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, corresponding author*
[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 then, 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 and the attacker(s) responding with a pure strategy after conducting surveillance of the defender's mixed strategy. A mixed strategy refers to a probability distribution over the possible actions while a pure strategy corresponds to selecting one of the possible actions. In this SSG, the defender mixed strategies are probability distributions over possible patrolling strategies, while the attacker's pure strategy corresponds to selecting a specific target to attack. 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^{θ}_{ij} and C^{θ}_{ij} 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

$$\begin{aligned} u_D(\mathbf{x}, (\mathbf{q}^{\theta})_{\theta \in \Theta}) &= \sum_{\theta \in \Theta} \sum_{i \in X} \sum_{j \in Q} p^{\theta} x_i q^{\theta}_j R^{\theta}_{ij} \\ u^{\theta}_A(\mathbf{x}, \mathbf{q}^{\theta}) &= \sum_{i \in X} \sum_{j \in Q} x_i q^{\theta}_j C^{\theta}_{ij} \ . \end{aligned}$$

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 $\mathbf{x}$ and $(\mathbf{q}^{\theta}(\mathbf{x}))_{\theta \in \Theta}$ form a SSE if they satisfy:

1. The leader (defender) maximizes utility: $u_D(\mathbf{x}, (\mathbf{q}^{\theta}(\mathbf{x}))_{\theta \in \Theta}) \geq u_D(\mathbf{x}', (\mathbf{q}^{\theta}(\mathbf{x}'))_{\theta \in \Theta})$ for any feasible $\mathbf{x}'$
2. The followers (attackers) play a best response: $u^{\theta}_A(\mathbf{x}, \mathbf{q}^{\theta}(\mathbf{x})) \geq u^{\theta}_A(\mathbf{x}, \mathbf{g})$ for any feasible $\mathbf{g}$.
3. The follower breaks ties in favor of the leader: $u_D(\mathbf{x}, (\mathbf{q}^{\theta}(\mathbf{x}))_{\theta \in \Theta}) \geq u_D(\mathbf{x}, (\bar{\mathbf{q}}^{\theta})_{\theta \in \Theta})$ for any $(\bar{\mathbf{q}}^{\theta})_{\theta \in \Theta}$ that is optimal for the followers, that is for any θ, $\bar{\mathbf{q}}^{\theta} \in \text{argmax}_{\mathbf{g}} u^{\theta}_A(\mathbf{x}, \mathbf{g})$.

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

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

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 solution procedures, 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

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

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.

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

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.

$$
\text{(ERASER)} \qquad
\begin{aligned}
&\max \sum_{\theta \in \Theta} p^{\theta} d^{\theta} && && (1)\\
&d^{\theta} - \sum_{i \in X} x_i R_{ij}^{\theta} \le (1 - q_j^{\theta}) M_1 && \forall j, \theta && (2)\\
&a^{\theta} - \sum_{i \in X} x_i C_{ij}^{\theta} \le (1 - q_j^{\theta}) M_2 && \forall j, \theta && (3)\\
&\sum_{i \in X} x_i C_{ij}^{\theta} \le a^{\theta} && \forall j, \theta && (4)\\
&\sum_{i \in X} x_i = 1 && && (5)\\
&\sum_{j \in Q} q_j^{\theta} = 1 && \forall \theta && (6)\\
&q_j^{\theta} \in \{0, 1\} && \forall j, \theta && (7)\\
&x_i \ge 0 && \forall i && (8)
\end{aligned}
$$

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,

$$M_1 = \max_{j,\theta} Rd_j^{\theta} - \min_{j,\theta} Pd_j^{\theta} \tag{9}$$

$$M_2 = \max_{j,\theta} Ra_j^{\theta} - \min_{j,\theta} Pa_j^{\theta} \tag{10}$$

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 an 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,

$$\begin{aligned} \bar{c}_i \;=\; \bar{c}_{\mathbf{v}} \;&=\; \sum_{j\in Q}\sum_{\theta\in\Theta} R^{\theta}_{ij}\beta^{\theta}_{j} + C^{\theta}_{ij}(\alpha^{\theta}_{j} - \sigma^{\theta}_{j}) - \delta && (11)\\ &=\; \sum_{j\in Q}\sum_{\theta\in\Theta} R^{\theta}_{j}(v_j)\beta^{\theta}_{j} + C^{\theta}_{j}(v_j)(\alpha^{\theta}_{j} - \sigma^{\theta}_{j}) - \delta && (12)\end{aligned}$$

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 the 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).

$$\begin{aligned} \max\quad & \sum_{i\in X}\sum_{\theta\in\Theta}\sum_{j\in Q} p^{\theta} z^{\theta}_{ij} R^{\theta}_{ij} && && (13)\\ & \sum_{i\in X}\sum_{j\in Q} z^{\theta}_{ij} = 1 && \forall\theta \qquad (\pi^{\theta}) && (14)\\ & \sum_{i\in X} z^{\theta}_{ij} = q^{\theta}_{j} && \forall j,\theta \qquad (\sigma^{\theta}_{j}) && (15)\\ & \sum_{i\in X}(C^{\theta}_{ij} - C^{\theta}_{ik}) z^{\theta}_{ij} \geq 0 && \forall j,k,\theta \qquad (\alpha^{\theta}_{jk}) && (16)\\ & \sum_{j\in Q} z^{\theta}_{ij} = x_i && \forall i,\theta \qquad (\beta^{\theta}_{i}) && (17)\\ & z^{\theta}_{ij} \in [0,1] && \forall i,j,\theta && (18)\\ & q^{\theta}_{j} \in \{0,1\} && \forall j,\theta && (19)\\ & x_i \in [0,1] && \forall i && (20)\end{aligned}$$

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.

