_{j,\theta} P a_j^{\theta} \tag{11}$$

These values of M guarantee the problem keeps its feasible region unchanged. We will show this in the next section for similar constants in problem MIPSG.

When solving these equivalent formulations, one observes that the ERASER linear optimization relaxation is easier to solve than DOBBS, as it has less variables, however it gives a larger integrality gap. A branch and price method for ERASER is introduced in [6] and is shown to be efficient in practice and able to solve large SSG problems. This algorithm will be used as a comparison for the decomposition algorithm presented in this work.

A Branch and Price method is based on using a column generation method to solve the LP relaxation.

In this column generation for ERASER, the master would solve the problem considering only a few of the defender strategies $\bar{X} \subset X$, obtaining and optimal master primal and dual solutions. Then, the method tests whether a defender strategy variable x_i would enter the master problem by checking if its reduced cost is positive. Given the reduced master optimal dual variables indicated in (2) - (6), the reduced cost for strategy $i \in X$, also represented by the vector $\mathbf{v} \in \{0,1\}^{|Q|}$, is as follows,

$$\bar{c}_i \;=\; \bar{c}_{\mathbf{v}} \;=\; \sum_{j \in Q} \sum_{\theta \in \Theta} R_{ij}^{\theta} \beta_j^{\theta} + C_{ij}^{\theta}(\alpha_j^{\theta} - \sigma_j^{\theta}) - \delta \tag{12}$$

$$= \sum_{j \in Q} \sum_{\theta \in \Theta} R_{j}^{\theta}(v_j) \beta_j^{\theta} + C_{j}^{\theta}(v_j)(\alpha_j^{\theta} - \sigma_j^{\theta}) - \delta \tag{13}$$

Using this reduced cost expression we can define the subproblem for the ERASER's column generation. In this case, the subproblem also includes resources and patrol constraints. The branch and price framework is used for ERASER is the same that is used for the MIPSG model that will be presented in the next section. Thus, the only difference between two models implementation are the branch and price tree nodes.

2.2. Strong Integer Optimization Formulation

A novel equivalent formulation of this problem, a variation on the DOBBS formulation, was introduced in [3]. In contrast to ERASER, this model has tighter linear representation but requires more variables. Below we present this optimization problem, referred to as Model Integer Problem for Security Games (MIPSG).

$$\max \quad \sum_{i \in X} \sum_{\theta \in \Theta} \sum_{j \in Q} p^{\theta} z_{ij}^{\theta} R_{ij}^{\theta} \tag{14}$$

$$\sum_{i \in X} \sum_{j \in Q} z_{ij}^{\theta} = 1 \qquad \forall \theta \qquad (\pi^{\theta}) \tag{15}$$

$$\sum_{i \in X} z_{ij}^{\theta} = q_j^{\theta} \qquad \forall j, \theta \qquad (\sigma_j^{\theta}) \tag{16}$$

$$\sum_{i \in X} (C_{ij}^{\theta} - C_{ik}^{\theta}) z_{ij}^{\theta} \geq 0 \qquad \forall j, k, \theta \qquad (\alpha_{jk}^{\theta}) \tag{17}$$

$$\sum_{j \in Q} z_{ij}^{\theta} = x_i \qquad \forall i, \theta \qquad (\beta_i^{\theta}) \tag{18}$$

$$z_{ij}^{\theta} \in [0,1] \qquad \forall i, j, \theta \tag{19}$$

$$q_j^{\theta} \in \{0,1\} \qquad \forall j, \theta \tag{20}$$

$$x_i \in [0,1] \qquad \forall i \tag{21}$$

In the above description we also give the notation for the dual variables of the linear relaxation of MIPSG for each of the four sets of constraints. This is indicated by the variable in parenthesis on each constraint.

Proposition 2.1. *Problem MIPSG is equivalent to DOBBS*

Proof Problem MIPSG and DOBBS are the same except for one constraint. While in MIPSG the solution $(\mathbf{z}, \mathbf{x}, \mathbf{q})$ satisfies $\sum_{i \in X}(C_{ij}^{\theta} - C_{ik}^{\theta})z_{ij}^{\theta} \geq 0 \ \forall j, k, \theta$ in DOBBS the solution $(\mathbf{z}, \mathbf{x}, \mathbf{q}, \mathbf{v})$ satisfies $0 \leq v^{\theta} - \sum_{i \in X} C_{ij}^{\theta} \sum_{k \in Q} z_{ik}^{\theta} \leq (1 - q_j^{\theta})M \ \forall j, \theta$. If $q_h^{\theta} = 1$ then the DOBBS solution satisfies $\sum_{k \in Q} z_{ik}^{\theta} = z_{ih}^{\theta}$ and therefore

$$\sum_{i \in X} C_{ij}^{\theta} z_{ih}^{\theta} = \sum_{i \in X} C_{ij}^{\theta} \sum_{k \in Q} z_{ik}^{\theta} \leq v^{\theta} \leq \sum_{i \in X} C_{ih}^{\theta} \sum_{k \in Q} z_{ik}^{\theta} = \sum_{i \in X} C_{ih}^{\theta} z_{ih}^{\theta} \ ,$$

which is equivalent to the MIPSG constraint.

Let us now consider a solution for MIPSG. If $q_h^{\theta} = 1$ then let $v^{\theta} := \sum_{i \in X} C_{ih}^{\theta} z_{ih}^{\theta}$. Since now we also have $\sum_{k \in Q} z_{ik}^{\theta} = z_{ih}^{\theta}$ we have from the MIPSG constraint that

$$v^{\theta} = \sum_{i \in X} C_{ih}^{\theta} z_{ih}^{\theta} \geq \sum_{i \in X} C_{ij}^{\theta} \sum_{k \in Q} z_{ik}^{\theta} \ .$$

This satisfies the DOBBS constraints as the only tight right hand inequality is the one that defines v^{θ}. □

The results in [3] show that a solution that is feasible for the linear relaxation of the MIPSG formulation is a feasible solution for the linear relaxation of the DOBBS formulation. Furthermore, the linear relaxation of the MIPSG problem equals the convex hull of the feasible integer solutions when there is only one adversary.

