UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FISICAS Y MATEMATICAS
DEPARTAMENTO DE INGENIERIA INDUSTRIAL
DEPARTAMENTO DE INGENIERIA MATEMATICA

ALGORITMO HEURÍSTICO PARA JUEGO DE SEGURIDAD DE STACKELBERG EN UNA RED

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN GESTIÓN DE OPERACIONES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO

TOMAS ENRIQUE SPENCER BRAVO

PROFESOR GUIA:
FERNANDO ORDOÑEZ PIZARRO

MIEMBROS DE LA COMISION:
JORGE AMAYA ARRIAGADA
RAUL MANASEVICH TOLOSA
RICHARD WEBER HAAS

SANTIAGO DE CHILE
ENERO 2013

Summary

The main objective of this work is to provide an algorithm that solves huge instances of Stackelberg's linear programming security problems reducing the amount of resources needed to compute said solution. For this we utilize column generation principles [1] to develop an algorithm that proceeds by solving a smaller problem (small number of restrictions). Then, iteratively, we add restrictions until the problem reaches defined stopping conditions. Basically, we start with a sub problem of the original with two players playing with a limited strategy space which considers only a limited number of restrictions. Iteratively, we verify if some player would like to change his strategy in order to increment their utilities, and we add that strategy and solve once again. This suggests a decomposition method that is able to guess the minimal set of restrictions to consider in order to also find the optimal solution for the problem.

In our studies we identified that our iteration process does not always find the global optimal solution for the problem. Then, we provide an analysis and characterization of the structure of utilities functions for both players that let us further understand the players' dynamics and identify situations where the global optimal solution is indeed found.

Later, we present an implementation that comprises real world data on a real network located downtown at Santiago, Chile. The rewards were calculated considering the historic average stolen on each location and an estimated value of unwillingness on going to prison regarding the assaulters. Finally, we compare our algorithm with other ones already in literature under similar scenarios. We show our methods let us efficiently provide reasonable solutions for security problems in real world size.

Resumen

El objetivo principal de este trabajo es proporcionar un algoritmo que resuelve instancias de gran tamaño para problemas de juegos de seguridad de Stackelberg con un énfasis en reducir el número de recursos necesarios requeridos para calcular dicha solución. Para ello se utilizan los principios de generación de columnas para desarrollar un algoritmo que procede mediante la resolución de un problema más pequeño (menor número de restricciones). Entonces, de forma iterativa, añadimos restricciones hasta que el problema llega a las condiciones de parada definidas. Básicamente, partimos de un problema secundario del original con dos jugadores que juegan cuentan con un espacio de estrategia limitado, este considera sólo un número limitado de restricciones. Iterativamente, verificamos si algún jugador le gustaría cambiar su estrategia con el fin de incrementar sus utilidades, añadimos la estrategia candidata y resolvemos una vez más. Esto sugiere un método de descomposición que es capaz de estimar el conjunto mínimo de restricciones a tener en cuenta con el fin de encontrar también la solución óptima para el problema global.

En el transcurso de nuestros estudios identificamos que el proceso de iteración no siempre encuentra la solución óptima para el problema global. Luego, proporcionamos un análisis y caracterización de la estructura de las funciones de utilidad para ambos jugadores con el fin comprender más la dinámica de los jugadores e identificar las situaciones en las que la solución óptima global efectivamente es encontrada.

Más tarde, se presenta una implementación que incluye datos del mundo real a través de una red en el centro de Santiago, Chile. Las recompensas se calcularon teniendo en cuenta el promedio histórico robado en cada lugar y un valor estimado de la falta de voluntad de ir a la cárcel para efectos de los asaltantes. Finalmente, comparamos nuestro algoritmo con los demás ya la literatura en escenarios similares. Mostramos que nuestros métodos nos permiten ofrecer de manera eficiente soluciones razonables para los problemas de seguridad en tamaño del mundo real. Además comparamos nuestros resultados con los resultados utilizando la metodología estándar de resolución de problemas lineales y mostramos que se pueden reducir ampliamente la necesidad de recursos computacionales y en algunos casos, el tiempo de ejecución para llegar a la solución.

“Since the fabric of the universe is most perfect and the work of a most wise Creator, nothing at all takes place in the universe in which some rule of maximum or minimum does not appear...”

Leonhard Euler, As quoted in The Anthropic Cosmological Principle (1986) by John D. Barrow and Frank J. Tipler, p. 150

1 Index

1.1 Graph Index

1.2 Table Index

2 Introduction

Last couples of years, several contributions have been made in models and algorithms relative to security games, especially on Stackelberg games. This class of games describes a situation where two players, an attacker and a defender, play to cover/attack different targets obtaining, in exchange, a reward. First, the defender (leader or master) allocates his resources among the targets he want to protect, and then the attacker (follower or slave) chooses the most profitable objective to assault. The main objective of these models is to find solutions that represent their best rational decision. Solutions for this family of problems are called Strong Stackelberg's Equilibriums (SSE) which has the particularity of complying with the condition of mutual best-responses and breaks ties in the benefit of the leader, in our case, the defender.

We consider this Stackelberg's Security Game on a Network where the adversary could attack some sequence of nodes, security forces (defender) decide to patrol a subset of nodes from the network. A strategy profile for this problem is a pair $< C, r >$ where C correspond to a probability distribution for the defense resources among the nodes and r is the chosen route to attack by the terrorists (attacker). This game could represent community crime, where pickpocketers travel across a commercial district and the decisions of police about which areas of the city they patrol.

On real world size implementation, instances for this problem have such a large number of routes that it promptly becomes computationally inoperable. If we consider a scenario with 100 points of interest, or nodes, we have up to 2^{100} possible different routes. Each route ends up being an additional restriction in the lineal problem (LP) to resolve.

Our work proposes a heuristic algorithm based in a decomposition method of column generation and restriction generation adjusted for Integer Programming (IP). Using a two stages algorithm, we begin with only one route and solve the problem for the defender and attacker iteratively. Each time the defender allocates his resources and the attacker, seeing the defender's coverage, chooses the best suited route for attack. For every not previously considered route, we add it to the problem until no new route is proposed. We also compare our results with the results using the standard methodology of solving linear problems and show that it can greatly reduce the need for computational resources and in some cases, the running time to reach the solution.

2.1 Stackelberg's Equilibrium

In game theory the usual concept of solution is a *Nash's equilibrium*. A Nash's equilibrium recommends a strategy profile (strategy for each player) such that no player can improve upon changing his strategy unilaterally. Since the other players are also rational, it is reasonable for each player to expect his opponents to follow the recommended strategy as well. [2]

Stackelberg's equilibrium is a refinement of Nash's equilibrium specific to Stackelberg games in the sense that no player takes benefit to unilaterally change his strategy. These kinds of games are thoroughly described on next chapter. This form of equilibria corresponds to a subgame perfect equilibrium [3]; it assumes each player chooses a best-response in any subgame of the original (where a subgame corresponds to partial sequences of actions). This point of view eliminates all Nash equilibrium profiles supported by non-credible threats off the equilibrium path.

Literature identifies two types of unique Stackelberg's equilibrium, these typically known as 'strong' [4] and 'weak' [5]. The strong form assumes that in case equilibrium is found and more than one target is indifferent for the attacker, then the tiebreakers are chosen in favor of the defender. That is, from within the indifferent best targets for the attacker, the one that gives the most benefit for the defender is the attacked one. Whereas the weak form assumes the follower will choose the worst strategy for the leader, from within the indifferent available targets. A strong Stackelberg's equilibrium exists in all Stackelberg games, but a weak Stackelberg's equilibrium may not [6]. In addition, a justification for the Strong Stackelberg's equilibrium is that the leader can often induce the favorable strong equilibrium by selecting a strategy arbitrarily close to the equilibrium that causes the follower to strictly prefer the desired strategy [7]. We adopt strong Stackelberg's equilibrium here due to the key existence result and because it is the most commonly adopted concept in related literature [8], [9] and [10].

Definition 1: A pair of strategies $(\delta_D, F_A(\delta_D)) \in \Delta_D \times \Delta_A$ forms a strong Stackelberg equilibrium (SSE) if they satisfy the following:

1. The leader plays a best-response:

$$U_D\big(\delta_D, F_A(\delta_D)\big) \geq \; U_D\big(\delta'_D, F_A(\delta'_D)\big) \;\; \forall \delta'_D \,\epsilon\, \Delta_D$$

Equation 1

2. The follower plays a best-response:

$$U_A\big(\delta_D, F_A(\delta_D)\big) \geq U_A(\delta_D, \delta_A) \;\; \forall \delta_A \,\epsilon\, \Delta_A, \delta_D \,\epsilon\, \Delta_D$$

Equation 2

3. The follower break ties optimally for the leader:

$$U_A\big(\delta_D, F_A(\delta_D)\big) \geq \; U_A(\delta_D, \delta_A) \;\; \forall \delta_D \; \in \; \Delta_D, \delta_A \; \in \Delta_A$$

Equation 3

With $\Delta_A^*(\delta_D)$ the set of follower best responses given a leader strategy δ_D.

Whether or not the Stackelberg leader benefits from the ability to commit depends on whether commitment to mixed strategies is allowed. Committing to a pure strategy can be either good or bad for the leader; for example, in the "Rock, Paper, and Scissors" game, committing to a pure strategy guarantees a loss. However, the ability to commit to a mixed strategy always weakly increases the leader's payoffs in equilibrium profiles of the game [7] again on the "Rock, Paper, and Scissors" game, if we assume payoffs as 1, 0 and -1 for a win, draw and loss situations respectively, then a mixed strategy of (1/3, 1/3, 1/3) will maximize each players expected payoff. In the context of a Stackelberg game, a deterministic policy is a liability for the leader, but a credible randomized security policy is an advantage, as the leader is granted the Nash's payoff by playing the Nash's strategy but can choose to deviate in case is convenient.

In a Stackelberg security game the defender decides first (leader), and the attacker selects his action afterwards (follower). Our model allows commitment to mixed strategies by the defender. The attacker, instead, can only choose deterministic strategies, as common literature have previously stated. This last assumption is without loss of generality, because as the utilities are all lineal functions, there is always a pure strategy in the set of best responses for the follower on the set $\Delta_A^*(\delta_D)$ [11] [12].

In other words, the defenders goal is to maximize its reward given the attacker will attack with knowledge of the defensive strategy the defender has chosen. In most cases, the optimal strategy for the defender is a randomized strategy in which it chooses a mixed strategy over all his possible resource assignments. Randomized policies are unpredictable since even thought the attacker knows the overall strategy, he is unable to predict the exact resource assignment for any day. The attacker, instead, is restricted to a pure strategy. As we previously stated this is with no loss of generality because there is always a pure strategy in the argmax of the problem solved by the attacker since they are all lineal functions.

2.2 Stackelberg Security Games

A Stackelberg security game (SSG) is a Game between two players: a defender and an attacker. These players need not represent individuals, but could also be groups that cooperate to execute a joint strategy, such as the police force or terrorist organization. The defender wishes to deploy up to m security resources to protect the set of targets T from the attacker. Similarly, the attacker aims to attack the target with the better payoff, subject to the defender's allocation strategy. Each player has a set of possible pure strategies denoted as, $\Sigma_D$and Σ_A, for the defender and the attacker respectively. A mixed strategy allows a player to play a probability distribution over pure strategies, we denote this sets as Δ_D and Δ_A. Payoffs for each player are defined over all possible joint pure strategy outcomes as follows: $\mathrm{U}_D\colon \Sigma_D \times \Sigma_A \rightarrow \mathbb{R}$ for the defender and similarly for the attacker with U_A. The payoff functions are extended to mixed strategies by taking the expectation over pure-strategy outcomes. In other words, we define $\mathrm{U}_D\colon \Delta_D \times \Delta_A \rightarrow \mathbb{R}$ such that $C \in \Delta_D$, then $C = \sum_{i\in\Sigma_D} q_i * C^i$ with $\sum_{i\in\Sigma_D} q_i = 1$, $q_i \geq 0$ and $C^i \in \Sigma_D$. Then, $\mathrm{U}_D(C,r) = \sum_{i\in\Sigma_D} q_i * \mathrm{U}_D(C^i,r)$. Let us note that as we previously mentioned, only the defender is allowed to act with mixed strategies as we assume the attacker attacks with pure strategies only.

Extending our definition of a normal form security game, Stackelberg games introduce a distinction between the players: the common literature considers a situation with two players; the “leader”, as the player who moves first and a “follower” who observes the leader strategy before acting. The concept behind this is to model the capacity of attackers or terrorists to employ surveillance and, this way, infer the defender’s strategy, before selecting the best target to attack. In this model, predictable defense strategies are vulnerable to exploitation by the attacker [13]. In formal game theory context, the attacker’s strategy in a Stackelberg security game is a function F_A that selects an attacking strategy in response to each leader’s defending strategy: $\mathrm{F}_A: \Delta_D \rightarrow \Delta_A$. Further specifications on how we define this strategy function are explained later in this document.

In a Stackelberg security game, the defender first commits to a strategy, and then the attacker decides its optimal attack with complete information about the defender’s strategy. Therefore, the defender’s goal is to maximize its reward given that the attacker will attack a single target with knowledge of the defensive strategy the defender has chosen. In most cases, the optimal strategy for the defender is a mixed strategy over all its possible resource allocations. The benefit of randomized policies is that they are unpredictable even though the attacker may know the overall strategy; the attacker is unable to predict the exact resource assignment for any specific day.

Previous literature in Stackelberg games have provided different approaches for solving optimum randomized allocations in the form of algorithms that solve Stackelberg games of huge size; ARMOR, ERASER, PROTECT, GUARDS, ORIGAMI to name a few [12] [14] [15] [16] and they have been deployed in practice in LAX airport and Federal Air Marshalls service (FAMS).

3 The ERASER algorithm as a starting point

In this chapter we present ERASER, the algorithm we use as inspiration during our studies. Then we explain how to identify a Stackelberg Security Game on a network with police and terrorists on a city. Detail an analysis of the structure of utility functions and characterize some properties that let us further understand the way dynamics between player works. Finally we show that our problem can always be modeled in compact form and how we use that to our advantage.

3.1 A state of the Art SSG model (ERASER)

For the past few years, several studies have developed research in security games, most recent work being ASPEN, IRIS, GUARD, PROTECT, ERASER and ARMOR programs (models or algorithms). Plenty of previous work has already focused on arbitrary schedules or multiple resource allocation strategies in the defender's topologies using Stackelberg [17] [10] [16]. In these models the attacker is the one who chooses a single specific target to attack. In our domain we utilize an approach where the attacker's pure strategies topology consists on a set of routes that can be considered as attacking subsequent nodes as the defender pure strategies; another way of looking at it is such as the attacker chooses a feasible set of nodes, with feasibility depending on the existence of connecting arcs.