The MIPSG formulation is similar to the DOBSS formulation of Stackelberg games. The only difference between these formulations is in how they represent the optimal response of the followers. In DOBBS this is done by two sets of $|Q||\Theta|$ constraints, with a big M constant, that define the value v^θ as the optimal reward value for follower θ. In MIPSG the characterization of the optimal follower response is done with the $|Q|^2|\Theta|$ constraints in (16). This means that MIPSG is a formulation with more constraints than DOBBS, but that does not need a big M constant.

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 $\sum_{i\in X}(C^\theta_{ij} - C^\theta_{ik})z^\theta_{ij} \geq 0\ \forall j, k, \theta$ in DOBBS the solution $(\mathbf{z}, \mathbf{x}, \mathbf{q}, \mathbf{v})$ satisfies $0 \leq v^\theta - \sum_{i\in X} C^\theta_{ij} \sum_{k\in Q} z^\theta_{ik} \leq (1 - q^\theta_j)M\ \forall j, \theta$. If $q^\theta_h = 1$ then the DOBBS solution satisfies $\sum_{k\in Q} z^\theta_{ik} = z^\theta_{ih}$ and therefore

$$\sum_{i\in X} C^\theta_{ij} z^\theta_{ih} = \sum_{i\in X} C^\theta_{ij} \sum_{k\in Q} z^\theta_{ik} \leq v^\theta \leq \sum_{i\in X} C^\theta_{ih} \sum_{k\in Q} z^\theta_{ik} = \sum_{i\in X} C^\theta_{ih} z^\theta_{ih}\ ,$$

which is equivalent to the MIPSG constraint.

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

$$v^\theta = \sum_{i\in X} C^\theta_{ih} z^\theta_{ih} \geq \sum_{i\in X} C^\theta_{ij} \sum_{k\in Q} z^\theta_{ik}\ .$$

This satisfies the DOBBS constraints as the only tight right hand inequality is the one that defines v^θ. $\square$

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 that 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 $0 \leq z^\theta_{ij}$, $q^\theta_j \in \mathbb{R}$ and $x_i \in \mathbb{R}$. Note that since $\sum_{i\in X}\sum_{j\in Q} z^\theta_{ij} = 1$ we still have that $z^\theta_{ij}, q^\theta_j, x_i \in [0, 1]$. 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:

$$\min \quad \sum_{i \in \Theta} \pi^\theta \tag{21}$$

$$p^\theta R_{ij}^\theta \leq \pi^\theta + \sigma_j^\theta + \beta_i^\theta + \sum_{k \in Q} (C_{ij}^\theta - C_{ik}^\theta)\alpha_{jk}^\theta \qquad \forall i, j, \theta \tag{22}$$

$$\sigma_j^\theta = 0 \qquad \forall j, \theta \tag{23}$$

$$\sum_{\theta \in \Theta} \beta_i^\theta = 0 \qquad \forall i \tag{24}$$

$$\alpha_{jk}^\theta \leq 0 \qquad \forall j, k, \theta \tag{25}$$

In the LP relaxation of MIPSG the constraint $\sum_{i \in X} z_{ij}^\theta = q_j^\theta$ becomes redundant as it defines the value of q_j^θ, but this variable no longer has to be an integer variable. This fact is reflected in that the corresponding dual variable σ_j^θ 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_{ij}^θ 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_{ij}^θ 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 (22) or (24), 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 since computing the reduced cost of variables z_{ij}^θ or x_i that have not been considered in the master (that is with $i \notin \bar{X}$), we need the dual variable β_i^θ. This is the dual variable corresponding to constraint (17) that is not present in the master problem if strategy $i \notin \bar{X}$ and therefore β_i^θ is not defined.

We address this difficulty by introducing a related optimization problem, which is based on the dual problem of MIPSG. Recall that given a vector $\mathbf{v} \in \{0,1\}^{|Q|}$ that represents a joint patrolling strategy, we denote $R(v_j)^\theta = Pd_j^\theta + v_j(Rd_j^\theta - Pd_j^\theta)$ and $C(v_j)^\theta = Ra_j^\theta - v_j(Ra_j^\theta - Pa_j^\theta)$ the utility of the defender and the θ-th attacker if target j is attacked. Assume a set S of individual patrols is given and for $r \in S$ let $t_{jr} \in \{0,1\}$ indicate whether patrol r covers target j or not. Consider the following optimization problem

(SUBP):

$$\max_{f,v,e,u} \quad \sum_{\theta\in\Theta} f^\theta \tag{26}$$

$$f^\theta \le p^\theta R(v_j)^\theta - \pi^\theta + M^\theta(1-e_j^\theta) - \sum_{k\in Q}(C(v_j)^\theta - C(v_k)^\theta)\alpha_{jk}^\theta \qquad \forall j,\theta \tag{27}$$

$$\sum_{r\in S} u_r \le N \tag{28}$$

$$\sum_{j\in Q} e_j^\theta = 1 \qquad \forall\theta \tag{29}$$

$$\sum_{r\in S} t_{jr}u_r = v_j \qquad \forall j \tag{30}$$

$$v_j \in \{0,1\},\ e_j^\theta \in \{0,1\},\ u_r \in \{0,1\} \qquad \forall j,\theta,r \tag{31}$$

The solution vector of this problem $\mathbf{v}$ corresponds to a joint patrol formed by selecting individual patrols from the set S. Let u_r be a binary variable that is 1 if the schedule $r \in S$ is used in the strategy $\mathbf{v}$ and 0 otherwise. These u_r must sum up to N, which is the number of available resources. Finally, the variable e_j^θ is a binary variable that enables $f^\theta = \max_{j\in Q}\left\{p^\theta R(v_j)^\theta - \pi^\theta - \sum_{k\in Q}(C(v_j)^\theta - C(v_k)^\theta)\alpha_{jk}^\theta\right\}$. This requires the use of a large constant M^θ. The constant M^θ should be large enough so that when $e_j^\theta = 0$ the constraint becomes redundant, at the same time it is desirable that it be the smallest constant that achieves this. The following result gives the best value for M^θ.