The total amount of defender' strategies increase exponentially with the number of targets and resources. Without additional feasibility constraints, the size of the set of possible defender strategies equals $\binom{Q}{N}$. This leads to problems that are too big to solve in a standard computer. It is therefore necessary to find a way to generate only the strategies are used by the model.

3. Column Generation for MIPSG

A column generation method on MIPSG aims at solving the linear relaxation of the problem by gradually considering more variables associated to the large set of defender strategies. The linear relaxation of MIPSG relaxes the integrality constraints and considers variables that satisfy $0 \leq z_{ij}^{\theta}$, $q_j^{\theta} \in \mathbb{R}$ and $x_i \in \mathbb{R}$. Note that since $\sum_{i \in X} \sum_{j \in Q} z_{ij}^{\theta} = 1$ we still have that $z_{ij}^{\theta}, q_j^{\theta}, x_i \in [0, 1]$. Below we give the dual problem of the linear relaxation of the MIPSG problem, using the dual variables identified in the statement of the MIPSG problem:

$$\min \quad \sum_{i \in \Theta} \pi^{\theta} \tag{22}$$

$$p^{\theta} R^{\theta}_{ij} \leq \pi^{\theta} + \sigma^{\theta}_{j} + \beta^{\theta}_{i} + \sum_{k \in Q} (C^{\theta}_{ij} - C^{\theta}_{ik}) \alpha^{\theta}_{jk} \qquad \forall i, j, \theta \tag{23}$$

$$\sigma^{\theta}_{j} = 0 \qquad \forall j, \theta \tag{24}$$

$$\sum_{\theta \in \Theta} \beta^{\theta}_{i} = 0 \qquad \forall i \tag{25}$$

$$\alpha^{\theta}_{jk} \leq 0 \qquad \forall j, k, \theta \tag{26}$$

In the LP relaxation of MIPSG the constraint $\sum_{i \in X} z^{\theta}_{ij} = q^{\theta}_{j}$ becomes redundant as it defines the value of q^{θ}_{j}, but this variable no longer has to be integer variable. This fact is reflected in that the corresponding dual variable σ^{θ}_{j} has a value of zero.

We now outline the column generation procedure that we propose for MIPSG. We begin by solving a version of the MIPSG problem in which only a set $\bar{X} \subset X$ of defender strategies are considered. This means that variables z^{θ}_{ij} and x_i with $i \notin \bar{X}$ are not considered in the master problem and assumed fixed at 0. After solving the reduced master problem, the method looks for profitable strategies in $X \setminus \bar{X}$. To identify a profitable strategy $i \in X$ we should look for a variable z^{θ}_{ij} or x_i with positive reduced cost. From linear programming duality we have that a positive reduced cost corresponds to a violated dual constraint. Indeed, the process of column generation in a problem is equivalent to generating the corresponding dual constraints in the dual problem [1]. Therefore to identify which variables (and corresponding strategies $i \in X$) to add to the master, our method requires we identify constraints, either (23) or (25), in this dual problem that are not being satisfied at the current dual optimal solution. Once the new variables are added to the master, we re-optimize the master problem until there are no violated dual constraints.

However, the generic column generation method described above cannot be implemented. As written, to determine the reduced cost of a variable z^{θ}_{ij} or x_i, with $i \notin \bar{X}$, we need to know the dual variable β^{θ}_{i}. This is the dual variable corresponding to constraint (18) that is not present if strategy $i \notin \bar{X}$ is not considered in the reduced master and therefore not defined.

We now address this difficulty by introducing the following problem which is based on the dual problem

of MIPSG.

$$\max_{v,e} \quad \sum_{\theta} f^{\theta} \tag{27}$$

$$f^{\theta} \leq p^{\theta} R(v_j)^{\theta} - \pi^{\theta} + M^{\theta}(1 - e_j^{\theta}) - \sum_{k \in Q} (C(v_j)^{\theta} - C(v_k)^{\theta}) \alpha_{jk}^{\theta} \qquad \forall j, \theta \tag{28}$$

$$\sum_{r \in |S|} u_r \leq N \tag{29}$$

$$\sum_{j \in Q} e_j^{\theta} = 1 \qquad \forall \theta \tag{30}$$

$$\sum_{r \in |S|} t_{jr} u_r = v_j \qquad \forall j \tag{31}$$

$$v_j \in \{0,1\},\ e_j^{\theta} \in \{0,1\} \qquad \forall j \tag{32}$$

$$u_r \in \{0,1\} \qquad \forall r \tag{33}$$

The resulting solution is a vector of joint schedules that are chosen from the set S of possible options. Let u_r be a binary variable that is 1 if the schedule $r = 1, ..., |S|$ is used for the new strategy $\mathbf{v}$ and 0 otherwise. These u_r must sum up to N, which is the number of resources available to assign. The parameter t_{jr} indicates which targets $j \in Q$ are covered by the schedule $r \in |S|$, i.e. it is a binary representation of the set schedule $s_r \in S$. Finally, the variable e_j^{θ} is just an auxiliary variable that defines which is the best target j for each θ.

For dealing with β_i^{θ} we use that the sum of these variables over $\theta \in \Theta$ is zero. If the subproblem finds a new column with optimal objective value less than zero, then there exists β_i^{θ} that satisfy the dual constraints. Thus, this allows us to not include these variables in the model and to determine when the column generation finishes.