The objective of EASSGAR is to find an efficient defender allocation of m resources on the nodes of a graph as strategy C, maximizing the worst case defender utility. More specifically, we assume that the attacker will always choose the most rewarding route. The key concept we introduce in our algorithm is the idea of not considering all the possible routes but only a subset of them. We will later show that this reduction in routes is actually possible but only up to a certain point.

We considered the ERASER algorithm [13] as the starting point of our research, this model focus on a representation of the game problem in the form of a Mixed Integer Lineal Problem (MILP) which is perfectly suited as a benchmark for further research, development and implementation.

The ERASER algorithm (Efficient Randomized Allocation of SEcurity Resources) takes as input a security game in compact form and solves for an optimal coverage vector corresponding to a SSE strategy for the defender. The algorithm is a mixed-integer linear program (MILP), presented in the equations 4 to 11.

Equations 4 to 7 restrict variables to represent probability distributions over possible strategies for the defender and deterministic decisions in the case of the attacker, as previously defined. Equations 8 and 9 guarantee the mutual best responses assumption. Imposing pure strategies for the attacker helps allowing the leader to determine the mutual best responses in order for the solution to be a SSE. That is, the leader can immediately know which route will be attacked given an allocation function and, this way, guarantees that the solution is a mutual best-response.

Equations 10 and 11 only illustrate the specific structure of utilities for each of the players and serves for clarification purposes.

More detailed information regarding the exact interpretations for each of the equations will be provided further in this document.

$$\max d$$

$$a_t \epsilon \; \{0,1\} \; \forall t \; \epsilon \; T$$

Equation 4

$$\sum_{t \epsilon T} a_t = 1$$

Equation 5

$$c_t \epsilon \; [0, 1] \; \forall t \; \epsilon \; T$$

Equation 6

$$\sum_{t \epsilon T} c_t \leq m$$

Equation 7

$$d - U_D(C, t) \leq (1 - a_t)Z \;\; \forall t \; \epsilon \; T$$

Equation 8

$$0 \leq k - U_A(C,t) \leq (1 - a_t)Z \;\; \forall t \, \epsilon \, T$$

Equation 9

With both players utilities defined as follows,

$$U_D(C,t) = \; c_t U_D^c(t) + (1 - c_t) U_D^u(t)$$

Equation 10

$$U_A(C,t) = \; c_t U_A^c(t) + (1 - c_t) U_A^u(t)$$

Equation 11

The optimal solution to the ERASER MILP corresponds to a SSE of the security game. Also, the coverage allocations vector obtained corresponds to a leaders mixed strategy that implements the coverage probabilities [13].

A key aspect in defining security games is identifying the space where each of the players will play strategies. Vectors a_t and c_t are each mapped into nodes and possible strategies for both players, this is, a mixed probability distribution allocation in the case of the leader (c_t) and a pure target strategy for the follower (a_t). Note that in the case just mentioned, the universe of pure strategies is the same for both players, similarly as in the *rock, paper and scissors* game. We introduce an extension of this concept and map different set of pure strategies for each player, similarly the utility function is also extended in order for the problem to make sense. This last modification implies large differences in the capability of obtaining exact optimal results as part of an efficient algorithm. We later introduce the set of pure strategies for the attacker as a set of connected nodes emulating a route being walked by the attacker.

3.2 An SSG model on a Network

Our proposal for the security game is a two player game between a defender and an attacker playing on a directed graph $\boldsymbol{G}$ with nodes and arcs $\boldsymbol{N}$ and $\boldsymbol{A}$ respectively. The attacker's pure strategy space Δ_A is a set of Routes $\boldsymbol{R}$ that could be attacked; as we will later discuss, each element $r \in \boldsymbol{R}$ is a set of subsequent nodes, $\boldsymbol{R} = \{\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{r_3} \ldots \boldsymbol{r_s}\}$. The defender's pure strategy space is a subset of $\boldsymbol{m}$ nodes from $\boldsymbol{N} = \{\boldsymbol{1}, \boldsymbol{2}, \boldsymbol{3}, .., \boldsymbol{N}\}$ to be covered, with the number of available security resources $\boldsymbol{m}$**.** In contrast with literature, we don't necessarily assume m to be less than N, because the utility function for the defender is only piece-wise continue as certain route changes can generate utility function to change a lot, then being able to allocate N resources in N nodes does not imply that all resources need to be allocated, this means that a trivial solution is not always the best solution. We assume that all resources are identical and may be assigned to any target. Associated with each node there are four payoffs defining possible outcomes for an attack at the target, as shown in Table 1 below. Two of these payoffs represent rewards for the attacker and the other two represent rewards for the defender. For each player we separate in two cases, depending on whether or not the node is covered by the defender allocations. The attacker's payoff for uncovered attacks for node i is $\boldsymbol{U_u^a}(\boldsymbol{i})$, and for a covered attack $\boldsymbol{U_c^a}(\boldsymbol{i})$. In the same way we define $\boldsymbol{U_u^d}(\boldsymbol{i})$ and $\boldsymbol{U_c^d}(\boldsymbol{i})$ as the defender's possible payoffs on node i.

	Covered	**Uncovered**
Defender	6	-13
Attacker	-10	17

Table 1

Table 1: Example payoffs for an attack on a specific target.

An important aspect of our model is that both players' utilities depend only on the nodes present in the attacked route and whether or not they are covered by the defender. Similarly for each route, it does not matter if the unattacked nodes are covered or not.

The defender's strategy space consists in assigning the m resources distributed optimally, in the sense of expected reward, to the set of nodes for protection. This in the form of a probability distribution for the m recourses defined as $< C(\boldsymbol{i}) >_{\boldsymbol{i}\in\boldsymbol{N}}$, indicating the probability in which one of the m resources is allocated in node i. We assume that each node may be covered by at most 1 resource.

On a similar way as the joint schedules introduced in ERASER security algorithm [18] we introduce $\boldsymbol{r} \in \boldsymbol{R}$ as a set of consecutive nodes connected by arcs, in this case, defining an attacked route. Each route must be a feasible set of nodes from an origin to a destination node. The feasibility of each route is determined exclusively from the structure of the graph. For this, we consider a set of directed arcs r to be a feasible route, if each arc's destination node is connected to the origin node of the subsequent one and there are no cycles on the route. This way, at the end of the final arc; the route reaches its destination node. A route r can be represented by the vector P_r = <P_{ra}>ϵ {0, 1}^{N} where P_{ra} represents whether or not arc a is utilized in route r, with a value of 1 or 0 in each case respectively. In our case, arc's attacking payoff is defined starting from its starting node, the origin node. Because each node i is covered with a probability of $C(i)$ and is uncovered with a probability of $1 - C(i)$, we define the utility of attacking an arc a_{ij}, defined from i to j exactly as the origin node reward. By this approach, the resulting utility function represents the rewards sum of all the involved nodes.

For a better understanding in the notations used in our work, we propose the following formalities: $C =< C(i) >_{i \in N}$ and a route $r =< r_j >_{j \in N}$, we say $r_j = 1$ if node j is included in of route r, and $r_j = 0$ if node j is not part of the route. On the other hand coverage $C(i)$ represents the probability in which a resource is covering node i.

It fits to note that we abuse the notation for denoting a strategy profile. Technically, C does correspond to a strategy by the usual literature definition, whereas the strategy that we call r doesn't actually denote a strategy but only a route. According to the formal definition, we should refer to attacker strategy vector as $a \coloneqq < a_s >_{s \in Routes}$ where $a_r = 1$ means that the attacked route is r, similarly $a_r = 0$ means route r is not being attacked. In our work we only considers attacking pure strategies where only one route is being attacked so with no loss of generality we denote r as the strategy where only route r is being attacked. Formally, strategy r correspond to vector $a \coloneqq < a_s >_{s \in Routes}$ with $a_r = 1$ and $a_s = 0 \forall s \neq r$. For simplicity purposes, and for the rest of the document, we identify an attacker strategy with the attacked route.

For a strategy profile $< C, r >$, the expected utilities for defender and attacker are respectively given by:

$$U^d(C,r) \coloneqq \sum_{i \in N} r_i \left(U_c^d(i)C(i) + U_u^d(i)\big(1 - C(i)\big)\right)$$

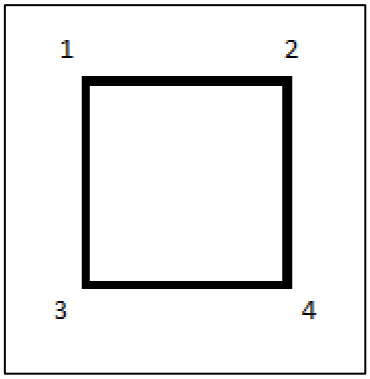
Equation 12

$$U^a(C,r) \coloneqq \sum_{i \in N} r_i \left(U_c^a(i)C(i) + U_u^a(i)\big(1 - C(i)\big)\right)$$

Equation 13

We adopt a Stackelberg's model in which the defender acts first and the attacker chooses a strategy observing the defender's mixed allocation strategy. Stackelberg games are common in domains were the attackers can observe the defender's strategy and act accordingly [10]. The standard solution concept is Strong Stackelberg's Equilibrium (SSE) [4] [5] [7], in which the *leader* (defender) selects an optimal mixed strategy based on the assumption that the *follower or slave* (attacker) will choose an optimal response, breaking ties in favor to the leader, this situation, as previously mentioned, occurs in the context that the defender can always assign an arbitrarily small amount of resources less to the target with most expected payoff and arbitrarily more to another target. Hence, the follower will choose the target breaking ties in favor of the defender. Also, because of linearity of utility functions, there always exists an optimal pure-strategy for the attacker in the *argmax* of the solution of the attacker, so we restrict out attention to this set in this paper.

Example: Consider a graph of 2 x 2 nodes (street corners), and arcs as in the figure below.



Graph 1

Graph 1: Basic example of a graph representation.

We consider 2 resources to be allocated between the 4 nodes. The set of feasible routes is R = {1-2, 1-3, 2-4, 4-2, 4-3, 3-1, 3-4, 1-2-4, 2-4-3, 4-3-1, 3-1-2, 4-2-1, 3-4-2, 1-3-4, 2-1-3, 1-2-4-3, 2-4-3-1, 4-3-1-2, 3-1-2-4, 4-2-1-3, 3-4-2-1, 1-3-4-2, 2-1-3-4}. The 2 resources will be allocated distributed on the 4 nodes depending on each of the nodes utility based on the point of view of the defender and, at the same time, considering which route is the one which gives the most expected payoff for the attacker.

Note: Our instance considers directed arcs instead of the non-directed arcs present in this example. But the logic is the same as described with the exception of directions allowed.

3.3 Understanding the Reward functions

As previously presented in this work, the utility functions are a sum of lineal functions; utility for route r for the defender and attacker are both respectively:

$$\sum_{i \in N} r_i \left(U_c^d(i)C(i) + U_u^d(i)\big(1 - C(i)\big)\right)$$

Equation 14

$$\sum_{i \in N} r_i \left(U_c^a(i)C(i) + U_u^a(i)\big(1 - C(i)\big)\right)$$

Equation 15

Only the nodes included in the route affects utilities for both players. In case of the attacker this utility also determines if he is willing or not to stay on that specific route or move to another one. This is, given the coverage, the attacked route necessarily corresponds to the best alternative for the attacker at all times. For this reason is that we study the behavior on the selected route as changing the coverage function C. Being more explicit, the attacker calculates in each instance of C, which route to attack. So he is also comparing the utilities of other routes at all times. As we previously stated, the utilities for each route are linear functions that depends on only the attacked nodes. So each attacked node adds a constant value and a linear one to the global utility of the attacker. Assume that given C, the best route for the attacker is r and node k is not included in r. This means that adding extra coverage on the node k will only decrease the utility on all routes that include node k and will not affect all the others. Similarly, for node j included in route r. Withdrawing coverage from node j will only increase the predilection of the attacker for routes with node j.

A crucial feature of the model is that payoffs depend only on the identity of the attacked target and whether or not it is covered by the defender. From a payoff perspective, many resource allocations are identical. We exploit this by summarizing the payoff relevant aspects of the defender strategy in a coverage vector C that gives the probability that each target is covered c_i. [13]

3.4 Joint schedules coverage mapping as an allocation vector

Many security domains, including both LAX and FAMS [13], involve allocating multiple resources to cover potential targets, an allocation of resources corresponds to a schedule for the defender, one of the computational issues of this was that the resources were needed to be allocated having into consideration that the targets were flights covered by The Federal Air Marshals Service (FAMS) inside the plane. Basically marshals had to do feasible flight connections and end their working day hometown. Each joint schedule represented a pure strategy consisting in an additional variable to consider. This representation is called normal form, which is the natural representation of a game; variables map directly the strategy space. Alternatively, compact form map variables into the targets, this representation is only possible if schedule restrictions are simple. The tradeoff with representing the problem in normal form is that a combinatorial explosion in the size of the variables needed and the payoff representation [13]. By calculating the aggregated coverage among the nodes instead of a pure strategy, we formulated a compact representation which is a lot more efficient in terms of resources.

We will demonstrate that a feasible compact form representation is equivalent to a normal form representation via a lineal transformation. From now on, we only refer to compact form.

Our approach is similar to other compact representations already studied [19] [20] [13]. This time, engineered specifically for security problems.

The pure strategies of the leader correspond to all possible coverage allocations with less than m resources among N nodes. This is, like dispatching up to m patrols, each of them destined to cover a specific corner or location. Differently from the FAMS approach, in our case, the leader strategy is not restricted to binary choices on the different targets but is able to choose a probability distribution and behave on a nondeterministic way, aiming for a long term solution in expectation.

Also, our model does not restrict the way the m resources are distributed. This assumption makes sense because unlike the FAMS approach, the defender need not to travel on a flight in order to cover a target and then to come back to its origin node, here the police may allocate the m recourses as he pleases between the N nodes. With this we identify the set of defender's pure strategies to have $L = \sum_{j \leq m} \binom{N}{j}$ components. Also, as we previously stated, we impose $N > m$ in order for the problem to make sense. On the other side, the follower only selects a single route to attack. Each route is composed of several nodes and the utilities, later described, are calculated depending on the included nodes.

Next we show how a feasible allocation vector is also a feasible lineal combination of schedules for the defender. Thus, representing the mixed strategies we previously mentioned.

Any security game represented in compact form can also be represented in normal form

Let us say we have an allocation vector as output of our problem $C\ =<c_i>_{i\epsilon\mathrm{N}}$, we wish to identify how allocations are interpreted and how this allocation corresponds to a mixed strategy from between feasible joint schedules. We define J as a $L\,xN$ matrix with the mapping of all possible joint schedules for the defender. As stated above these elements are vectors with up to m components with a value of 1 and the rest components with a value of 0.