Proposition 3.1. *For all $\theta \in \Theta$, the smallest value of M^θ that guarantees constraints (27) are redundant with $e_j^\theta = 0$ is*

$$M^\theta = p^\theta(\max_j Rd_j^\theta - \min_j Pd_j^\theta) - 2|Q| \min_{jk} \alpha_{jk}^\theta \,(\max_j Ra_j^\theta - \min_j Pa_j^\theta) \tag{32}$$

Proof Recall that f^θ will equal $\min_{j\in Q}\left\{p^\theta R(v_j)^\theta - \pi^\theta - \sum_{k\in Q}(C(v_j)^\theta - C(v_k)^\theta)\alpha_{jk}^\theta\right\}$. Then considering $e_j^\theta = 0$ in constraint (27), we have that an M^θ that makes the constraint redundant has to satisfy

$$M^\theta + p^\theta R(v_j)^\theta - \pi^\theta - \sum_{k\in Q}(C_{ij}^\theta - C_{ik}^\theta)\alpha_{jk}^\theta \ge \max_{j\in Q}\left\{p^\theta R(v_j)^\theta - \pi^\theta - \sum_{k\in Q}(C_{ij}^\theta - C_{ik}^\theta)\alpha_{jk}^\theta\right\} \qquad \forall j \in Q,\ \theta\in\Theta\ .$$

The M^θ that satisfies this for all $j \in Q$ satisfies

$$M^\theta + \min_{j\in Q}\left\{p^\theta R(v_j)^\theta - \pi^\theta - \sum_{k\in Q}(C_{ij}^\theta - C_{ik}^\theta)\alpha_{jk}^\theta\right\} \ge \max_{j\in Q}\left\{p^\theta R(v_j)^\theta - \pi^\theta - \sum_{k\in Q}(C_{ij}^\theta - C_{ik}^\theta)\alpha_{jk}^\theta\right\}\ .$$

To ensure we satisfy the above inequality, let us rearrange and bound:

$$
\begin{aligned}
&\max_j \left\{ p^\theta R(v_j)^\theta - \sum_{k\in Q}(C^\theta_{ij} - C^\theta_{ik})\alpha^\theta_{jk} \right\} - \min_j \left\{ p^\theta R(v_j)^\theta - \sum_{k\in Q}(C^\theta_{ij} - C^\theta_{ik})\alpha^\theta_{jk} \right\} \\
\leq & p^\theta(\max_j Rd^\theta_j - \min_j Pd^\theta_j) - 2\max_j \sum_{k\in Q}(C^\theta_{ij} - C^\theta_{ik})\alpha^\theta_{jk} \\
\leq & p^\theta(\max_j Rd^\theta_j - \min_j Pd^\theta_j) - 2|Q|\max_j Ra^\theta_j \ \min_{jk} \alpha^\theta_{jk} + 2|Q|\max_j \max_k \left\{ C^\theta_{ik}\alpha^\theta_{jk} \right\} \\
\leq & p^\theta(\max_j Rd^\theta_j - \min_j Pd^\theta_j) - 2|Q| \ \min_{jk} \alpha^\theta_{jk} \ (\max_j Ra^\theta_j - \min_j Pa^\theta_j) \\
= & M^\theta
\end{aligned}
$$

□

We now show that the optimal solution to SUBP either becomes the joint patrol that should be added to the master problem or it proves that the column generation found the optimal solution. The optimal solution to SUBP identifies a joint patrolling strategy $\mathbf{v}$. If the objective function $\bar{f} = \sum_{\theta\in\Theta} f^\theta$ of this solution is positive then that strategy violates a constraint in the dual and must be incorporated into the master problem. To prove this define $F^\theta_{ij} = p^\theta R^\theta_{ij} - \pi^\theta - \sum_{k\in Q}(C^\theta_{ij} - C^\theta_{ik})\alpha^\theta_{jk}$.

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

$$
\begin{aligned}
& p^\theta R^\theta_{ij} \leq \pi^\theta + \sigma^\theta_j + \beta^\theta_i + \sum_{k\in Q}(C^\theta_{ij} - C^\theta_{ik})\alpha^\theta_{jk} \\
& p^\theta R^\theta_{ij} - \pi^\theta - \sigma^\theta_j - \sum_{k\in Q}(C^\theta_{ij} - C^\theta_{ik})\alpha^\theta_{jk} \leq \beta^\theta_i \\
& \sum_\theta \left(p^\theta R^\theta_{ij} - \pi^\theta - \sigma^\theta_j - \sum_{k\in Q}(C^\theta_{ij} - C^\theta_{ik})\alpha^\theta_{jk} \right) \leq \sum_\theta \beta^\theta_i \\
& \bar{f} = \sum_\theta F^\theta_j \leq \sum_\theta \beta^\theta_i = 0 \\
& \bar{f} \leq 0
\end{aligned}
$$