In this model, the utility of defender from strategy $\mathbf{v}$ is $R(v_j)^{\theta} = Pd_j^{\theta} + v_j(Rd_j^{\theta} - Pd_j^{\theta})$, and the utility of the attacker is $C(v_j)^{\theta} = Ra_j^{\theta} - v_j(Ra_j^{\theta} - Pa_j^{\theta})$. We also have chosen an appropriate M^{θ} for all $\theta \in \Theta$, such that the constraint would work with every arbitrary parameters and be as tight as possible. Moreover, we want a value for M^{θ} small that does not make the subproblem infeasible.

Proposition 3.1. *For all $\theta \in \Theta$, the smallest value of M^{θ} that does not eliminate any feasible patrol is given by*

$$M^{\theta} = p^{\theta}(\max_j Rd_j^{\theta} - \min_j Pd_j^{\theta}) - 2|Q|\ \min_{jk} \alpha_{jk}^{\theta}\ (\max_j Ra_j^{\theta} - \min_j Pa_j^{\theta}) \qquad \forall \theta \tag{34}$$

Proof FO: FIX THIS The M^{θ} are the minimum value the constraint (28) can get plus M^{θ} be greater than the maximum value, for all $\theta \in \Theta$. For every $\mathbf{v}$ the model has, this constraint is guaranteed will be satisfied. We need to prove that given $j \in Q$ and $\theta \in \Theta$, the values for M^{θ} need to satisfy the next equation:

$$M^\theta + \min_j \left\{ p^\theta R(v_j)^\theta - \pi^\theta - \sum_{k\in Q}(C_{ij}^\theta - C_{ik}^\theta)\alpha_{jk}^\theta \right\} \geq \max_j \left\{ p^\theta R(v_j)^\theta - \pi^\theta - \sum_{k\in Q}(C_{ij}^\theta - C_{ik}^\theta)\alpha_{jk}^\theta \right\} \qquad \forall\theta \tag{35}$$

For all θ we have,

$$\begin{aligned}
&\max_j \left\{ p^\theta R(v_j)^\theta - \sum_{k\in Q}(C_{ij}^\theta - C_{ik}^\theta)\alpha_{jk}^\theta \right\} - \min_j \left\{ p^\theta R(v_j)^\theta - \sum_{k\in Q}(C_{ij}^\theta - C_{ik}^\theta)\alpha_{jk}^\theta \right\} \\
\leq & p^\theta(\max_j Rd_j^\theta - \min_j Pd_j^\theta) - 2\max_j \sum_{k\in Q}(C_{ij}^\theta - C_{ik}^\theta)\alpha_{jk}^\theta \\
\leq & p^\theta(\max_j Rd_j^\theta - \min_j Pd_j^\theta) - 2|Q|\max_j Ra_j^\theta \ \min_{jk}\alpha_{jk}^\theta + 2|Q|\max_j \max_k \left\{C_{ik}^\theta \alpha_{jk}^\theta\right\} \\
\leq & p^\theta(\max_j Rd_j^\theta - \min_j Pd_j^\theta) - 2|Q| \ \min_{jk}\alpha_{jk}^\theta \ (\max_j Ra_j^\theta - \min_j Pa_j^\theta) \\
\leq & M^\theta
\end{aligned}$$

□

This subproblem searches the maximum joint reduced cost of MIPSG for a new strategy, the optimal solution of the problem. Once we have the best strategy to add, we check whether the objective function, $\bar{f} = \sum_\theta f^\theta$ is positive. In that case, that strategy must be incorporated into the master problem, because that column is violating a constraint in the dual. In order to show this statement, we define F_{ij}^θ.

Let F_{ij}^θ a value such as takes for all i, j, θ and π^θ, α_{jk}^θ fixed, the follow expression:

$$F_{ij}^\theta = p^\theta R_{ij}^\theta - \pi^\theta - \sum_{k\in Q}(C_{ij}^\theta - C_{ik}^\theta)\alpha_{jk}^\theta \tag{36}$$

It is easy to check that if $\bar{f} = \max \sum_\theta f^\theta = \max \sum_\theta F_j^\theta$ is greater than zero, then we can not satisfy the dual. In fact,

$$\begin{aligned}
& p^\theta R_{ij}^\theta \leq \pi^\theta + \sigma_j^\theta + \beta_i^\theta + \sum_{k\in Q}(C_{ij}^\theta - C_{ik}^\theta)\alpha_{jk}^\theta \\
& p^\theta R_{ij}^\theta - \pi^\theta - \sigma_j^\theta - \sum_{k\in Q}(C_{ij}^\theta - C_{ik}^\theta)\alpha_{jk}^\theta \leq \beta_i^\theta \\
& \sum_\theta \left(p^\theta R_{ij}^\theta - \pi^\theta - \sigma_j^\theta - \sum_{k\in Q}(C_{ij}^\theta - C_{ik}^\theta)\alpha_{jk}^\theta \right) \leq \sum_\theta \beta_i^\theta \\
& \bar{f} = \sum_\theta F_j^\theta \leq \sum_\theta \beta_i^\theta = 0 \\
& \bar{f} \leq 0
\end{aligned}$$

In this set of equations we are using that $\sigma_j^\theta = 0$, from the dual equation (24), when $q_j^\theta \in \mathbb{R}$, i.e., when there are no integer conditions.

We show the condition we need in order to determinate whether we can terminate the column generation or not. This condition is sufficient to guarantee optimality.