We wish to identify a vector δ_s>0 :

$$\delta_\mathrm{s} : \{1,2,\dots,L\} \rightarrow [0,1].$$

Equation 16

$$\sum_{\mathrm{s}\epsilon\{1,..,\mathrm{L}\}} \delta_\mathrm{s} \leq 1$$

Equation 17

.

Such that,

$$J * \delta = C$$

Equation 18

Let us note that the Matrix J has a rank N because each canonic vector is also a pure strategy hence, J has complete rank.

The example below illustrates how a coverage vector corresponds to a mixed strategy, for $m = 2$ resources and $N = 3$ nodes:

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \\ \delta_5 \\ \delta_6 \\ \delta_7 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

Equation 19

For $m = 2$ resources and $N = 4$ nodes:

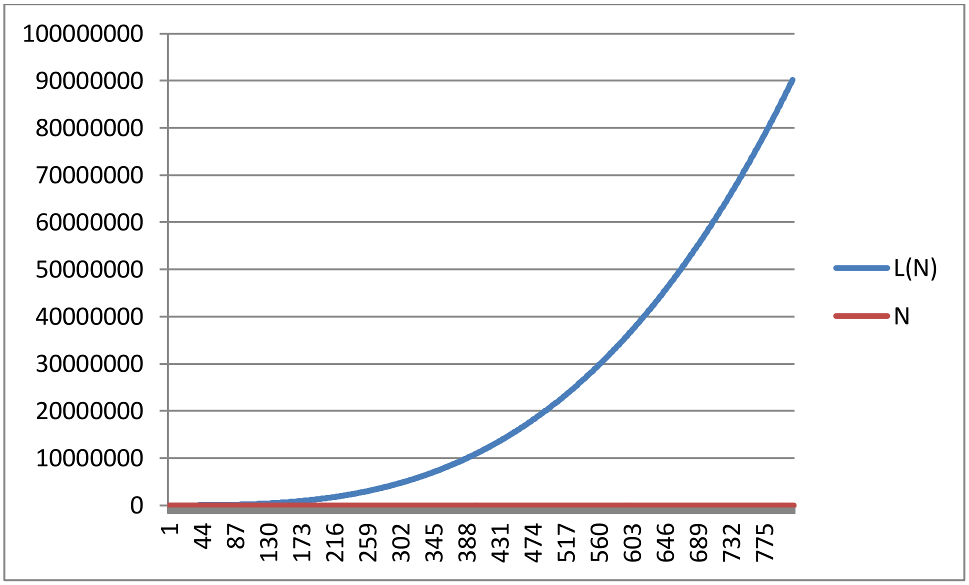
$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_{10} \\ \delta_{11} \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}$$

Equation 20

The example above illustrates the combinatorial explosion in calculations and variables just for passing from three to four nodes. Literature denote this structure as Compact Form. In compact form, each strategy is represented by N continuous variables. In contrast, the defender's strategy in normal form requires L(N) variables, while the attackers' strategy space remains the same.

$$L(N) = \sum_{j \leq m} \binom{N}{j}$$

Equation 21



Graph 2

Graph 2: Variables needed to describe our game in normal vs. compact form and m=3.

In case of the attacker, we are modeling a space that is similarly as big as the one in normal form just described. We consider a space of possible routes available for the attacker to assault. Our approach specifically considers instances with high number of routes.

4 An Efficient Allocation for a SSG with Attacked Routes (EASSGAR) Algorithm

In this chapter we start by identifying how fast the strategy space grows in bigger instances of the algorithm, later, we show how we use the leader-follower structure to decompose the algorithm in two phases where an iteration process is natural. Then we define some restrictions to utility functions in order for the problem to make sense and how to exploit the utility structure to identify when optimal conditions are met. Finally we show the two phases of the algorithm and which problem is solved on each of those phases.

4.1 The Size Problem

Our work focuses on problems that cannot be solved by ERASER algorithm because of its size. Our model considers scenarios where the follower's pure strategy space size is of about 2^n routes and coverage is a continuous function. In our case the size explosion of the problem comes from the follower's pure strategy space size, which in our case, is very big. Other works consider target attributes and characteristic in order to predict crime [21]. Our approach is mainly algorithmic.

For each iteration l we introduce a subset $R_l \subseteq R$. This subset of routes iteratively incorporates other routes to be considered when solving the problem. This way we will not be required to use all the restrictions naturally involved in the problem. We also show that our approach ends on actually reducing the number of required routes but, in some instances, delivering only a heuristic of the solution and not the global optimal solution we were originally looking for.

EASSGAR problems can be formulated as mixed integer programs, as in the ERASER model described above, in which adversary strategies are represented by integer variables $\boldsymbol{a} =< \boldsymbol{a_r} >$ with $\boldsymbol{a_r} = \mathbf{1}$ if route r is attacked and 0 otherwise. Like most security games on big networks we identify two main computational challenges for processing information as this formulation suggests. First, the space of possible strategies for the attacker suffers from combinatorial explosion: an EASSGAR instance with as little as 100 nodes, goes up to 2^{100} possible routes. Second, integer variables are a renowned issue for optimization [13]. Our algorithm presents a bi-level programming model where we only consider a subset of the restrictions, aiming to a heuristic of the solution. Like most lineal problems with lots of restrictions, in the end only a few of those are used, thus only a subset of them is enough to compute the optimal solution. In fact, we know that only the tight inequalities are necessary to calculate to optimum. On our case, these inequalities represent the possible routes for the follower to attack. The complication arises in determining which inequalities are going to end up being tight. We propose an algorithm that initializes its execution with just one single route and iteratively add new routes in order to converge to a reasonable solution.

In order to achieve the global solution, we will have to iterate, to a maximum of, as many possible routes our graph or instance allows. Each iteration l consists of considering a subset of all possible routes $R_l \subseteq R$ and solve for the defender. More specifically, given the attacker can select any route from the subset R_l, the defender allocate resources assuming that R_l is the complete set of routes the attacker can select, hence, the defender optimizes his resources to a subset of possible responses of the attacker. On the second level, the attacker solves the problem of which route to attack given the allocation C decided by the leader. The way to select the best route to attack is simply by solving a minimum cost network flow (MCNF) lineal problem with analog rewards as utility for arcs. This reflects that the attacker is always considering all possible routes and not only a subset of them each time he selects an attacking target, given the coverage C. On iteration l, if the selected route r' by the attacker second level problem is not in our set R_l, we proceed by adding the new route to the set of considered ones, this is: $R_{l+1} = R_l \cup \{r'\}$. Nevertheless, if the selected route r' was already present on R_l then, given the current defender allocation there are no more routes in the complete problem that increases the utility for the attacker and our algorithms stops iterating. Let us note that the final strategy $< C, r' >$ obtained by our algorithm, not necessarily corresponds to a global SSE of the problem as we will later demonstrate.

Depending on the starting point of the iterative process, the global optimal solution could not be met. We will later show some structural properties in the rewards that will let us understand certain characteristics of the solutions.

4.2 A Two-staged Decomposition approach to a SSG

Our algorithm considers an approach where we solve a Stackelberg security game (SSG) with arbitrary joint schedules and routes as attack targets. Our approach takes as input the utilities for each node being attacked of our graph for two players, attacker and defender, and solves an heuristic solution, corresponding to a coverage vector in form of a randomized allocation of defender recourses in order to maximize the reward for the defender, this reward can be presented as a social reward or, as in our case, economic costs.

For this, we propose an algorithm that uses a column and restriction generation approach, decomposed in two separate steps or phases where two different problems are solved, the master (defender) and the slave or follower (attacker) problem. The master solves for the defender strategy $\boldsymbol{C}$, given a restricted set of columns (i.e. routes) $\boldsymbol{P}$. The objective function for the slave is updated based on the allocation solution of the master, and the slave is solved to identify the best new column to add to the master problem (explained in detail later). If no new column can improve the slave´s solution, the algorithm terminates.

Algorithm 1 EASSGAR Column Generation

```
1. Construct initial set of routes $P$.
2. Solve "Master Problem($P$)".
    3. Get master coverage allocation $C$ from Master Problem.
4. Solve "Slave Problem($C$)"
    5. Get route $r$ from Slave Problem.
If r != "New Route" then
    6. Return $(\boldsymbol{C}, \boldsymbol{P})$
Else
    7. Update matrix P with route r.  $r -> \boldsymbol{P}$
    8. Repeat from Step 2
```

Observation: A strategy pair $< C, r >$ returned as a result of an ending process of our algorithm corresponds to an actual feasible strategy pair for the security game. This is, given the Master plays C, the best response for the Slave is r.

4.3 Characterization of Strong Stackelberg's Equilibrium

We wish to characterize situations identifying a Strong Stackelberg's Equilibrium (SSE) result as the end our algorithm. For this, we must first identify the number of possible multiple SSE and its respective common elements. We exploit structural properties of the problem and rewards in order to characterize the solutions. By considering a special class of games where the defender always benefits by having additional resources covering the attacked targets and the attacker, on his side, always get less benefit attacking a more heavily covered node. Also, we define that attacking a covered node is always prejudicial for the attacker's utility. These assumptions are quite reasonable on security games where being captured supposedly means getting into jail or paying a fine; and not covering an attacked node means social damage. Formally, we restrict payoff functions to be as $U_u^a(t) > 0 > U_c^a(t)$ and $U_u^d(t) < 0 < U_c^d(t)$. The first one helps for the attacker not to value more a longer route than a shorter one when deciding optimum routes and the latter represents the necessity for defender on covering attacked nodes and the reward of catching a delinquent when certain node is attacked, this tend to be classical assumptions in Stackelberg games [13]. This is similar in spirit to a zero-sum assumption, but somewhat less restrictive. It is well-known that zero-sum security games often admit more efficient solution algorithms because of the structure we give the payoff functions. Some algorithms exploit this class of games by using the crossed information between the rational players as a better understanding on other player strategies [22]. Note that $U_u^a(t) > 0 > U_c^a(t)$ also implies $U_u^a(t) > U_c^a(t)$, this means that we are describing a subset of this, last mentioned, kind of games, thus, usual results are mostly also valid in our domain.

4.4 Exploiting the utility structure

We proceed to further analyze the structure of our utilities functions in order to identify characteristics that will provide useful information on defining a useful stop condition. For this, we remember that only the attacked nodes provide utility on each player.

$$\sum_{i\in N} r_i\left(U_c^a(i)C(i)+U_u^a(i)\left(1-C(i)\right)\right)$$

Equation 22

We will introduce a different factorization of this same function and the parameters $\Delta U^a(i)$ and $\Delta U^d(i)$ denoting $(U_c^a(x)\text{-}\ U_u^a(x))$ and $(U_c^d(x)\text{-}\ U_u^d(x))$, thus factorizing we have utility functions rewritten like this:

$$\sum_{i\in N} r_i\left(\Delta U^a(i)(x)C(x)+U_u^a(x)\right)$$

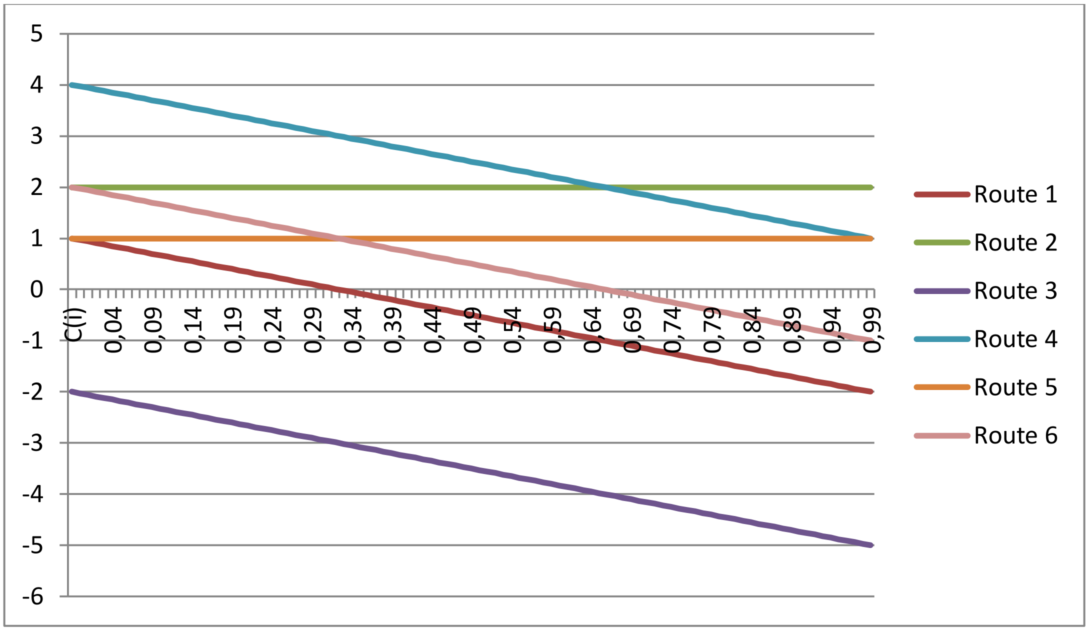
Equation 23

$$\sum_{i\in N} r_i\left(\Delta U^a(i)(x)C(x)+U_u^d(x)\right)$$

Equation 24

Note that the strictly better preferences conditions for the covered and uncovered utilities on each of the players, implies that$\Delta U^a(i)<0$ and $\Delta U^d(i)>0\ \forall i\epsilon N$. This concept is key because we identify that on each node x used for the selected route, that node provides utility equal to a linear function on $C(i)$. With $\Delta U^d(i)$ and $U_u^d(i)$ as slope and y-intercept respectively in case of the defender whereas $\Delta U^a(i)$ and $U_u^a(i)$ for the attacker. We also note that each route represents a lineal function on the vector C=$<C(i)>_{\mathrm{i\epsilon N}}$. Let us note that each route have non-negative gradient in the case of the defender and non-positive for the attacker. The slope for the utility in each direction $C(j)$ will be $\Delta U^d(i)$ or 0 for any route. It fits to highlight that a slope of 0 in direction j implies that node j is not included in selected route and a slope of $\Delta U^d(i)$, implies the opposite. This specific property is a key concept when describing the payoffs structure for each player because we can identify a form of parallelism on all the routes from the point of view of the payoffs function. This is, all routes that include node j will be parallel on direction j. The same goes for routes that do not include node j.

Graph 3, displays a *ceteris paribus* scenario of the payoff function of coverage in node i C_i, for a sample of routes. The graph represents **attacker's utility vs. coverage in node $\boldsymbol{i}$**, with the rest of variables C_j fixed. C_i Start from zero coverage $C_i=0$, to full coverage $C_i=1$. Note that all utilities only decrease or stay constant because those routes become less attractive as the node is more covered in expectance. We can also appreciate the decrements at two possible rates only (ΔU_i^a in case node i is present on the route and 0 if not).



Graph 3

Graph 3: *Ceteris paribus* example of payoff functions versus node coverage.