In this set of equations we are using that $\sigma^\theta_j = 0$, from the dual equation (23), when $q^\theta_j \in \mathbb{R}$, 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 $\bar{f} = \max \sum_{\theta\in\Theta} f^\theta = \sum_{\theta\in\Theta} \max F^\theta_{ij} \leq 0$ for a new strategy i in the subproblem, then there is no new column that must 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 arbitrary β_i^θ that satisfies 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:

$$F_{ij}^\theta \leq \beta_i^\theta \qquad \forall j, \theta \tag{33}$$

$$\sum_{\theta\in\Theta} \beta_i^\theta = 0 \tag{34}$$

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:

$$\bar{f} = \sum_{\theta\in\Theta} \max F_{ij}^\theta \leq 0$$

$$\max F_{ij}^{\bar{\theta}} + \sum_{\theta\in\Theta\setminus\{\bar{\theta}\}} \max F_{ij}^\theta \leq 0$$

$$\max F_{ij}^{\bar{\theta}} \leq \beta_i^{\bar{\theta}}$$

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 (33) 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. $\square$

3.1. Upper and Lower Bounds

Good upper and lower bounds can help speed up the column generation and branching process. Moreover, if we set optimality tolerances, then having tight gaps lead to faster running times. We therefore are 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,

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

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. This is further proof that when $\bar{f} \leq 0$ we have found the optimal solution using 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 $q_j^\theta \in \mathbb{R}$. In the next subsection we discuss how to generate columns (how to conduct pricing) when branching starts.

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 standard branch and price scheme. 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 presented in (21) - (25), there is a dual variable σ_j^θ related to the constraint that defines the 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, then the dual variable is no longer zero and it should be included in the subproblem. Hence, as we branch in the branch and price tree, some of these σ_j^θ become active, changing the subproblem. The termination condition for the column generation is $\bar{f} \leq 0$. This condition remains a valid termination condition for the column generation in branched nodes, but the non-zero σ_j^θ variables modify the value of $\bar{f}$.

The strategy we follow to implement the branch and price is going through the tree following a depth-first search procedure, instead of a breadth-first search 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

Column generation are iterative methods that can run into convergence problems when used to solve linear optimization problem. Vanderbeck [18] listed some of the typical issues that arise when implementing column generation methods due to the dual variables. Among the problems they have detected are: (i) slow convergence, a phenomenon called the *tailing-off effect*; (ii) irrelevant columns generated in first iterations; (iii) 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 behavior. Lubbbecke and Desrosiers [12] described the three important methods: Weighted Danzing-Wolfe decomposition, Trust region method and Stabilization approach using primal and dual strategies. We will use the third method because it has shown a good performance solving classical problems and it is easy to implement [15].

A detailed description and analysis of a Stabilization approach using a smoothing strategy is given in [15]. That work also shows this algorithm improves the runtime for solving classic large problems, such as Machine Scheduling, Bin Packing and Capacitated Vehicle Routing, reducing solution time by a factor of up to 5. They also develop a smoothing technique used for a hybridization of column generation with sub-gradient method. The smoothing technique in its simplest version is as follows. Let $\mathbf{y^t}$ be the dual solution at iteration $t \geq 2$ and $0 \leq \alpha \leq 1$ be a weighting parameter, then the dual $\tilde{\mathbf{y}}^\mathbf{t}$ for the pricing problem for the next iteration is

$$\tilde{\mathbf{y}}^\mathbf{t} = \alpha \hat{\mathbf{y}}^\mathbf{t} + (1-\alpha)\mathbf{y^t} \ . \tag{35}$$

Here $\hat{y}$ is the dual associated to the best (min/max) Lagrangian dual solution so far. At each iteration the dual values are adjusted using as reference the best dual solution values. This gives some stability to the dual variables considered, preventing these dual variables from changing radically from one iteration to the next.

In this simple smoothing scheme we can face three situations: (i) updated duals give us a positive reduced cost column; (ii) we get a new dual bound and improve the optimality gap; or (iii) a mis-pricing occurs and the next iteration smoothed prices get closer to $\mathbf{y^t}$. A mis-pricing is when the subproblem finds a solution with non-positive reduced cost with $\hat{\mathbf{y}}^\mathbf{t}$, but that has positive reduced costs if we use $\mathbf{y^t}$. Under these conditions, the column generation method with smoothing pricing approach converges to an optimal solution in a finite number of iterations [15].

A fixed α gives a smoothing scheme that convergences after some iterations. A better approach considers an auto-adaptive α, which increases and decreases as upper-lower bound gap changes. In [15] they propose an algorithm for this adaptive method, which is based on a sub-gradient information and a mis-pricing sequence for a given initial α.

Algorithm 1: Sub-gradient routine

```
1  $\alpha^0 = \alpha$; $t = 0$;
2  while reduced cost $> 0$ do
3      Solve the master problem;
4      Call subproblem with
         $\hat{\mathbf{y}}^{\mathbf{t}} = \bar{\alpha}\hat{\mathbf{y}}^{\mathbf{t}} + (1-\bar{\alpha})\mathbf{y}^{\mathbf{t}}$;
5      if mis-pricing occurs then
6          Start the mis-pricing schedule
             (Algorithm 2);
7      else
8          Let $\mathbf{g}^{\mathbf{t}}$ be the sub-gradient;
9          if $\mathbf{g}^{\mathbf{t}}(\hat{\mathbf{y}}^{\mathbf{t}} - \mathbf{y}^{\mathbf{t}}) > 0$ then
10             $\alpha_t = f_{inc}(\alpha)$;
11         else
12             $\alpha_t = f_{dec}(\alpha)$;
13     $t = t + 1$;
```

Algorithm 2: Mis-pricing sequence

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

In Algorithm 1, we use functions for increasing and decreasing α. These functions are as follows,

$$f_{inc}(\alpha_t) = \alpha_t + (1-\alpha_t)\cdot 0.1 \tag{36}$$

$$f_{dec}(\alpha_t) = \begin{cases} \frac{\alpha_t}{1.1} & \text{if } \alpha_t \in [0.5, 1) \\ \max\{0, \alpha_t - (1-\alpha_t)\cdot 0.1\} & \text{otherwise} \end{cases} \tag{37}$$