Proposition 3.2. *If $\bar{f} = \max \sum_{\theta\in\Theta} f^\theta = \sum_{\theta\in\Theta} \max F_{ij}^\theta \leq 0$ for a new strategy i in the subproblem, then there is no new column that mus be included to the master problem. This problem does not need more columns to be solved optimally.*

Proof The first thing we should notice is the β_i^θ values can take arbitrary values because their primal constraint is always feasible. Indeed, $\sum_j z_{ij}^\theta = x_i$ is true for all strategy in or out of the master problem at any iteration. Hence, if we find some β_i^θ arbitrary that satisfy the dual problem for a non positive reduced cost strategy i, then it is not necessary to include that strategy.

In fact, we know that in the dual problem we have to satisfy:

$$F_{ij}^\theta \leq \beta_i^\theta \qquad \forall j, \theta \tag{37}$$

$$\sum_{\theta\in\Theta} \beta_i^\theta = 0 \tag{38}$$

We can take an arbitrary $\bar{\theta} \in \Theta$ and set $\beta_i^{\bar{\theta}}$ such that $\beta_i^{\bar{\theta}} = -\sum_{\theta\in\Theta\setminus\{\bar{\theta}\}} \max F_{ij}^\theta$, and for all remaining $\theta \in \Theta\setminus\{\bar{\theta}\}$ set $\beta_i^\theta = \max F_{ij}^\theta$.

These $\beta_i^{\bar{\theta}}$ for strategy i satisfies:

$$\begin{aligned}
&\bar{f} = \sum_{\theta\in\Theta} \max F_{ij}^\theta \leq 0 \\
&\max F_{ij}^{\bar{\theta}} + \sum_{\theta\in\Theta\setminus\{\bar{\theta}\}} \max F_{ij}^\theta \leq 0 \\
&\max F_{ij}^{\bar{\theta}} \leq \beta_i^{\bar{\theta}}
\end{aligned}$$

Using this last inequality it is easy to verify that for all j, θ, the values we have set for β_i^θ meets the first set of constraints in (37) and also $\sum_{\theta\in\Theta} \beta_i^\theta = 0$. Therefore, when $\bar{f} \leq 0$ we have the conditions necessary to finish the column generation. □

3.1. Upper and Lower Bounds

Good upper and lower bounds can make the column generation and the branching process much more efficient. Moreover, if we set optimality tolerances, then tight gaps lead to faster running times we can take advantage of. We therefore are very interested in being able to bound well the distance between the optimal and the current solution.

Let L be the Lagrangian relaxation of MIPSG obtained by relaxing the adversaries best response constraint with a Lagrangian multiplier of α_{jk}^{θ}. This relaxation is therefore a function of α and will be updated every step, providing an upper bound for our problem. Next, we could write this function as follows,

$$\begin{aligned}
L(\alpha) \quad = \max_{\mathbf{z},\mathbf{x}} \quad & \sum_{i\in X}\sum_{\theta\in\Theta}\sum_{j\in Q}\left(p^{\theta}R_{ij}^{\theta} - \sum_{k\in Q}(C_{ij}^{\theta} - C_{ik}^{\theta})\alpha_{jk}^{\theta}\right) z_{ij}^{\theta} \\
& \sum_{i\in X}\sum_{j\in Q} z_{ij}^{\theta} = 1 \qquad\qquad \forall\theta \\
& \sum_{j\in Q} z_{ij}^{\theta} = x_i \qquad\qquad \forall i,\theta \\
& z_{ij}^{\theta} \in [0,1] \qquad\qquad \forall i,j,\theta \\
\leq & \sum_{\theta\in\Theta} \max_{i\in X, j\in Q}\left(p^{\theta}R_{ij}^{\theta} - \sum_{k\in Q}(C_{ij}^{\theta} - C_{ik}^{\theta})\alpha_{jk}^{\theta}\right) \\
= & \sum_{\theta\in\Theta}\pi^{\theta} + \sum_{\theta\in\Theta}\max_{j\in Q, i\in X} F_{ij}^{\theta}
\end{aligned}$$

The inequality in the second line is because we removed the constraints that involved the x_i variables. This further relaxed problem gives a value greater than the $L(\alpha)$. In this way, we know in every iteration the optimal value is greater than $\sum_{\theta\in\Theta}\pi^{\theta}$ and less than $\sum_{\theta\in\Theta}\pi^{\theta} + \sum_{\theta\in\Theta}\max_{j\in Q,i\in X} F_{ij}^{\theta}$, therefore, the gap is $\bar{f} = \sum_{\theta\in\Theta}\max_{j\in Q,i\in X} F_{ij}^{\theta}$, the objective function of the subproblem. We have also shown by an alternative way that when $\bar{f} \leq 0$ we have reach the optimal solution with the column generation.

So far, we have described how to identify new columns in our problem when the integrality constraints of the primal are relaxed, i.e., when $q_j^{\theta} \in \mathbb{R}$. We have not discussed how to generate columns (how to conduct pricing) when branching starts. Fortunately, for this problem we are able to show that this branch and price is very straightforward for MIPSG.

3.2. Branching scheme

When the variable $\mathbf{q}$ is relaxed, we solve the master problem and the subproblem until the optimal solution is reached. However, the $\mathbf{q}$ variable must be integer for the general case, so we implement a branch and price scheme, a very standard approach. At every node we solve the relaxed problem using column generation and then, if any of the integer variables is fractional, we branch on it.

In the dual MIPSG model we posed at (22) - (26), there is a dual variable σ_j^{θ} relate to q_j^{θ} primal variable. If this primal variable is relaxed, the dual variable is equal to zero. However, when we branch on q_j^{θ}, it is no longer zero, and then it has to be included into the subproblem. Hence, as we branch tree's node in the branch and price some of these σ_j^{θ} become active, which changes the subproblem.