With route equations given as follows:

Route	ΔU(i)	Constant
1	-3	1
2	0	2
3	-3	-2
4	-3	4
5	0	1
6	-3	2

Table 2

Table 2: Parameters for example routes on Graph 3.

As our model considers that the attacker will always choose the most convenient route for him we can understand that in case of the attacker, and with a *ceteris paribus* assumption in every direction except of i, only up to 2 routes will only matter. For instance, in graph 3, route 4 starts being as the most attractive route for attacking. Note that this route have a strict negative slope, this means that this specific route includes node j. As coverage on this node increases, it becomes less attractive as a target for the attacker. All routes that also include node j will decrease its value at the same rate as route 4. This yields that the only way that a new route can start being attractive is that this new route does not include node i. We also conclude that given certain coverage $C(i)$ on node i if selected route does not include node j then adding more resources to this specific node does not produce a change on the chosen route for attack. Analogously, if a route includes node j, then decreasing coverage $C(j)$ will only result on a benefit for the attacker meaning that the attacked route will remain the same because no other route can become attractive whatsoever.

We sum up these just described characteristics of the defenders reward in **Property 1** as follows.

Property 1

Let $< C, r >$ be a feasible strategy profile for an instance of our problem. Having the information if a node is used or not in a certain route lets us guarantee a region of coverage where the same selected route r is also a best choice for the attackers.

$$r_j = 1 => \ \forall c_i < C(i), \qquad c_i \geq 0; \ < (C_{-i}, c_i), r >$$

Equation 25

Corresponds to a feasible strategy profile

$$r_j = 0 => \ \forall c_i > C(i), \ \ c_i \leq 1,; < (C_{-i}, c_i), r >$$

Equation 26

Corresponds to a feasible strategy profile

Where coverage (C_{-i}, c_i) correspond to the same coverage C except for node i, that is replaced by c_i instead of $C(i)$.

What we are basically saying is that if a certain route is selected this immediately implies that within a certain region that specific selected route must also be a valid best-response kind of strategy. This way we can calculate a region where our solution is valid; in other words, for any feasible strategy profile $< C, r >$, we define:

$$C_r \coloneqq \{\gamma \in [0,1]^N : \gamma_i \leq C(i)\ \forall i: r_i = 1\ \&\ \gamma_i \geq C(i)\ \forall i: r_i = 0\}$$

Equation 27

Property 1 let us assure that elements of C_r do not change route r as the best response for the attacker, formally:

$$< \gamma, r > \text{ Corresponds to a feasible strategy profile } \forall \gamma \in C_r$$

The demonstration takes advantage of the property concluded on **graph 3**. We start with the fact that we are certain on the veracity of the property if only one variable is modified.

As we previously showed, the key understanding best-responses restrictions is the reward function of the attacker. For this reason we study how does the utility for each route changes when we move variable C. We already noted that if certain node j is not being attacked, adding resources to it will only maintain the route reward and decrease the reward for all the routes that do not include node j. For all routes that do include node i, we can decrease all the amount of coverage we want because it will only make it more attractive for the attacker in relation to all the other routes. Note that routes without node j present no changes on the reward when variable $C(j)$ is modified. Clearly, other routes that include node i will also increase their reward but our model structure let us guarantee that this decrement will occur at the same rate as on route r, with this we conclude that the same route will remain being the best-response for the attacker when we do movements as we just described.

This yields that region C_r represents an N-rectangle where we guarantee chosen route r will remain being the selected one as a best-response.

4.5 Master problem

Master problem (Equations 28 to 35) calculates a probability distribution for allocating resources from among the nodes of a graph $G(N, A)$ that maximizes the defender reward.

Master problem operates directly on columns of matrix $\boldsymbol{P}$ with $P_{(i,j,r)} = 1$ if route r includes arc (i, j) and 0 if not. Vector $\boldsymbol{C}$ describes the coverage allocation in form of a probability distribution. Equations 31, 32 and 33 enforce the SSE condition that the attacker chooses a best-response, mirroring similar constrains from the ERASER engine. The defender expected payoff for route r is given by $\sum_{(i,j)\in A^*} P_{(i,j,r)} \left(U_c^d(i)C(i) + U_u^d(i)\left(1 - C(i)\right)\right)$. Similarly, the attackers expected payoff is given by constrains from equations 32 and 33. Constrain 30, 34 and 35 that only a single route may be attacked. Let us note that last two mentioned constrains are only active for the single route r^* attacked $\alpha_{r^*} = 1$ whereas for all other routes the inequality compares to M which is a very large number (we used instances of M=1000*V where V is the largest payoff from among all players). Variable C is defined in a way it ensures that the allocation of resources defines a probability distribution over them.

$$max\ \ d$$

$$s.t.$$

$$\sum_{i\in N} C(i) \le m$$

Equation 28

$$C(i) \le 1 \quad \forall i \epsilon N$$

Equation 29

$$\sum_{r\in R} \alpha_r = 1$$

Equation 30

$$d - \sum_{(i,j)\in A^*} P_{(i,j,r)} \left(U_c^d(i)C(i) + U_u^d(i)\big(1 - C(i)\big) \right) \leq (1 - \alpha_r)M, \qquad \forall r \epsilon P$$

Equation 31

$$0 \leq l - \sum_{(i,j)\in A^*} P_{(i,j,r)} \left(U_c^a(i)C(i) + U_u^a(i)\big(1 - C(i)\big) \right) \leq (1 - \alpha_r)M, \qquad \forall r \epsilon P$$

Equation 32

$$\sum_{(i,j)\in A^*} P_{(i,j,r)} \left(U_c^a(i)C(i) + U_u^a(i)\big(1 - C(i)\big) \right) \leq l, \qquad \forall r \epsilon P$$

Equation 33

$$C, \alpha \geq 0$$

Equation 34

$$\alpha_r \in \{0, 1\} \ \forall r \in P$$

Equation 35

Note that instead of using the set of arcs A we utilize A^*, where A^* is the set of arcs including all the arcs that goes from and to the auxiliary source and destination nodes. Let node 0 be the origin node and node $n+1$ the destination node, this is:

$$\mathrm{A}^* \coloneqq \mathrm{A} \cup \{(0, \mathrm{i}) \colon \mathrm{i} \in \mathrm{N}\} \cup \{(\mathrm{j}, \mathrm{n}+1) \colon \mathrm{j} \in \mathrm{N}\}$$

Equation 36

Equations 28, 29 and 34 restricts variable C to generate a feasible allocation in the form of a mixed strategy for m resources.

4.6 Slave problem

The slave problem finds the best route, in the sense of rewards, to include in P. Generally, we should calculate reduced costs over all the routes not considered but as our approach considers a large number of possible routes this would be very inefficient. Instead, we solve another Mixed-integer Lineal Problem (MILP) which finds the optimum new route to include in P that increases the value of the slave's objective function. Given a coverage allocation vector $< \boldsymbol{cov}(i) >_{i \in \mathrm{N}}$ we solve a Minimum Cost Network Flow (MCNF) problem that finds the optimal route; in the sense of best response for the attacker. A feasible route corresponds to a feasible flow from the auxiliary origin node to the auxiliary destination node taking into consideration that the route should maximize utility for the attacker. Is in this step that our model defines constrains in order for the new route to be feasible in the algorithm.

For an instance of our problem we build a graph $\boldsymbol{G}$ as follows. Given a set of nodes $\boldsymbol{N}$ and arcs $\boldsymbol{A}$ we create two auxiliary nodes that represent a source and a destination, $\boldsymbol{s}$ and $\boldsymbol{d}$ respectively. From here, we add arcs with no utility for either player from the source node to each non-auxiliary node and from each non-auxiliary node to the target. Then, we impose that a flow of 1 has to come out from the source node and a demand of 1 has to arrive the destination node. A route in this context would be a set of nodes that define a routing from source to destination passing through our graph in the most rewarding flow (one target correspond to a selection of a set of nodes). The way a flow is defined, ensures the schedule is a feasible column to add to $\boldsymbol{P}$, we also say that the selected route is feasible to our graph $\boldsymbol{G}$. The capacities are all set to 1 and the default cost for each arc is 0. Additionally, further arc values are included in order to represent the utility for players on the usage of each arc.

In our case we consider Nodes as the main source of rewards for each player, we simply say that the utility for each arc (i, j) is equal to the reward of the initial node, or in this case node i.

$$U^d_{(i,j)} = U^d(i)$$

Equation 37

$$U^a_{(i,j)} = U^d(i)$$

Equation 38

Constrains 43 through 49 are meant to generate a flow of 1 from origin to destination. Constrain 50 is used as the utility function for the slave problem, meaning we are maximizing that value.

$$max\ l$$

$$s.t.$$

$$\sum_{(i_0,j)\in A^*} f_{(i_0,j)} - \sum_{(i,i_{0,})\in A^*} f_{(i,i_0)} = 0 \quad i_0 \in N$$

Equation 39

$$\sum_{(0,j)\in A^*} f_{(o,j)} = 1$$

Equation 40

$$\sum_{(i,0)\in A^*} f_{(i,0)} = 0$$

Equation 41

$$\sum_{(i,n+1)\in A^*} f_{(i,n+1)} = 1$$

Equation 42

$$\sum_{(\ n+1,j)\in A^*} f_{(n+1,j)} = 0$$

Equation 43

$$1 - f_{(i,j)} - f_{(j,i)} \geq 0, \qquad (i,j) \in A^*$$

Equation 44

$$\sum_{(i_0,j)\in A^*} f_{(i_0,j)} \leq 1, \qquad i_0 \in N$$

Equation 45

$$\sum_{(i,j)\in A^*} f(i,j)\left(U_c^a(i)cov(i) + U_u^a(i)\big(1 - cov(i)\big)\right) \geq l$$

Equation 46

$$f \geq 0$$

quation 47

The output of this problem is the variable f that represents a route within the graph. This route was determined by rewarding the best payoff for the attacker from among all routes. This corresponds to the route that is the best candidate to be added to $\boldsymbol{P}$, given the coverage of the defender. If this route was previously not considered as part of the master's allocation problem then, we add it to P and the process iterates. Basically we are representing a situation where the police allocate assuming they are fully aware of all the possible terrorist strategies. Then the attackers, based on the decision made by the police, identify which is the most profitable route from among all of them. If the route was not previously considered by the defenders, then they add it to matrix P representing as if they have that route *in mind* when re-allocating the resources for coverage on the next iteration. Otherwise, if the route was already considered by the master in the previous iteration, then no new other route can improve the solution of the attacker computed by the master because the allocation is already considering all the best choices for the slave.

Note that we are not declaring that a given output $< C, r >$ of the algorithm will represent an actual SSE for the original problem. On next chapter, we illustrate a counter-example where an output of the algorithm does not represent a global optimal solution of the problem but only a local one.

Notably, a strategy profile $< C, r >$ obtained as output of our algorithm is not guaranteed to comply with the 3 properties that characterize a SSE (equations 1, 2 and 3). Nevertheless, it does always accomplish to be the best response for the attacker given the coverage, equation 2.

5 Algorithm Results Examples

Chapter 5 starts with a motivating example where optimal solution is not met as a result of the stopping conditions of the EASSGAR algorithm. This example evidences structural properties that let us identify situations where the optimal solution is guaranteed as a result of the algorithm. Finally, we describe the zero-sum scenario and prove that found solutions on these instances are always a Strong Stackelberg's Equilibrium, or what is the same, the optimal solution for the problem.

5.1 Motivating example where stopping conditions do not meet an optimal solution

Let us consider a scenario that consists in allocating up to $m = 2$ resources between $N = 2$ nodes and 3 possible routes as the available strategies for the attacker; we summarize these conditions on graph 4 further below.

Table 3 define rewards for the defender on each of the nodes, similarly, table 4 describe rewards for the attacker:

	Covered	**Uncovered**
Node 1	4	-1
Node 2	2	-2

Table 3

Table 3: Rewards for the defender on counter-example.

	Covered	**Uncovered**
Node 1	-3,2	0,8
Node 2	-2,4	0,6

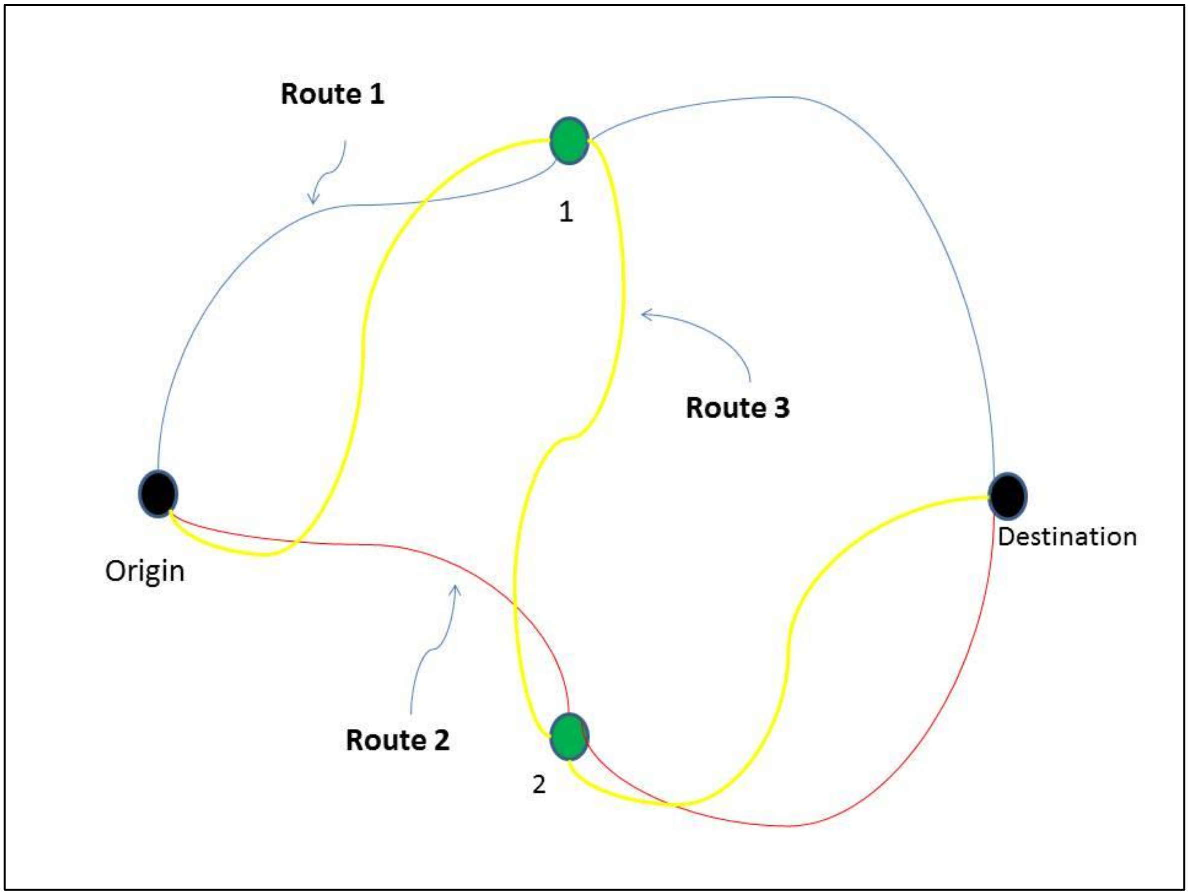
Table 4

Table 4: Rewards for the attacker on counter-example.