The vector $\mathbf{g}^{\mathbf{t}}$ is the sub-gradient for a given dual $\mathbf{y}^{\mathbf{t}} = [\pi, \alpha]$ solution. This vector is computed as follows,

$$\mathbf{g}^{\mathbf{t}}\mathbf{y}^{\mathbf{t}} = \sum_{\theta\in\Theta} M^\theta(1-e_j^\theta) - \pi^\theta - \sum_{k\in Q} C(v_j))^\theta - C(v_k)^\theta)\alpha_{jk}^\theta \tag{38}$$

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

3.4. Greedy Algorithm for Subproblem

On one hand, the master problem solution for the leader corresponds to the best mixed strategy using available schedules. On the other hand, the subproblem finds the best schedule to include for a new column using the dual values from master problem. Since the number of resources is limited, it seems natural to solve this subproblem with a greedy heuristic. Those targets with the highest value (from duals) for the objective function should probably be picked for the new column. Algorithm 4 describes this greedy heuristic in detail.

We use this algorithm as an additional speed up subroutine for the column generation. First, we solve the Greedy Algorithm, if the column we find has a positive reduced cost, then we add it to Master problem. If not, then we try with SUBP described in Section 3, this optimization problem must be used for checking optimality at the last step.

In the Greedy algorithm, we try all the schedules over set of feasible schedules S and we keep in the new strategy only those with the highest reduced cost. We repeat this process until no more resources can be assigned. Finally, we return the best strategy and its reduced cost. The algorithm we implement is Algorithm 3.