The condition for terminating the column generation we state says $\bar{f} \leq 0$. In the branched nodes this condition still holds, but the σ_j^{θ} variables active might change the value of $\bar{f}$.

The strategy we follow to implement the branch and price is going through the tree from top to bottom, instead of left to right along nodes. In other words, we quickly find integer solutions, lower bounds of the problem and then check other branches. We also identify the variables those values are close to 0.5 in the first place, because they might be more decisive in the objective function.

3.3. Column Generation Stabilization by Dual Price Smoothing

Solving the LP relaxation of a model using column generation has shown some drawbacks when converging solutions. Vanderbeck [18] listed some of the typical issues that arise when implementing a column generation due to dual variables. Among them are: (i) a slow convergence, a phenomenon called the *tailing-off effect*; (ii) first iterations produce irrelevant columns; (iii) the restricted master solution value keeps constant for several iterations; (iv) dual values that change considerably from one iteration to another; (v) Langrangian dual bounds do not convergence monotonically.

Some techniques have been developed in order to deal with these undesirable converging behaviour. Lubbbecke and Desrosiers [12] described the three most important methods, Weighted Danzing-Wolfe decomposition, Trust region method and Stabilization approach using primal and dual strategies. We will use the first one because it has shown a very good performance solving classical problems and it is easy to implement.

Pessoa et al. [15] give a very complete description and analysis of weighting method. They also show how this algorithm improves the runtime for solving typical large problems, such as, Machine Scheduling, Bin Packing and Capacitated Vehicle Routing. Some instances even reduces their solving time by a factor of 5. They also develop a smoothing technique which uses a hybridization of column generation with sub-gradient method.

The smoothing technique in its simpler version is as follows. Let $\mathbf{y^t}$ be the dual solution at stage t and $0 \leq \alpha \leq 1$ be a weighting parameter, then the dual $\tilde{\mathbf{y}}^{\mathbf{t}}$ for the pricing problem for next iteration is,

$$\tilde{\mathbf{y}}^{\mathbf{t}} = \alpha \hat{\mathbf{y}}^{\mathbf{t}} + (1 - \alpha)\mathbf{y^t} \tag{39}$$

where $\hat{y}$ is the dual associated to the best dual solution so far.

For this smoothing scheme we can have three situations: (i) updated duals give us a positive reduced cost column; (ii) we get a new dual bound and so we improve the optimality gap; or (iii) it occurs a mis-pricing and the smoothed prices of the next iterate get closer to $\mathbf{y^t}$. A mis-pricing happens when the subproblem finds a solution with reduced cost non positive with $\hat{\mathbf{y}}^{\mathbf{t}}$, which is positive when we use $\mathbf{y^t}$. Under this conditions, Pessoa et al. [15] proved that in finite number of iterations column generation with smoothing pricing converges to an optimal solution.

A fix α leads a smoothing scheme that convergences after some iterations. A better approach considerer an auto-adaptive α, which increases and decreases as upper-lower bound gap changes. In [15] they propose

an algorithm for this situation, which is based on a sub-gradient information and a mis-pricing sequence for a given initial α.

Algorithm 1: Mis-pricing sequence

$k = 1$; $\mathbf{y^0} = \hat{\mathbf{y}}^\mathbf{t}$;
$\bar{\alpha} = \alpha$;
while $\bar{\alpha} \neq 0$ **do**
  $\bar{\alpha} = [1 - k \cdot (1 - \alpha)]^+$;
  $\hat{\mathbf{y}}^\mathbf{t} = \bar{\alpha}\hat{\mathbf{y}}^\mathbf{t} + (1 - \bar{\alpha})\mathbf{y^t}$;
  $k = k + 1$;
  Solve subproblem using $\hat{\mathbf{y}}^\mathbf{t}$;
  if *mis-pricing doesn't occurs* **then**
    Let $t = t + 1$, solve the master and continue sub-gradient algorithm;

Algorithm 2: Sub-gradient routine

$\alpha^0 = \alpha$; $t = 0$;
while *reduced cost* > 0 **do**
  Solve the master problem;
  Call subproblem with $\hat{\mathbf{y}}^\mathbf{t} = \bar{\alpha}\hat{\mathbf{y}}^\mathbf{t} + (1 - \bar{\alpha})\mathbf{y^t}$;
  if *mis-pricing occurs* **then**
    Start the mis-pricing schedule (Algorithm 1);
  else
    Let $\mathbf{g^t}$ be the sub-gradient;
    if $\mathbf{g^t}(\hat{\mathbf{y}}^\mathbf{t} - \mathbf{y^t}) > 0$ **then**
      $\alpha_t = f_{inc}(\alpha)$;
    else
      $\alpha_t = f_{dec}(\alpha)$;
  $t = t + 1$;

Function for increasing and decreasing α are: $f_{inc}(\alpha_t) = \alpha_t + (1 - \alpha_t) \cdot 0.1$, while $f_{dec}(\alpha_t) = \alpha_t / 1.1$ if $\alpha_t \in [0.5, 1)$ and $f_{dec}(\alpha_t) = \max\{0, \alpha_t - (1 - \alpha_t) \cdot 0.1\}$ otherwise. The $\mathbf{g^t}$ sub-gradient value for a given dual $\mathbf{y^t} = [\pi, \alpha]$ solution is computed as follows,

$$\mathbf{g^t y^t} = \sum_{\theta \in \Theta} M^\theta (1 - e_j^\theta) - \pi^\theta - \sum_{k \in Q} C(v_j))^\theta - C(v_k)^\theta) \alpha_{jk}^\theta \tag{40}$$

The values of e_j^θ and v_j for $\mathbf{g^t}$ are those we find through the subproblem at iteration t.