With this structure of nodes we can define up to 3 possible routes, these routes consist in:

- Route 1: Node 1
- Route 2: Node 2
- Route 3: Nodes 1 and 2

Graph 4 below shows a visual representation of the routes present in our example. Each route consists in a different set of subsequent nodes.



Graph 4

Graph 4: Visual representation of a 2 node scenario and available routes.

The attacker rewards define the regions where each of the routes is the one preferred by the attacker, given the coverage C. Equations 48, 49 and 50 display attacker’s utility function for each of the 3 routes.

$$L1: \; -4C_1 + 0{,}8$$

Equation 48

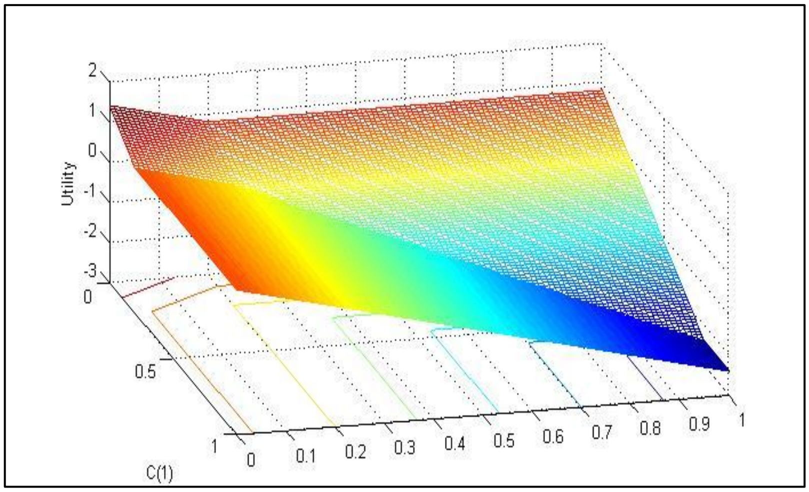
$$L2: \; -3C_2 + 0{,}6$$

Equation 49

$$L3: \; -4C_1 - 3C_2 + 1{,}4$$

Equation 50

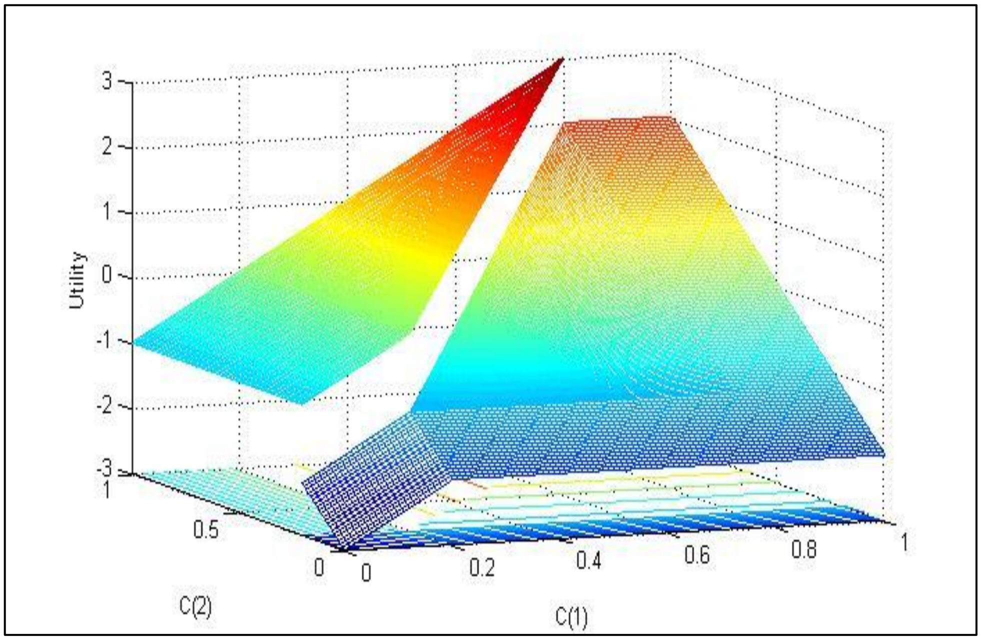
These utilities determine which of the routes will be preferred by the attacker for each allocation. Calculating the regions where each of these routes gives the maximum return for the attacker, we divide the area as displayed in graph 5, below. Each slope represents a different preferred route.



Graph 5

Graph 5: Attacker's utility function example.

Similarly, the utility structure for the defender can be better understood with graph 6, below. Each piece of surface represents the utility obtained by considering the route with most return for the attacker.

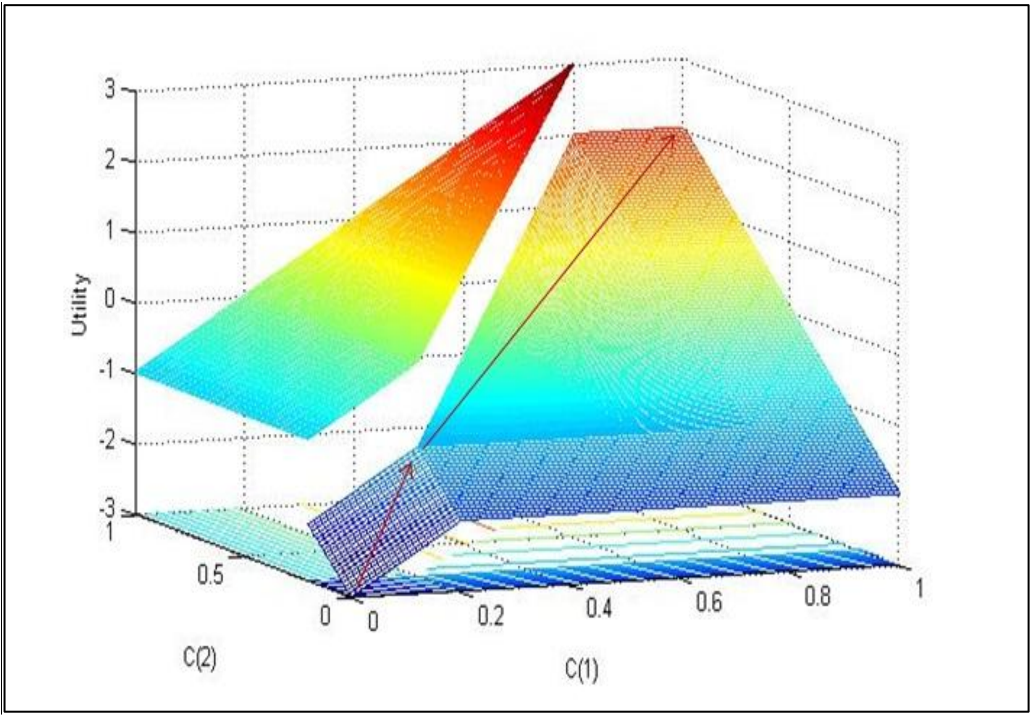


Graph 6

Graph 6: Defender’s utility function example.

In this case, the exactness of the solution depends exclusively on the initial conditions arbitrarily set at the beginning of the algorithm. For the example’s sake, we start the algorithm with values $C_1 = 0$ and $C_2 = 0$. This yields to incorporate route 3 as the initial one. After first iteration the master returns a coverage of $C_1 = 1$ and $C_2 = 1$. Similarly, the slave verifies if a new route applies to be included. In this case route 2 is incorporated to the algorithm. Finally, the master solves again to deploy a coverage of $C_1 = 1$ and $C_2 = 1$. Finally, the slave verifies that no new route qualifies to be added, therefore, the algorithm stops and delivers the final coverage and route attacked as $C = (1, 1)$ and $r = 2$, with a defender’s utility of 2. Nevertheless, we can verify the optimum values are $C = (0.8, 1)$ and $r = 1$ with a defender utility of 3. Thus, the obtained strategy profile is not an optimal solution for the problem, more specifically, found solution is does not correspond to a SSE.

The process just described can be further understood with graph 7, just below:



Graph 7

Graph 7: Iteration process described as a counter-example.

On the other hand, if we define as initial conditions the coverage of $C_1 = 0$ and $C_2 = 1$. Then the algorithm iterates as follows. Master deploys a coverage of $C_1 = 1$ and $C_2 = 1$. Slave answers by adding the route 1 to the problem. Finally the master returns $C_1 = 0{,}8$ and $C_2 = 0$ which is the value that returns the most output for the defender and as we previously knew, hence we reached a SSE or optimal solution of the problem.

The example just described has a direct implication that the algorithm does not necessarily ends on an optimum solution of the problem. For this reason is that we need further analysis and details on how the utilities and rewards are structured in order to understand convergence issues. On next chapter we present the zero-sum scenario and prove that EASSGAR find a SSE in this context.

5.2 Zero-Sum Utilities scenario

For our studies so far, the only restrictions made for the rewards data is that for each node i, $U_u^a(i) > 0 > U_c^a(i)$ and $U_u^d(i) < 0 < U_c^d(i)$. Zero-sum describes situations where the adding all players rewards, and consequently subtracting the costs, it always sum up to zero. In our case, this will represent a dynamic where the reward obtained by the defender is the exact opposite from the one obtained by the attacker for the same action. Equations 51 and 52 illustrate the zero-sum condition:

$$U_u^a(i) = -U_u^d(i)$$

Equation 51

$$U_c^a(i) = -U_c^d(i)$$

Equation 52

This gives the following intuition: money society saves when covering a possible attacked spot is the exact amount of money that would be stolen by a thief in that node. Also, being captured on a certain node gives the society the same amount of utility lost by the captured attacker. We extend these implications not only to nodes, but also for routes. It is very easy to verify that for any feasible strategy profile $< C, r >$, utility for both players will be the exact opposite. As a direct implication of equations 51 and 52; equation 53 shows utility for the defender at the left side; and the additive inverse utility of the attacker on the right side:

$$\sum_{i \in N} r_i \left(U_c^d(i)C(i) + U_u^d(i)\big(1 - C(i)\big) \right) = -\sum_{i \in N} r_i \left(U_c^a(i)C(i) + U_u^a(i)\big(1 - C(i)\big)\right)$$

Equation 53

Since attacker's utility is a continuous function, is direct that defender's utility will be continuous as well. This last aspect is a key concept because the master will have an actual idea of what directions the utility functions grow and where not, whereas the general case supports scenarios where defender's utility presents no monotony whatsoever. We further discussed and illustrated this case with a counter-example on previous chapter in this same work.

Our algorithm considered 2 phases: one where the master proposes a coverage allocation, and another one, where the slave proposes a better route, in case this route exists. This iterative process is very similar to the Newton method for finding optimal values of a function [23]. Basically the master moves the coverage in the most rewarding direction, as utilities are lineal functions the gradient could be represented by vector $< r_i \Delta U^d(i) >_{i \in N}$; with zeros in directions corresponding with non included nodes (i.e. $r_i = 0$) and a slope of $\Delta U^d(i) > 0$ for directions involving nodes included in r (i.e. $r_i = 1$).

Let us note that utility for the defender is not only continuous but also non-decreasing as our gradient evidences.

We now prove that within the context of a zero-sum scenario, the algorithm ends on a SSE for the security game. This is a global exact solution and not only a heuristic as in the general case.

Theorem 1

Given a zero-sum instance of the EASSGAR algorithm with m resources, let $< C, r >$ be the resulting strategy profile obtained as the output after the stopping conditions of the algorithm are met. The following affirmation is true:

Strategy profile $< C, r >$ corresponds to a SSE of the security game

As cases with $m = 0$ and $m = N$ are both trivial, further on we suppose $N > m > 0$. Clearly the first one only allows for no resources to be allocated and the latter plays a full coverage strategy. We justify last affirmation by mentioning the non-decreasing properties of defender's utility.

Proof

To prove $< C, r >$ optimality, we need to verify whether or not it accomplishes the SSE conditions described in equations 1, 2 and 3. As the slave problem solves the MCNF that obtains the most reward from among all of possible routes, is clear that attacked route r is a best response to coverage C. Besides, defender's reward continuity and zero-sum payoffs directly imply ties are never needed to be broken in favor of the leader, because there can be no ties. If two routes have the same utility for the attacker, then they provide the same utility for the defender.

We only need to prove is that C provides the most benefit possible in the game for the defender, i.e. equation 1. By contradiction, we start supposing there is another strategy pair $< C^{*}, r^{*} >$ that returns a better reward for the defender than $< C, r >$.

We define l^{*} and l as value of the defender's utility for strategies $< C^{*}, r^{*} >$ and $< C, r >$ respectively. Formally we say:

$$l^{*} = \sum_{i \in N} r_i^{*} \left(U_c^d(i) C^{*}(i) + U_u^d(i) \left(1 - C^{*}(i)\right) \right)$$

Equation 54

$$l = \sum_{i \in N} r_i \left(U_c^d(i) C(i) + U_u^d(i) \left(1 - C(i)\right) \right)$$

Equation 55

Note that by defining $< C^{*}, r^{*} >$, we supposed $l^{*} > l$.

On the given case that routes were exactly the same, this is $r^{*} = r$. This would mean that when the master problem was solved, route r^{*} was already being considered as part of available routes. Also, the only way that no new route was proposed for including on last iteration is that after allocating resources C on the nodes, the attacker couldn't find a better route for him to attack. Or what is the same, C and r^{*} also form a feasible strategy pair as route r plays as a best response for coverage C. This yields that no greater reward could be feasibly obtained on route r then $< C, r >$ should at least provide the same reward than $< C^{*}, r^{*} >$. This result contradicts the $\mathrm{l}^{*} > l$ hypothesis so we conclude that both routes are necessarily different, $\mathrm{r}^{*} \neq \mathrm{r}$.

Utility for the defender can be understood as the minimum of a family of linear functions, this kind of functions are known to be concave [24] This fact implies that the plane defined starting from C of the utility function will correspond to an upper bound for the defender's reward. Besides, since $U^d(C, r) < U^d(C^{*}, r^{*})$, then the plane starting from C on route r complies:

$$U^d(C^*, r^*) \leq \ U^d(C, r) + \nabla_C U^d(C, r) * (C^* - C)$$

Equation 56

Note that the left side only includes route r^* to make C^* a feasible allocation.