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 $i = 1; i \leq N; i = i + 1$ do
4      $index = 0$; $best = -\infty$;
5      for $r = 1; r \leq |S|; r = r + 1$ do
6          Set schedule $s_r$ temporally to vector v;
7          if $best < value(\mathbf{v})$ then
8              $best = value(\mathbf{v})$;
9              $index = r$;
10     Include schedule index into v;
11     reduced cost = $value(\mathbf{v})$;
12 return reduced cost and vector v;
```

4. Computational Results

We randomly generate a set of instances to be solved for each solution method. The base algorithms considered are the branch and price methods for the MIPSG and the ERASER formulations of the problem, we refer to these solution algorithms as MIPSG-C and ERASER-C, respectively. The ERASER-C algorithm is the state of the art benchmark from prior work [6]. In addition we solve each instance using the greedy subroutine and the stabilization approach presented above to attempt to speed up the column generation step when solving the MIPSG formulation. We refer to these as GREEDY and STAB, respectively. In summary we present computational results to compare four solution methods: ERASER-C, MIPSG-C, GREEDY and STAB over randomly generated instances.

We generate random instances by sampling rewards and penalties for the leader and the attacker and also by generating a random set of initial patrols or schedules for each instance. We generate reward values using a log-normal distribution because this guarantees that the reward is positive. Furthermore the log-normal distribution depends on two parameters μ and σ that can be used to adjust the distribution. If $X \sim$ Log-normal(μ, σ) then $X = e^{\mu + \sigma Z}$ with Z a standard normal distribution. We set $\mu = 3.107304$ and $\sigma = 1.268636$ so that the coeffecent of variation $CV_X = \sqrt{e^{\sigma^2} - 1} = 2$ and the mean is $\mathbb{E}(X) = e^{\mu + \sigma^2/2} = 50$. Penalties are generated in a similar way but with a negative log-normal. This guarantees that the penalty is always less than the reward for every attacker and defender. The set of available schedules is sampled from a discrete random variable in a way we do not have repeated schedules. We have seen that a coefficient of variation of 2 corresponds to a large input variability.

Our instances consider 1 adversary, and as a base case 70 different zones or targets, 5 police resources to be allocated, 600 individual schedules that each police resource can choose from with each of these schedules covering 5 targets. We conduct sensitivity analysis on these problem parameters by varying them as indicated in Table 1, base case is indicated by bold:

Number of targets	50, 60, **70**, 80, 90, 100
Defender resources	1, 2, 3, 4, **5**, 6, 7, 8, 9, 10
Number of schedules	200, 400, **600**, 800, 1000
Targets covered per schedule	2, 3, 4, **5**

Table 1: Problem parameters considered in computational results. Base case is in bold.

For each combination of the first three problem parameters we generate 25 random problem instances also selecting randomly the value of targets per schedule. We vary one of the first three parameters at a time and remove repeated instances. This gives a total of 500 problem instances that are solved by the four solution algorithms. To solve each instance we use *CPLEX 12.4* with a runtime limit set to 2 hours.

4.1. Algorithm Comparison

The first thing to note is that no algorithm is able to solve all of the 500 random problem instances in the 2 hour time limit. In Table 2 we present the total percentage of problems solved and the number of nodes in the branch and price tree used for each of the four solution algorithms considered MIPSG-C , ERASER-C, GREEDY and STAB. Methods that use MIPSG as base model only need one node in B&P tree because the linear relaxation of this problem with one adversary gives the integer solution. ERASER needs to branch more because the big M formulation gives a larger integrality gap. Overall the ERASER-C algorithm is able to solve the most instances, which suggest that it is more efficient. In what follows we present detailed results to understand for which problem parameters one algorithm is preferable over the other.

Algorithm	Problems Solved (%)	Nodes
MIPS G-C	87.6	1
ERASER-C	94.2	138
GREEDY	85.4	1
STAB	86.4	1

Table 2: Computational results summary.

The number of columns generated in the B&P method indicates the amount of work that is needed to solve a problem. In Table 3 we show the average number of columns generated by each algorithm for different number of resources. The remaining problem parameters are fixed at the base case: 70 targets, 600 feasible schedules and 5 targets covered by schedule. We present results for 2, 4, 6, 8 and 10 resources.

Resources	MIPSG-C	ERASER-C	GREEDY	STAB
2	484	115	578	487
4	482	401	367	406