3.4. Greedy Algorithm for Subproblem

While the master problem is choosing the best mixed strategy for the leader, the subproblem is finding the best schedules to include for a new column. In each iteration we solve the master problem, we get new duals and then we use them for the subproblem. This subproblem has a limited number of resources to add, probably those with highest value for the objective function.

Henceforth, it is reasonably to think this problem as a knapsack one, since we need to find the most profitable schedules for a given capacity. A greedy algorithm is a good approach, we describe it in Algorithm 4. We use this algorithm as a speed up additional routine for the column generation.

In the greedy algorithm, we basically try all the schedules over set S and we finally add to the new strategy only those with the highest reduced cost. We repeat this process until no more resources can be assigned. Then, we return the best strategy and its reduced cost.

Algorithm 3: Column Generation Greedy

```
1  Include the initial set of basic strategies;
2  while reduced cost > 0 do
3      Solve the Master problem and get the
       new dual variables;
4      Get reduced cost from Greedy
       subroutine;
5      if reduced cost > 0 then
6          Include the new column;
7      else
8          Get reduced cost from subproblem;
9          if reduced cost > 0 then
10             Include the new column
```

Algorithm 4: Greedy subroutine

```
1  Let v be a new strategy vector;
2  reduced cost = value(v);
3  for i = 1; i ≤ N; i = i + 1 do
4      index = 0; best = −∞;
5      for r = 1; r ≤ |S|; r = r + 1 do
6          Set schedule s_r temporally to
           vector v;
7          if best < value(v) then
8              best = value(v);
9              index = r;
10     Include schedule index into v;
11     reduced cost = value(v);
12 return reduced cost and vector v;
```

However, this greedy does not guarantee we are going to find the optimal solution. Indeed, if it returns a non positive reduced cost when it finishes, we cannot be sure the optimal solution is reached. Thus, the subproblem model we describe in (27) - (33) it must be used for checking optimality at the last step. The algorithm we implement is the Algorithm 3.

4. Computational Results

We randomly generate a set of instances to be solved for each model. The models we consider for the computational experiments are MIPSG and ERASER, both using column generation. In addition we solve these instances with the greedy algorithm and the dual stabilization approach presented. In summary our computational results will compare four solution methods.

As base case we have 70 different zones or targets, one adversary, 5 resources for allocating, 600 schedules to choose and each of this schedules covers 5 targets. In total we study 1000 instances, given by the four dimensions of factors in the following way,

- Zones: 50, 60, 70, 80, 90, 100
- Schedules: 200, 400, 600, 800, 1000

- Resources: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
- Cover targets per schedule: 2, 3, 4, 5

Then, we can study how each method responds when we change the parameters of the model in one of the dimensions. We generate random instances, by randomly generating the reward and penalties for both the leader and the attacker and also generating a random set of initial patrols. We generate the reward values for both the defender and the attacker using a log-normal distribution. In this way, reward is always positive. For the penalty, we let it be minus the value obtained for the reward. This way the penalty is always a negative value. This guarantees that the penalty is less than reward for every attacker and defender. On the other hand, the patrol set is sampled from a discrete random variable in such a way we do not have two patrols that are the same. We used the Log-normal to generate the rewards because we can easily modify the samples obtained and because this distribution has two parameters that provide concise expression for the coefficient of variation. This coefficient is set to 2 and it can be easily computed as $cv = \frac{\text{standard deviation}}{\text{mean}}$. A coefficient of variation of 2 corresponds to a large input variability.

We use *CPLEX 12.4* and maximum runtime is set to 2 hours.

4.1. Algorithm Comparison

In Table 1 we have computed the average number of columns generated by each algorithm for each resource number. The parameters we have considered correspond to case base's parameters: 70 targets, 600 feasible schedules and 5 targets covered by schedule.

Resources	MISPG-C	ERASER-C	GREEDY	STAB
2	-	130	-	-
4	474	326	312	495
6	-	874	-	-
8	450	378	386	242
10	230	135	219	225

Table 1: Number of columns: Targets: 70, Schedules: 600, T/S: 5

The only column generation algorithm that can find one or more solutions for every instance is ERASER the other algorithms cannot solve any instance for 2 and 6 resources. From this table we can see that the number of columns is not directly related to the number of resources. The ERASER column shows that for 2 and 10 resources the number of needed columns is not too large, about 130, but for 4 the algorithm needs many more. We also see that the number of columns for MIPSG-c and MIPSG stab is more than the greedy version generates.

In Figure 1, 2 and 3 we have plotted the average solving running time for each algorithm. When we vary the number of resources is not clear which is the fastest solving method, however it shows that more

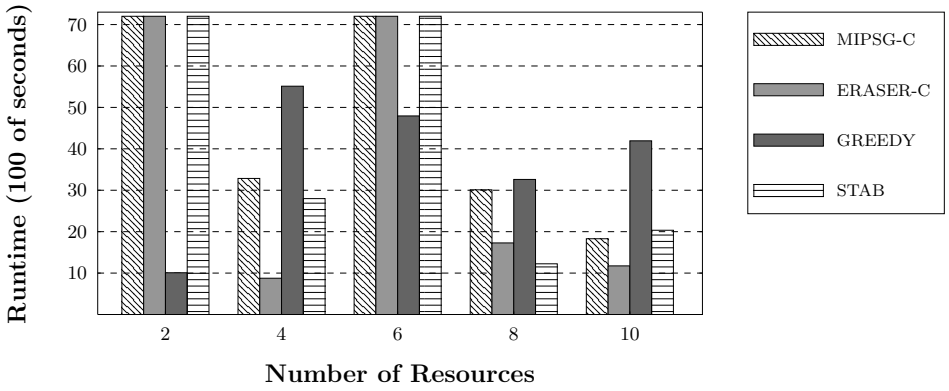

Figure 1: Comparison (Targets: 70, Schedules: 600, T/S: 5)

resources make the problem faster to solve. With 8 or 10 resources the average time is about the half of the cases with fewer number.