We say $\nabla_C U^d(C, r)$ is the gradient in direction of U^d on point C. Also, for simplicity issues when calculating the gradient, we extend the definition of derivative on C_0 by utilizing the concept of lower derivative when inconsistencies are found, formally:

$$\nabla_i U^d(C_0, r) = lim_{\varepsilon \to 0} \frac{U^d(C_0, r) - U^d(C_0 - \varepsilon e_i, r)}{\varepsilon}$$

Equation 57

Let us note that this limit always exists and is also unique for this kind of functions. With $\varepsilon > 0$ and e_i as the i-th canonic vector of $\mathbb{R}^N$. Note that linear plane defined on equation (56) corresponds exactly to the utility function for the defender allocating resources on C when attacked route is r. Even thought strategy pair $< C^*, r >$ is not feasible, it would at least imply that the attacker would have proposed a new route to incorporate. Evidently, defender would have strictly preferred to play C^* than C because it would have provided more reward for the defender, in fact: $U^d(C^*, r) \geq U^d(C^*, r^*) > U^d(C^*, r^*)$. This contradicts the optimization process done by the master problem, hence, it wouldn't have met stop conditions as we initially assumed and additional routes would have been proposed. Then we conclude there are no other feasible solutions that increase the defender's utility.

Now that we have verified that conditions shown in equations 1, 2 and 3 are fully met for any instance of a zero-sum running on EASSGAR, thus it corresponds to a SSE for the problem.

6 Experimental results

The EASSGAR algorithm has been tested on several scenarios using real world networks representing Santiago and simple grids with randomized utilities for testing purposes. We put special attention to Santiago's 119 nodes network were we set up a real world data set provided by the results of a clustering analysis to Santiago's downtown between 2002 and 2004 obtained from 1st Police Station at Santiago, Chile [26]. It is important to note that the problem solved in [26], which motivates the instances used here, differs from this thesis as it does not consider a route focus in the modeling. Our computational results are separated in three groups. The first explores the solution time sensitivity of the EASSGAR algorithm as we change the problem size and number of resources. Our second set of results present a Zero-Sum instance for downtown Santiago; we recreate experiments with different amount of resources and show how they yield different results. Finally, or last set of results compare the solution time and quality of our method with a previously existing algorithm (ERASER-C) which has to consider the full set of possible routes for the attacker.

Construction for the zeros-sum scenario utilities data was made by establishing that the expected reward for attacking an uncovered node was exactly the average stolen on that node; whereas the cost of not covering an attacked node is that same amount of average money stolen. Similarly, capturing and being captured was defined as saving or losing the total amount of money historically stolen in that corner for each player respectively.

We summarize this payoff structure for attacked node i on table 5 below:

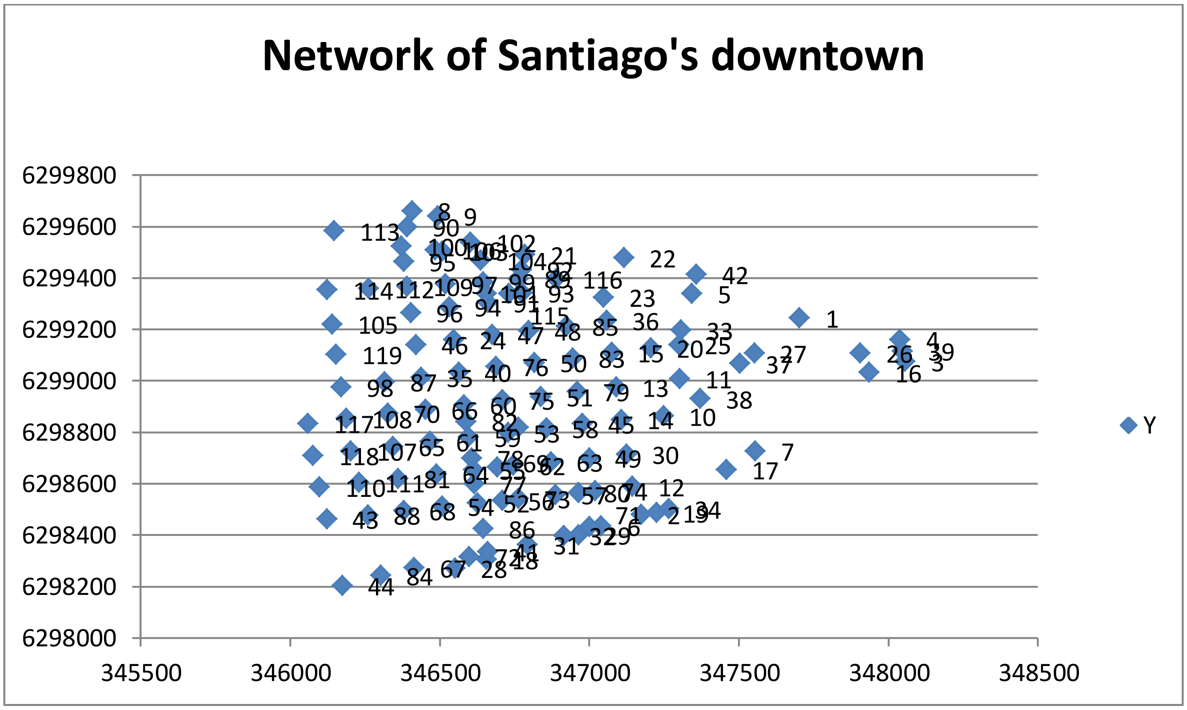
	Covered/ Capture	**Uncovered/ Steals**
Defender	$Total\ Stolen(i)$	$-Average\ Stolen(i)$
Attacker	$-Total\ Stolen(i)$	$Average\ Stolen(i)$

Table 5

Table 5: Data structure for the Zero-sum scenario.

With this we present results based on performance type, quality of solutions and amount of iterations needed. Our baseline for comparing EASSGAR is the RUGGED algorithm [27]. RUGGED algorithm is very similar to ours in the sense of modeling a Stackelberg security problem where the attacker moves in the form of routes, the main difference is the structure of utilities. In RUGGED case, the utility function is route oriented whereas in our case is node oriented.

Santiago´s network structure for nodes is displayed on graph 8 below,



Graph 8

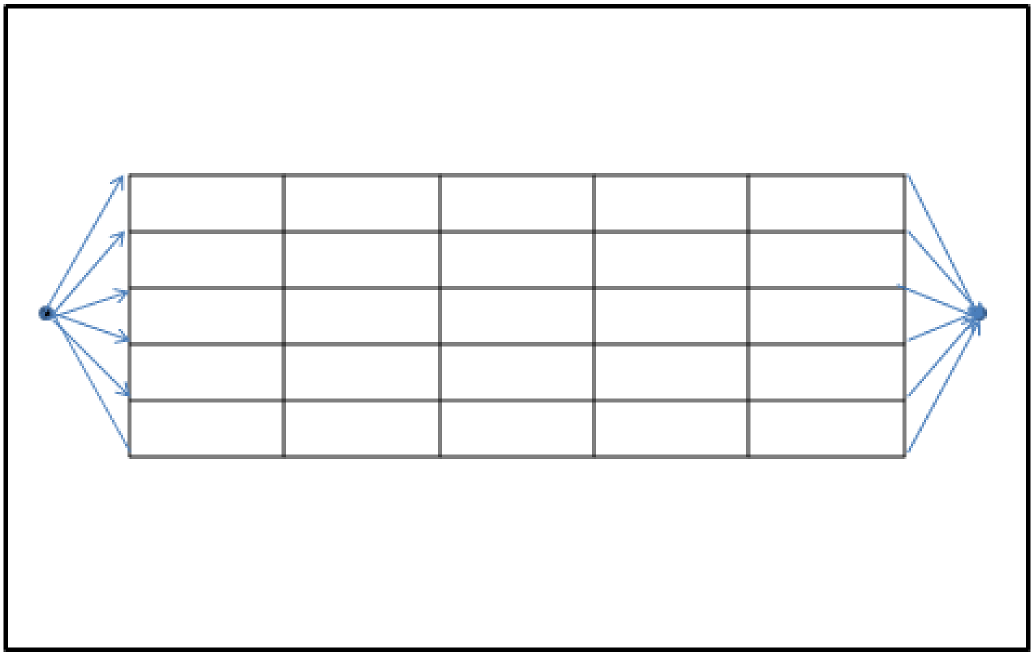
Graph 8: X vs. Y representation of Santiago's network numbered nodes.

The arcs structure was constructed manually based on a www.mapcity.cl map (available on graphs 17 and 18 further ahead in this document). Also note that it is only important to be aware of which nodes are used in each route and not the order of these nodes. A same set of nodes in different order does not change utilities value for either player whatsoever. For that reason, technically we refer to those equivalent routes as the same route. We are interested in differences where routes are distinguished by the involved nodes and not the order in which the route is walked.

6.1 Algorithm Dynamics Analysis

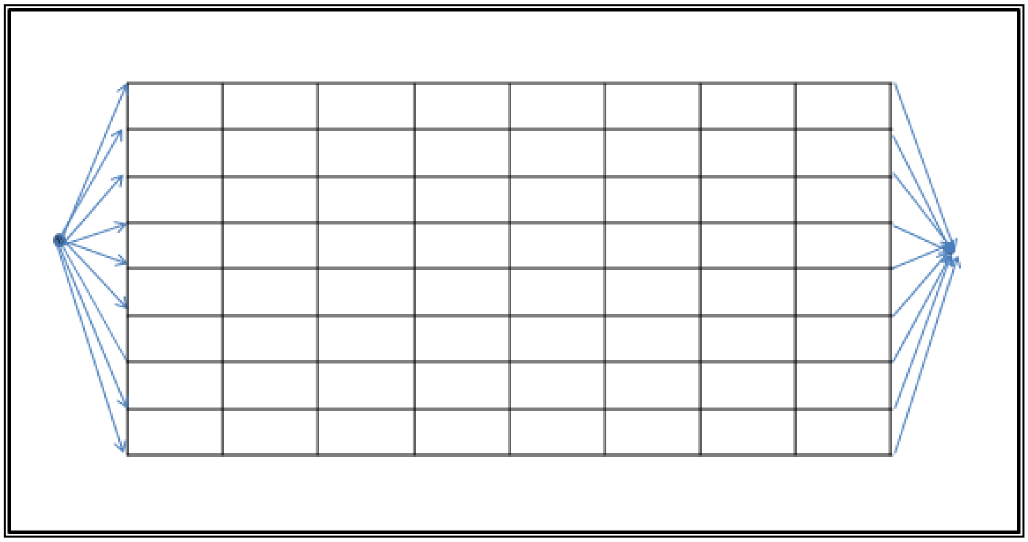
As a benchmark, we generated an instance of our algorithm on a grid of 10x10 nodes with directed arcs that goes from left to right and down to up. This is, routes are set of subsequent nodes from node (1, 1) up to node (n, n) where on each node, attacker can move up or right. With this instance we can compare results obtained between our EASSGAR algorithm and the complete problem. Both problems were solved with m=3 police resources.

In order to analyze the algorithm and its scalability capabilities we generated random instances of a network with different number of nodes and resources. All the data used for these instances was randomly generated complying with the motivational conditions described in chapter 4.3. The graphs correspond to a squared grid of $n \ x \ n$ nodes. See graphs 9 and 10, below.



Graph 9

Graph 9: Visual representation of the testing grid for n=6.

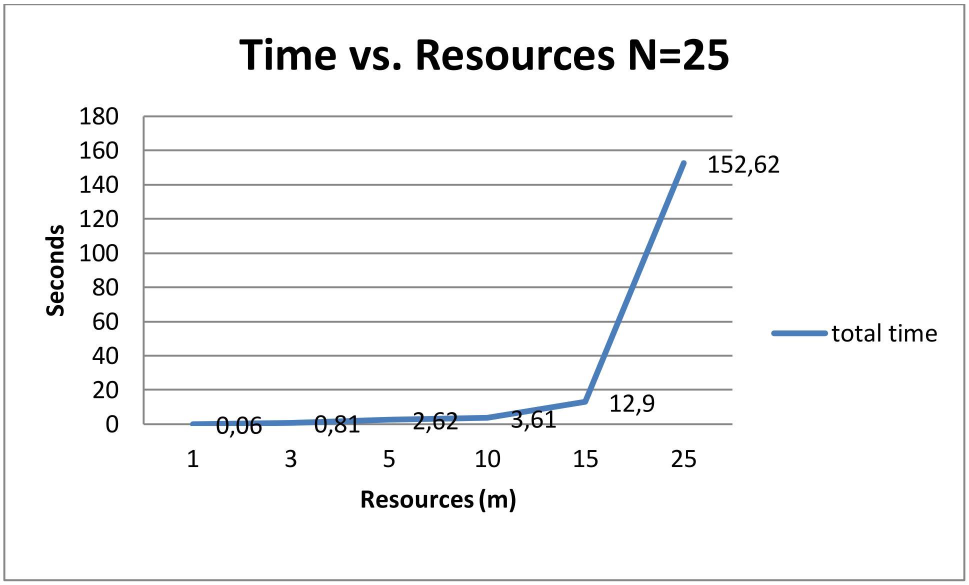


Graph 10

Graph 10: Visual representation of the testing grid for n=9.

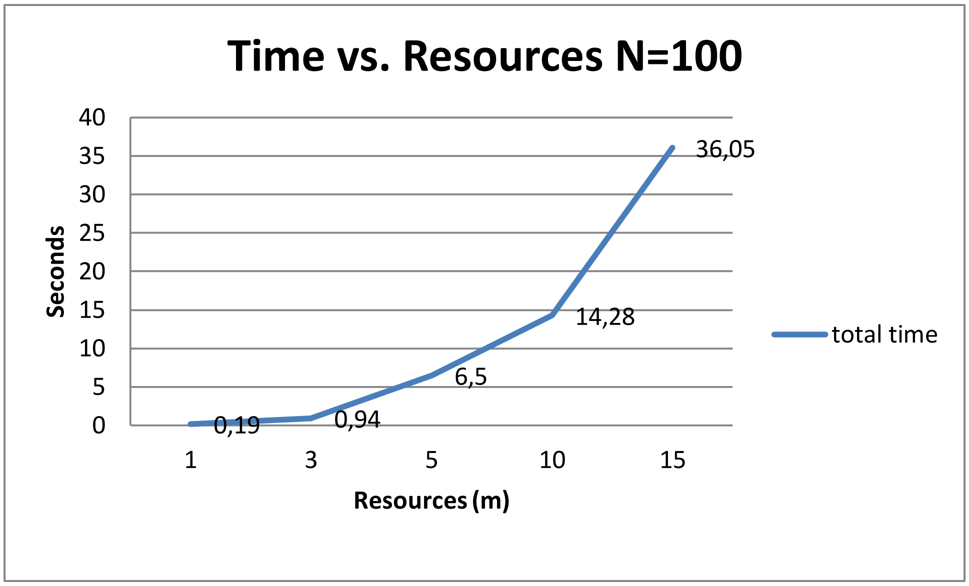
On each instance we generated scenarios just like the ones displayed above. For each node random feasible numbers were generated as utility for each of the players. CPU processing and user times were registered for each of the scenarios tested.