6	734	886	639	680
8	158	388	209	120
10	115	140	118	54

Table 3: Average number of generated columns. Targets 70, schedules 600, targets/schedule 5.

The average number of columns that are generated are comparable for all algorithms and varies with the number of resources. The results suggest that instances with a large number of resources require less columns and are thus easier to solve. ERASER-C does very well with few resources as well. Finally, STAB reduces the number of columns generated by MIPSG-C for all cases with resources greater than 4. GREEDY does not seem to reduce the number of columns generated.

In Figure 1, 2 and 3 we have plotted the average solution time for each algorithm. When a method is not able to solve an instance within the given time limit, we consider the maximum time of 2 hrs. In each plot we present sensibility with respect to one parameter, keeping the rest in the base case. In Figure 1 we see the solution times as a function of the number of resources; in Figure 2 as a function of the number of schedules; finally in Figure 3 as a function of the number of targets. The running time increases more for ERASER-C than for the algorithms that tackle the MIPSG formulation. For large number of resources (8 and 10) the runtime for MIPSG-C and GREEDY is smaller than when there is less resources. The increase in number of resources does create a significant difference between MIPSG-C and the speed-up methods, showing comparable runtimes regardless of the number of resources. The runtimes of MIPSG-C is close to 5 times smaller than the runtimes for ERASER-C when there are many resources. Figure 2 shows that the algorithms that tackle the MIPSG formulation increase a ittle as the number of schedules that form the possible joint patrols increases. ERASER-C is more sensitive showing a significant increase as the number of schedules goes from 600 to 1000. For the instances with schedules from 600 to 1000 ERASER-C has more than 50% runtime than MIPSG-C. The speedup alternatives (GREEDY and STAB) give comparable

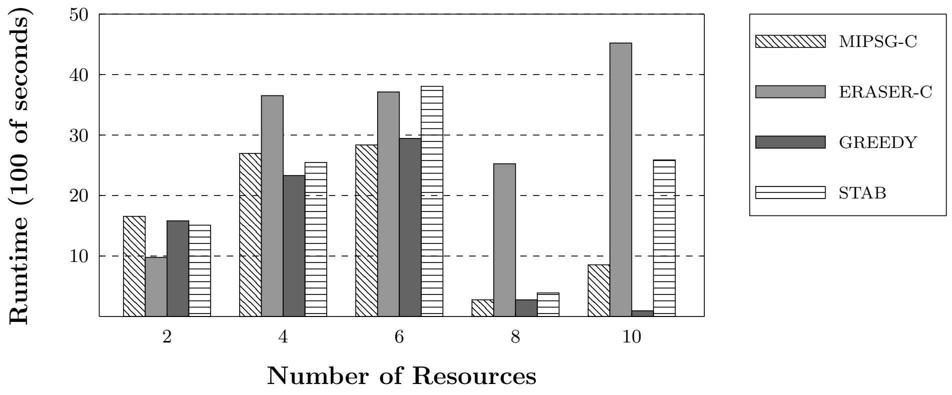

Figure 1: Running time versus number of resources.

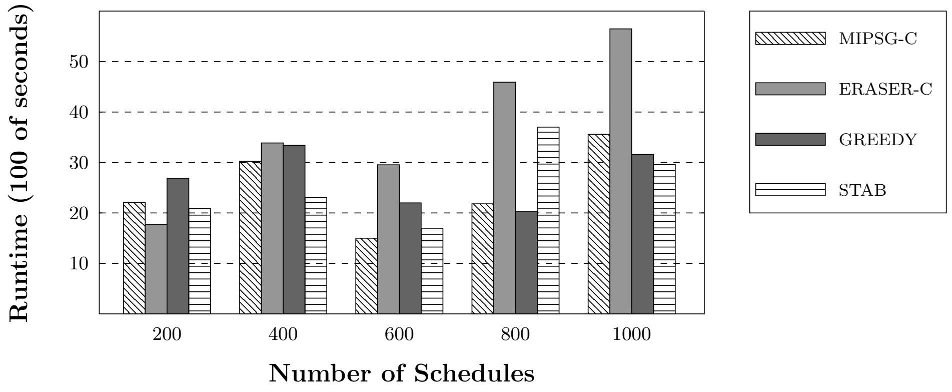

Figure 2: Running time versus number of schedules.

runtimes to MIPSG-C. Finally, in Figure 3 we see that all four solution algorithms increase their runtime as the number of targets increase. In particular the runtimes increase significantly when the targets go from 70 to 80. Again MIPS-G shows a slightly better runtime than ERASER-C and a comparable performance when compared to GREEDY and STAB.

4.2. MIPSG Column Generation Results

Figure 4 shows how the number of targets affects the runtime for MIPSG-C. We have made this analysis showing separately the result for each number of targets per schedule (T/S). The other parameters, schedules and resources, are the same as base case. From this plot, first we see that as we increase the number of targets, the running time increases for all target to schedule values. Note that this relation is not linear, which is consistent with the combinatorial nature of the problem. An instance with too many targets is hard to solve even when we have a limited number of schedules and resources, in the base case set to 600 and 5, respectively. We can also see that the T/S impacts the solution time, a 2 T/S instance is easier to solve than a 5 T/S instance. We can see a direct relation between T/S and runtime. An explanation for this is

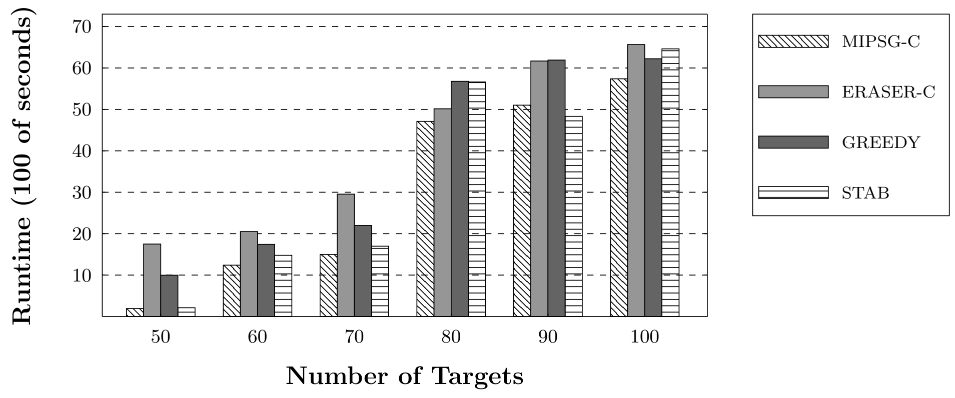

Figure 3: Running time versus number of targets.

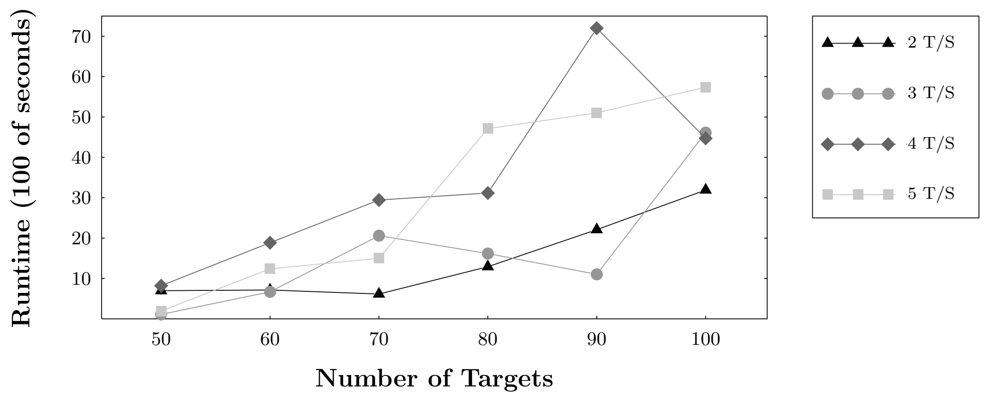

Figure 4: Running time versus number of targets and T/S for MIPSG-C.

that the number of T/S of an instance is related to how flexible it is to cover targets. The case with 2 T/S assigns patrols to resources that more easily can be combined to protect targets of interest.

In Figure 5 we show the average runtime for each resource number and T/S value. The number of resources represents how many schedules can be in the same strategy. It is not clear how the number of resources impacts on time, but the extreme values, 1 and 10, seem to be faster to solve than the values in the middle, for example 5 or 6. Finally, in Figure 6 we have runtime versus number of available schedules and T/S, there is no clear dependency of the runtime on the number of schedules, as in all the plots that separate results for each T/S value se see that there is a tendency to increase solution time as the number of T/S increases.

4.3. MIPSG Stabilization

For MIPSG model it is critical to find useful speedup strategies for the column generation to be able to reduce the overall runtime. When dual values vary substantially from one iteration to the next, we probably generate irrelevant columns as Vanderbeck has established [18]. The dual stabilization method we