When we plot the number of available schedules, the running time seems stable.

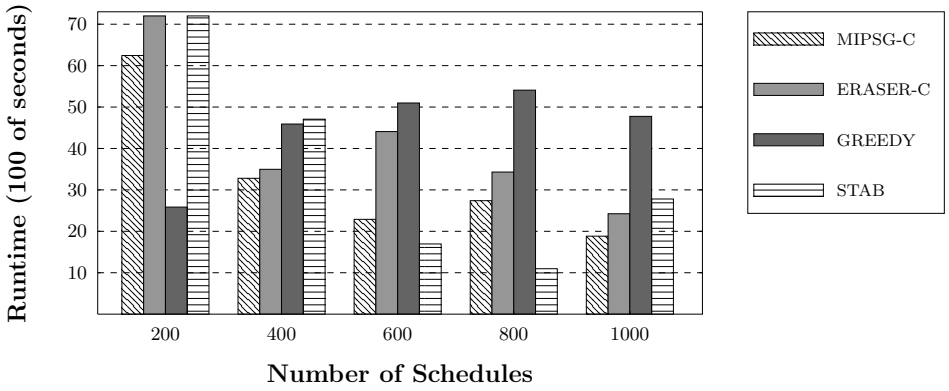

Figure 2: Comparison (Targets: 70, Resources: 5, T/S: 5)

In Figure 3 we can see how the number of targets affects the total solving time. More targets make the problem harder to solve.

4.2. MIPSG Column Generation Results

In Table 2 we have the average number of columns generated by MIPSG for the case base considering different number of resources and targets/schedule. As before, not all cases can be solved, for some parameters the algorithm cannot solve any of the instances. In particular, for 3 T/S, there are three cases where no instance was able to be solved.

In Figures

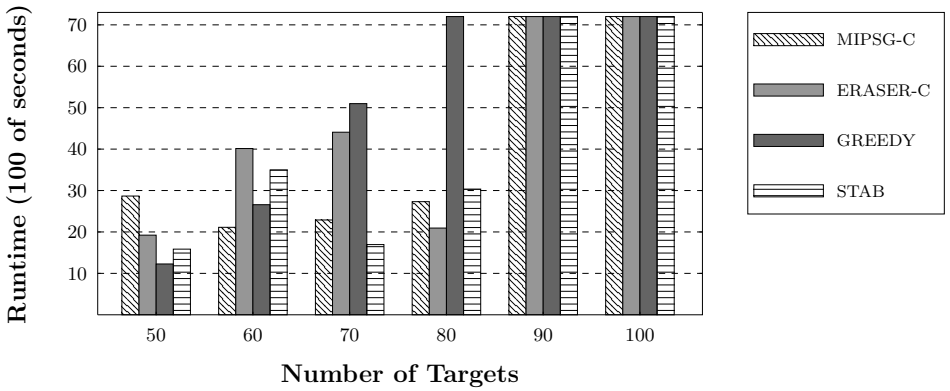

Figure 3: Comparison (Schedules: 600, Resources: 5, T/S: 5)

Resources	2 T/S	3 T/S	4 T/S	5 T/S
2	671	659	627	-
4	112	-	-	474
6	180	246	-	-
8	547	-	679	450
10	-	-	286	230