We noted an increase in the time of processing was positively correlated with the number of nodes and also with the number of resources. First and as expected, more nodes meant longer processing times. Similarly, as more resources were considered, a bigger number of iterations were needed in order to meet stopping conditions, hence, longer times. All data present in graphs below correspond to data collected during our studies. For the complete data set used, see Annex 1.



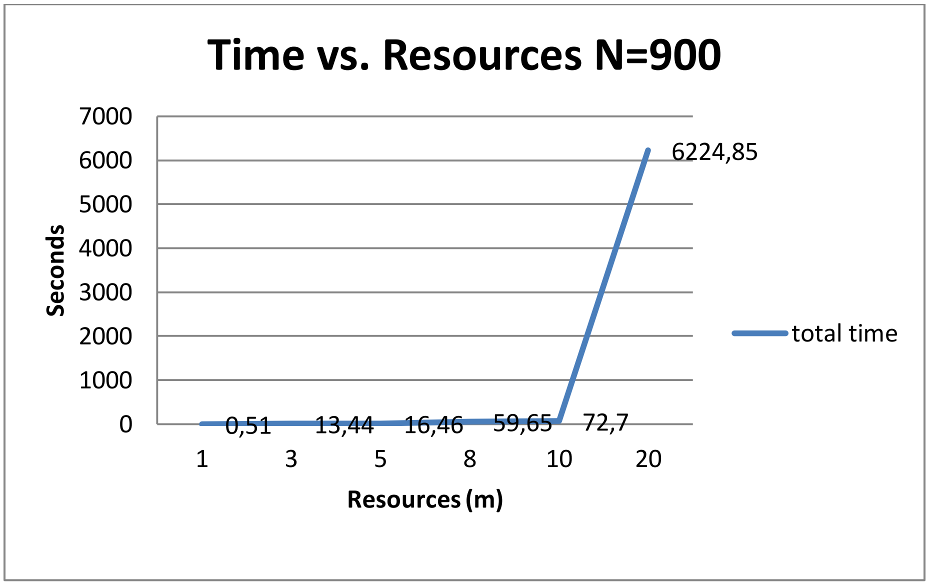
Graph 11

Graph 11: Time vs. Resources N=25.



Graph 12

Graph 12: Time vs. Resources N=100.

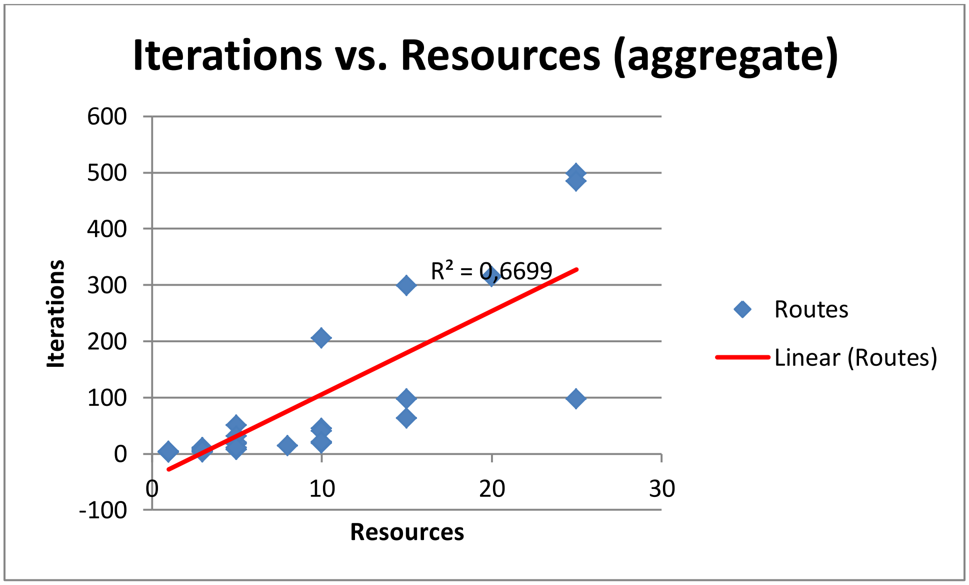


Graph 13

Graph 13: Time vs. Resources N=900.

Graphs 11, 12 and 13 displays the increment of time needed as more resources are available. This is somehow expected, because allocating more resources generates a bigger feasible coverage region on strategy space for the defender, implying that more route restrictions will probably be tight in the attacker's problem.

Our studies revealed that increasing the number of available resources, increases the amount of iterations needed by EASSGAR in order to meet stopping conditions. This result yields that more routes are needed before achieving stopping conditions when more resources are available. Note that this result is consistent with structural properties described in chapter 4. More specifically, a big number of resources imply a big feasible region of routes for the attacker to consider. Thus, the algorithm needs to generate more routes in order to propose, at least, a local optimal solution for the algorithm.

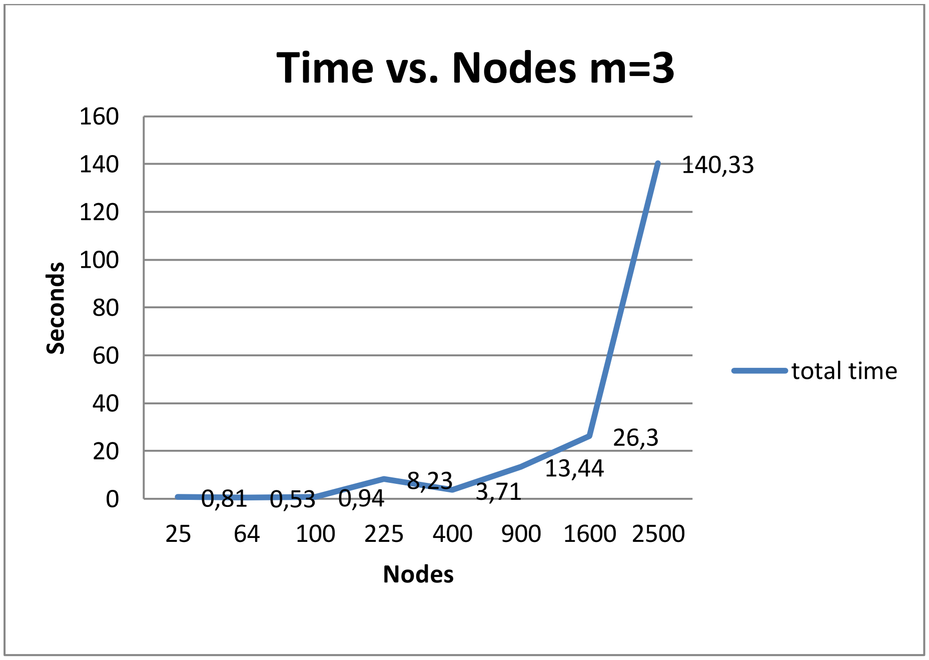


Graph 14

Graph 14: Iterations vs. Resources (aggregate)

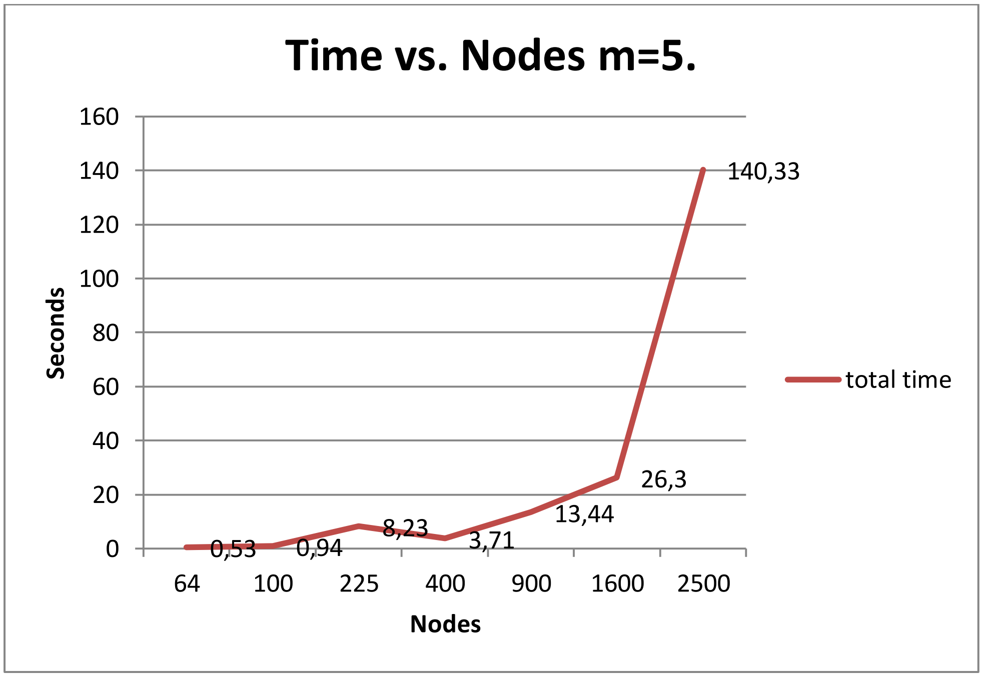
Our instances include scenarios with data sets from 25 to 2500 nodes and from 1 to 25 resources. The greatest number of iterations was 499 with N=64 and m=25. A scenario with those characteristics has up to 2^{64} possible routes. Instead, we considered only 499 restrictions for describing routes and not the original 2^{64}, greatly reducing the amount of resources needed for a problem of this size.

Executions for scenarios of around 10.000 or more nodes were automatically canceled by the system for resource reasons.



Graph 15

Graph 15: Time vs. Nodes m=3.

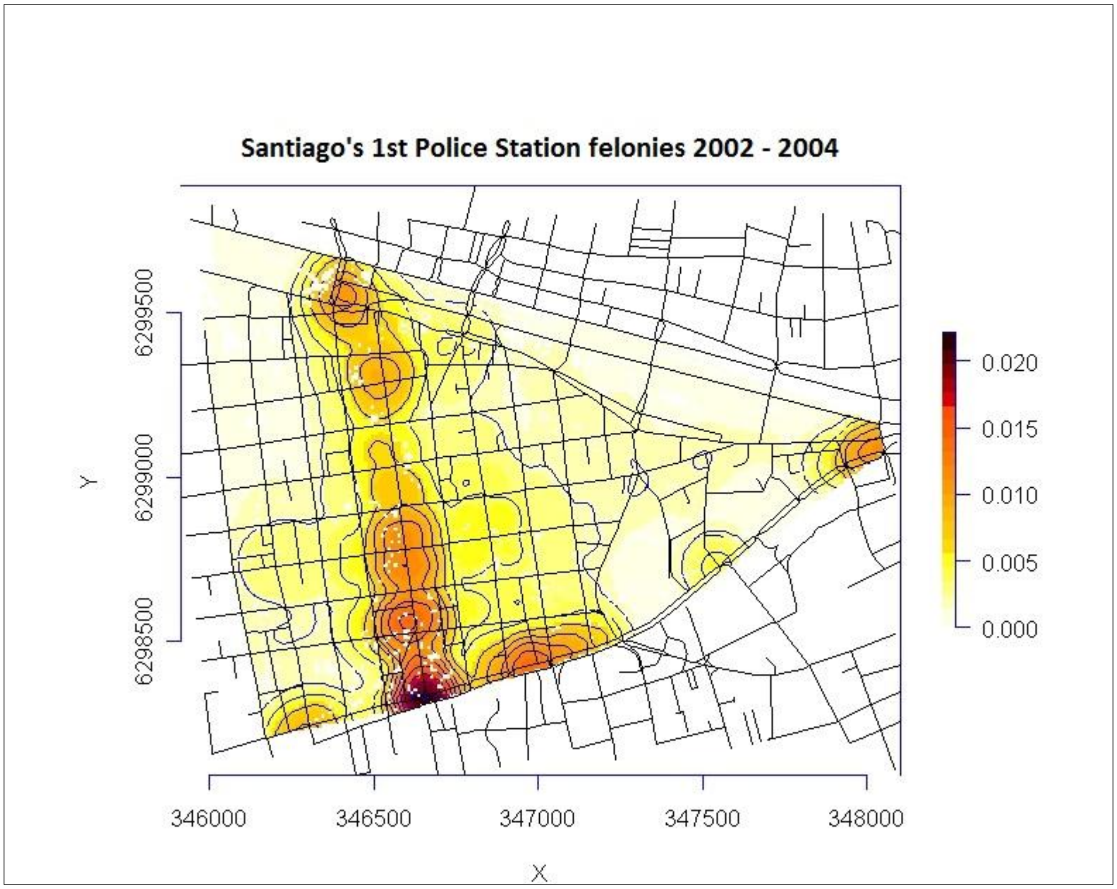


Graph 16

Graph 16: Time vs. Nodes m=5.

6.2 Real World-sized problem

As we previously denoted, we run our algorithm with real world data on Santiago, Chile downtown considering 119 nodes, 195 arcs and the amount of money stolen between 2002 and 2004 on each corner of the considered neighborhood. Scenarios with zero-sum instances with the utilities schemes as displayed on tables 6 and 7 where the considered ones. We tried scenarios with 1, 2 and 3 resources. Initial Data represents a crime scenario displayed by the following graph.

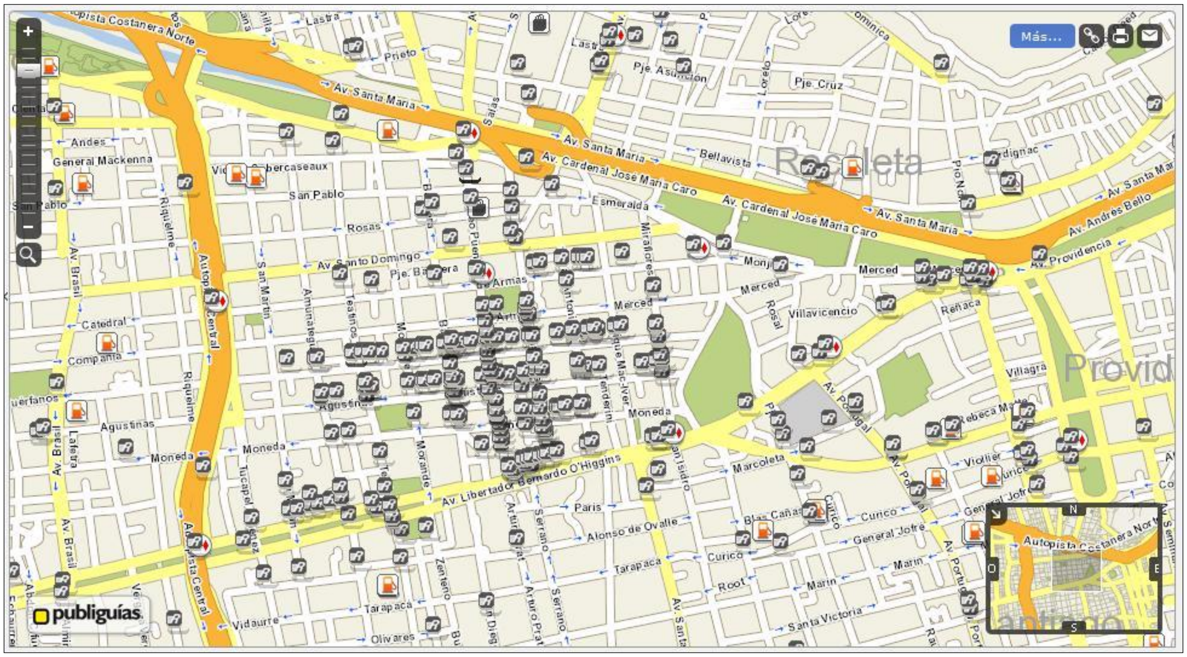


Graph 17

Graph 17: Visual representation of Santiago's downtown crime scene. Period: 2002 – 2004

Colored area indicate the frequency of attacks, with dark red representing more frequently attacked areas and white the least attacked ones.