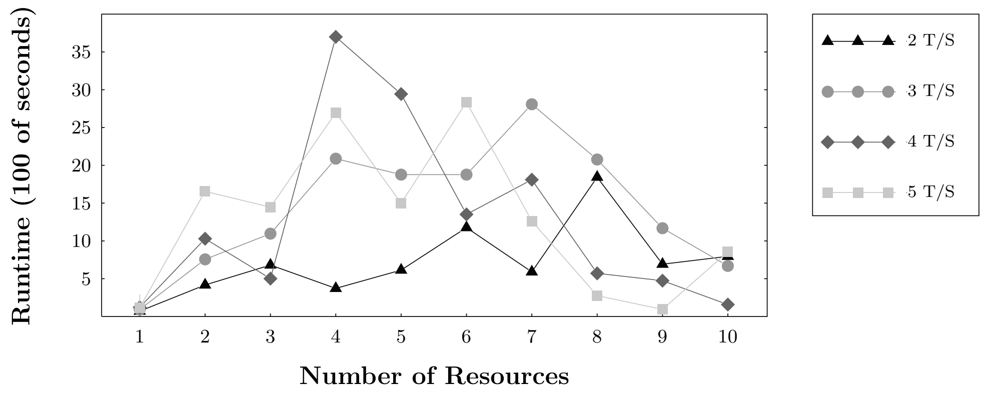

Figure 5: Running time versus number of resources and T/S for MIPSG-C.

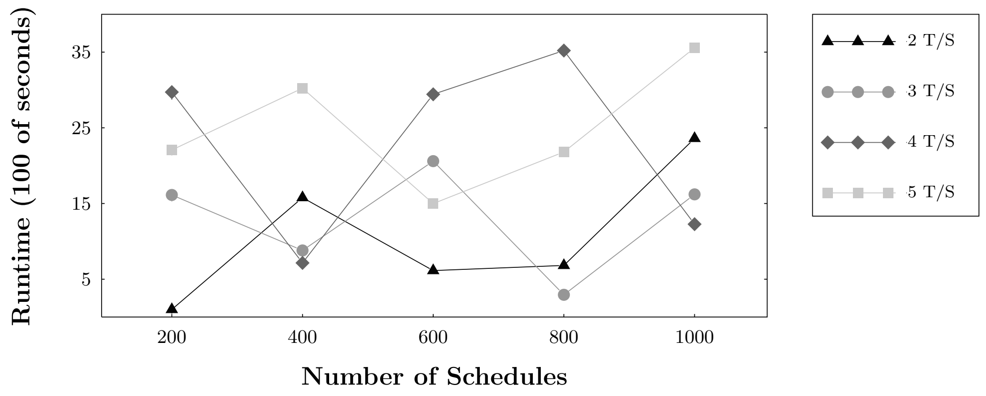

Figure 6: Running time versus number of schedules and T/S for MIPSG-C.

implemented addresses this issue directly. Figure 7 shows the upper and lower bounds for an instance, where we observe that STAB, the stabilization procedure that smoothes the dual variables, has a more monotonic behavior. In this case, we not only have a smoother curve but also the method finishes in fewer iterations, about 45 versus 50. In addition, note that the lower bound is affected, the STAB bound reaches a higher value faster than MIPSG-C bound. The graph however shows that there still is some variability and the rate of convergence, although smaller is comparable to the un-stabilized case. This improved performance has not translated to a significant runtime reduction as can be seen from the agregate results presented earlier. Further work is needed to check whether other stabilization methods could lead to better performance.

5. Conclusions

In this paper we have introduced a column generation method to solve a novel mixed integer formulation of Stackelberg Security Games. Decomposition methods are key to be able to solve ever larger problem instances and strong formulations, such as MIPSG, provide good opportunities to develop efficient algorithms. That said, even if the linear relaxation problems of MIPSG yield better integrality gaps than other existing

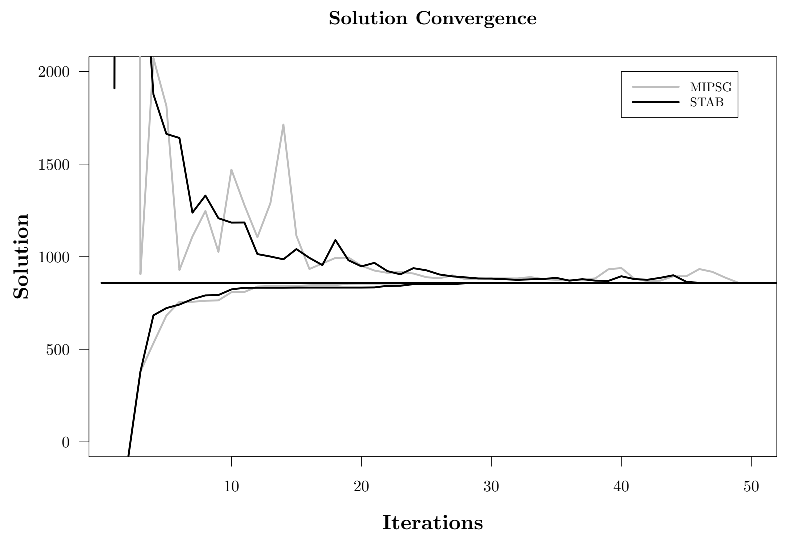

Figure 7:

formulations, the relaxations of MIPSG turn out to be challenging to solve due to the large number of variables in the formulation.

A standard column generation model would not work for this MIPSG problem, as the reduced cost of candidate variables depend on undefined dual variables. We circumvent this challenge by formulating a related problem to replace the standard column generation subproblem, and use this to generate candidate joint strategies and to detect optimality. We are able to obtain upper and lower bounds at every iteration. Furthermore we explore methods to speed up the column generation approach, including greedy heuristics to solve the subproblem SUBP, or dual smoothing techniques.

Problem structure in SSG can be exploited to create a polynomial mixed integer formulation for SSG when defender strategies consist of all patrols that deploy N resources on $|Q|$ targets. The approach is based on a formulation of the frequency with which each target is protected and the fact that there are no patrol feasibility constraints allows to obtain an implementable solution strategy from this optimal frequency, [10]. In the game considered in this work however, joint strategies are constructed from a given set of fixed patrolling alternatives. This makes it impossible to use the polynomial reformulation forcing the use of exact decomposition algorithms.

Our preliminary computational results evaluate the efficiency of the column generation method presented, as well as the greedy heuristic and the stabilization by dual price smoothing. Furthermore we contrast these results with ERASER, a state of the art column generation method, built on a different mixed integer formulation. We note that ERASER branches more than the branch and price method proposed for MIPSG, ERASER also generates more columns as the number of resources increases. However since each linear relaxation for ERASER is easier to solve, it remains a competitive solution alternative. Our computational results show that MIPSG-C exhibits a significantly smaller runtime when either the number of resources or the number of schedules increases.

We note that there remains a challenge on how to solve the linear relaxations for the MIPSG problem faster. Our computational results show slight improvements from using both the Greedy heuristic and the stabilization procedure, but further work is necessary to make these speedup methods beneficial overall. Being able to more efficiently solve the linear relaxations of MIPSG can help construct an efficient large scale algorithm for Stackelberg games as a column generation on MIPSG solves far fewer linear relaxation problem.

6. Acknowledgements

Research was supported by CONICYT through Fondecyt grant 1140807 and the Complex Engineering Systems Institute, ISCI (CONICYT: FB0816).

Bibliography

[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. A study of general and security Stackelberg game formulations. Working paper, Département d'Informatique, Université Libre de Bruxelles, 2017.

[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, D. Bazin, R. Hill, and L. Urch. *Market Structure and Equilibrium.* Springer Berlin Heidelberg, 2010.

[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.