Table 2: Number of columns: Targets: 70, Schedules: 600 for MIPSG-C

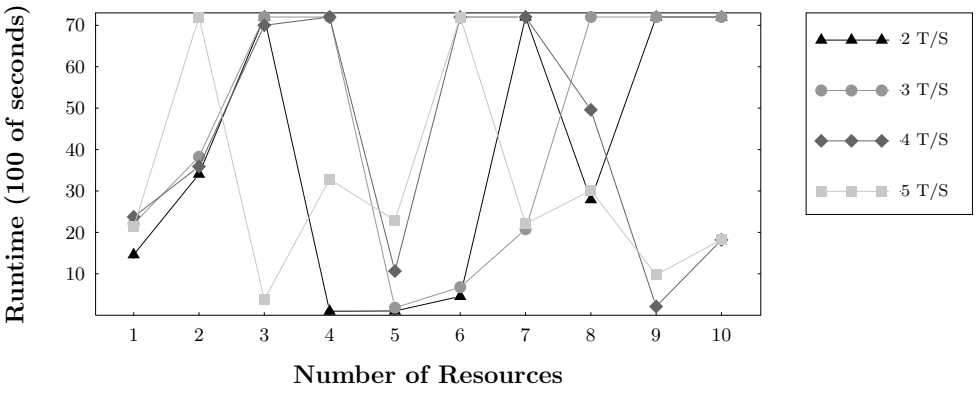

Figure 4: Comparison (Schedules: 600, Targets: 70) for MIPSG-C

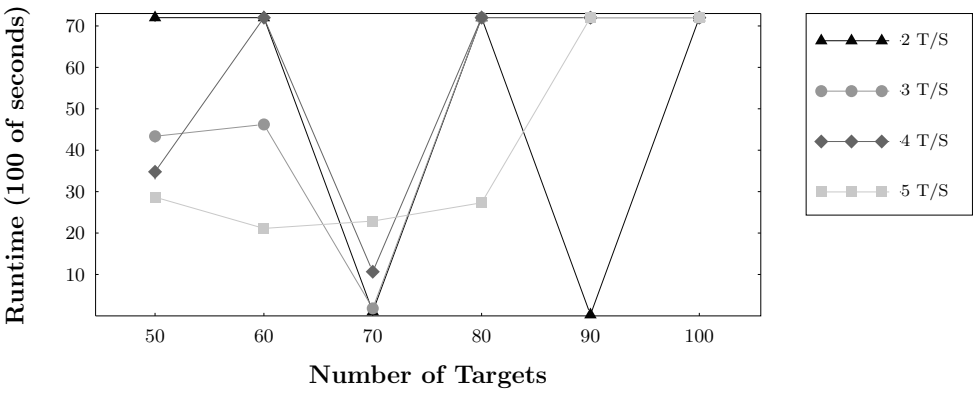

Figure 5: Comparison (Schedules: 600, Resources: 5) for MIPSG-C

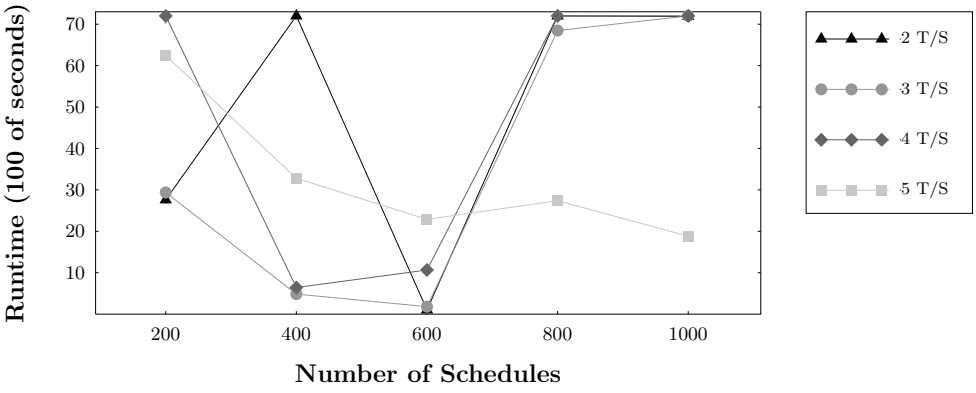

Figure 6: Comparison (Targets: 70, Resources: 5) for MIPSG-C

5. Conclusions

[1] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[2] M. Breton, A. Alj, and A. Haurie. Sequential stackelberg equilibria in two-person games. *Journal of Optimization Theory and Applications*, 59(1):71–97, 1988.

[3] C. Casorrán-Amilburu, B. Fortz, M. Labbé, and F. Ordóñez. Novel formulations for stackelberg security games. Working paper, Département d'Informatique, Université Libre de Bruxelles, 2014.

[4] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *Proc. of the 7th ACM conference on Electronic commerce*, pages 82–90, 2006.

[5] D. S. Hochbaum, C. Lyu, and F. Ordóñez. Security routing games with multivehicle chinese postman problem. *Networks*, 64(3):181–191, 2014.

[6] M. Jain, E. Kardes, C. Kiekintveld, F. Ordóñez, and M. Tambe. Security games with arbitrary schedules: A branch and price approach. In *Proc. of The 24th AAAI Conference on Artificial Intelligence*, pages 792–797, 2010.

[7] M. Jain, C. Kiekintveld, and M. Tambe. Quality-bounded solutions for finite bayesian Stackelberg games: scaling up. In *Proc. of The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2011.

[8] M. Jain, D. Korzhyk, O. Vanek, M. Pechoucek, V. Conitzer, and M. Tambe. A double oracle algorithm for zero-sum security games on graphs. In *Proc. of The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2011.

[9] M. Jain, J. Tsai, J. Pita, C. Kiekintveld, S. Rathi, M. Tambe, and F. Ordóñez. Software assistants for randomized patrol planning for the LAX airport police and the federal air marshal service. *Interfaces*, 40:267–290, 2010.

[10] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, M. Tambe, and F. Ordóñez. Computing optimal randomized resource allocations for massive security games. In *Proc. of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 689–696, 2009.

[11] M. Labbé, P. Marcotte, and G. Savard. A bilevel model of taxation and its application to optimal highway pricing. *Management Science*, 44(12-part 1):1608–1622, 1998.

[12] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[13] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordóñez, and S. Kraus. Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. In *Proc. of The 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 895–902, 2008.

[14] P. Paruchuri, J. P. Pearce, M. Tambe, F. Ordóñez, and S. Kraus. An efficient heuristic approach for security against multiple adversaries. In *Proc. of The 6th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 311–318, 2007.

[15] A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck. In-out separation and column generation stabilization by dual price smoothing. In *Experimental Algorithms*, pages 354–365. Springer, 2013.

[16] J. Pita, M. Tambe, C. Kiekintveld, S. Cullen, and E. Steigerwald. GUARDS - game theoretic security allocation on a national scale. In *Proc. of The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 37–44, 2011.

[17] E. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, and G. Meyer. PROTECT: A deployed game theoretic system to protect the ports of the United States. In *Proc. of The 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012.

[18] F. Vanderbeck. Implementing mixed integer column generation. In *Column Generation*, pages 331–358. Springer, 2005.

[19] H. von Stackelberg. *Market Structure and Equilibrium*. Springer-Verlag, Berlin Heidelberg, 2011. Translation from the German language edition: "Marktform und Gleichgewicht" by H. von Stackelberg, Springer-Verlag Wien New York 1934.

[20] R. Yang, A. X. Jiang, M. Tambe, and F. Ordóñez. Scaling-up security games with boundedly rational adversaries: A cutting-plane approach. In *Proc. of the 24nd International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[21] Z. Yin, D. Korzhyk, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. Nash in security games: interchangeability, equivalence, and uniqueness. In *Proc. of The 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1139–1146, 2010.