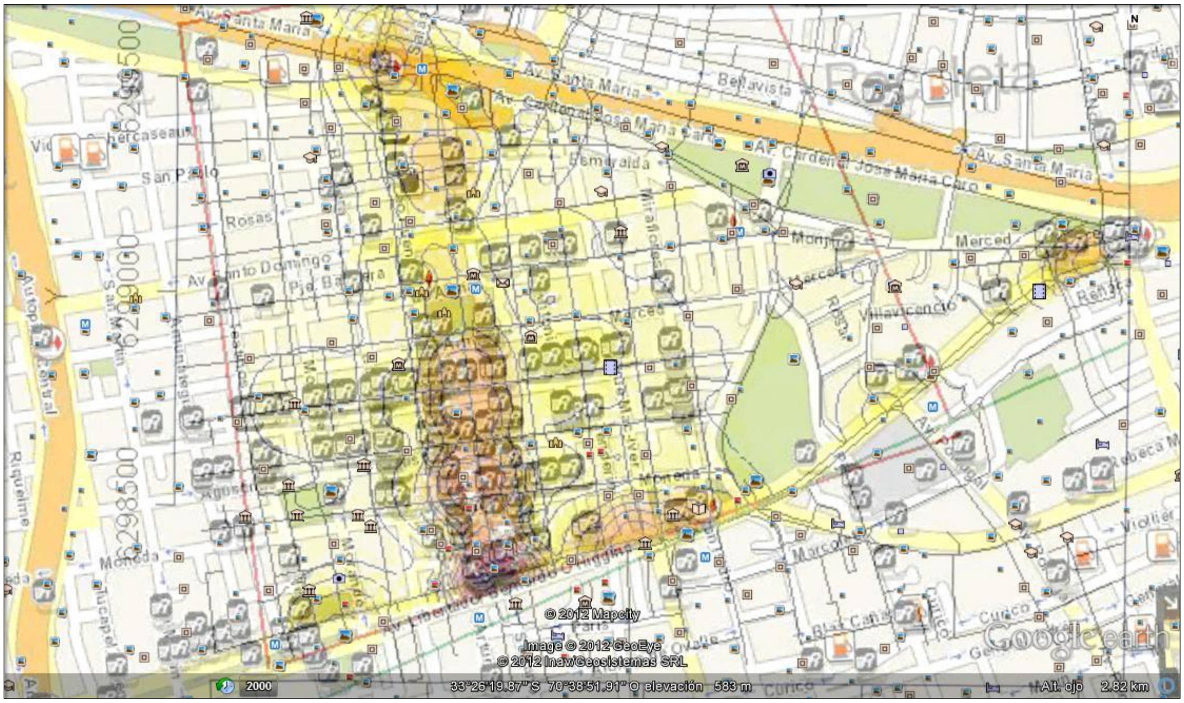
It’s interesting to note that frequency of attacks is highly correlated with the existence of ATMs (Automatic Teller Machines) nearby. The following graph illustrates this by displaying the distribution of ATMs on our studied region.



Graph 18

Graph 18: ATMs locations on Santiago’s downtown. Source: www.publiguias.cl

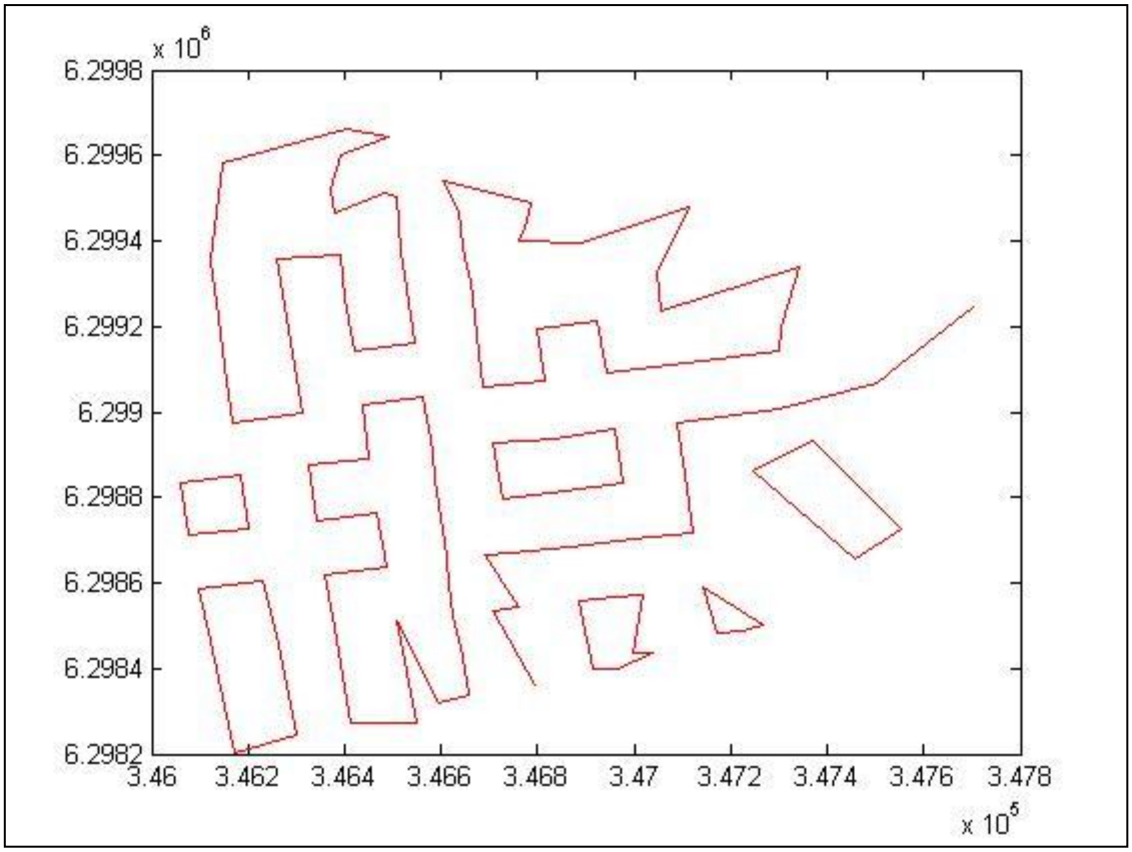
It can be noted in graph 19 by superposing graphs 17 and 18 a sense of how crime is distributed in relation to the ATM density. Previous work has further studied several ways on constructing payoffs and modeling expected crime based on attributes from the environment (ATMs, Drugstores, Government dependencies, etc) [21]. Our contribution is mainly algorithmic and this kind of properties analysis is focused on the implementation and calibration of the model. For this, we didn’t consider ATMs distribution at any point of our studies and only use it to grasp a sense of social behavior.



Graph 19

Graph 19: Overlap of ATMs distribution and crime frequency on Santiago's downtown.

As a preliminary analysis of the situation, we ran an instance of our algorithm with no defending resources, this is m=0.



Graph 20

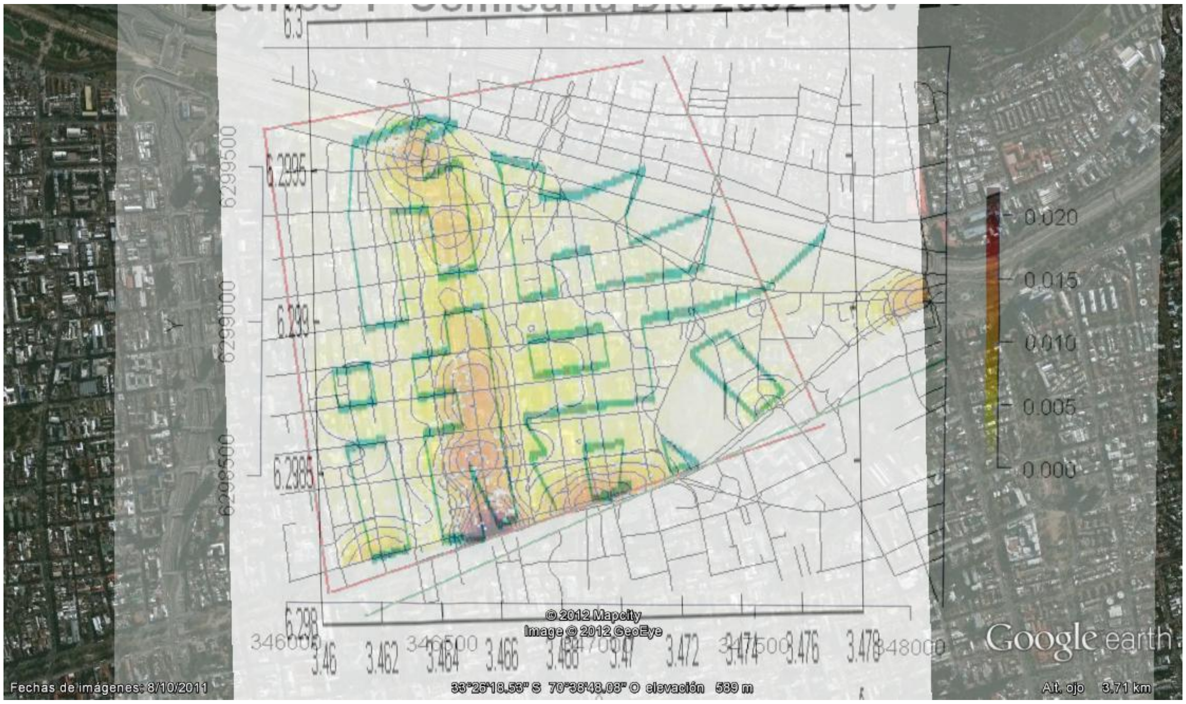
Graph 20: Route delivered by EASSGAR in non-zero-sum and zero-sum instances, m=0.

As can be appreciated, there is more than one non connected route on graph 20. Nevertheless, in our modeling, this is considered as a single route. This situation becomes triggered when we defined the attacker's lineal problem (slave) feasibility constrains (equations 40 – 45). As outputs evidence, a single route complies with the "origin to destination" condition of the LP, we name this as the main route, plus circular routes of nodes not used in this main origin to destination route. These circular routes correspond to positive utility cycles for the attacker on certain regions of the map. This is, a sequence of consecutive nodes not involved in the main route that generates a positive amount of utility to the attacker.

Graph 20, above displays a total of 10 routes, this suggest that 10 different areas concentrate crime and work as independent group of individuals. We think that this situation provides a closer to reality approach, in the sense that crime is not a single force but an aggregate of the actions of several individuals that can work independently.

As we assign more resources to protect the city, is natural for attackers to benefit less from their crime, hence, obtaining a lower utility. If we assign one resource to each node, the attacker would perceive the worst case scenario where each node returns the lowest utility possible evidently making the city a lot less attractive for the attacker to attack. As more police forces are present in the city, there will be less cycles that provide a positive utility for the attacker.

Comparing the output of the algorithm with the current crime scene at Santiago's downtown we can note that routes returned as an output of EASSGAR are consistent with more heavily mugged regions in the map. With this we can have some sort of confirmation about an actual relation between the utilities for the attacker and the data we chose for that same purpose in our model, see graph 21 below.

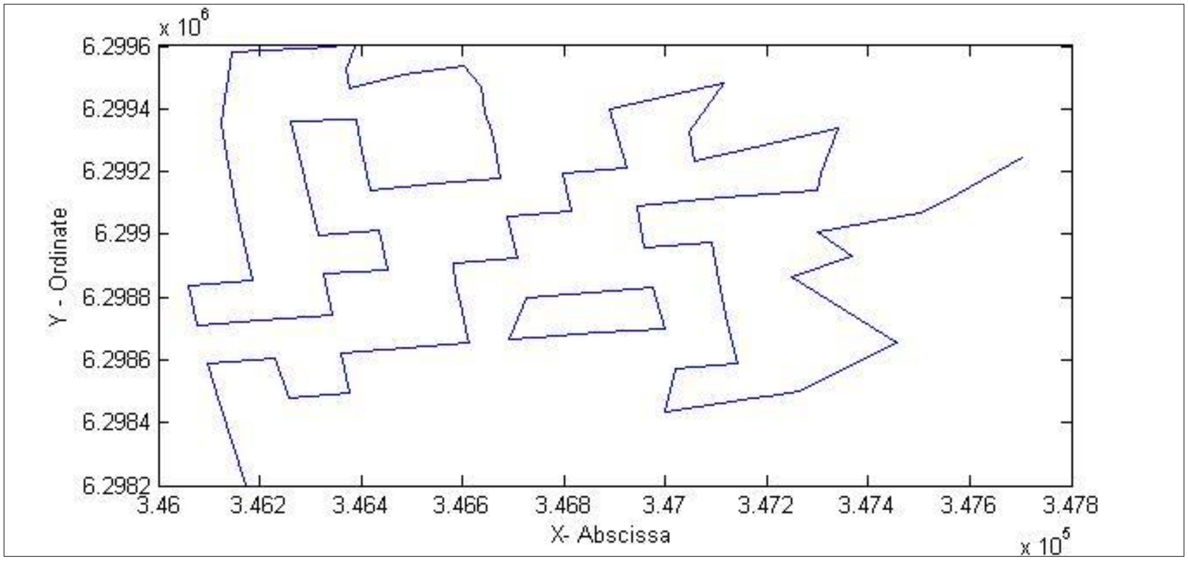


Graph 21

Graph 21: Overlap between Santiago's downtown crime scene vs. EASSGAR output at zero-sum instance, m=0.

6.2.1 EASSGAR Santiago's Downtown results, m=1, 2 and 3.

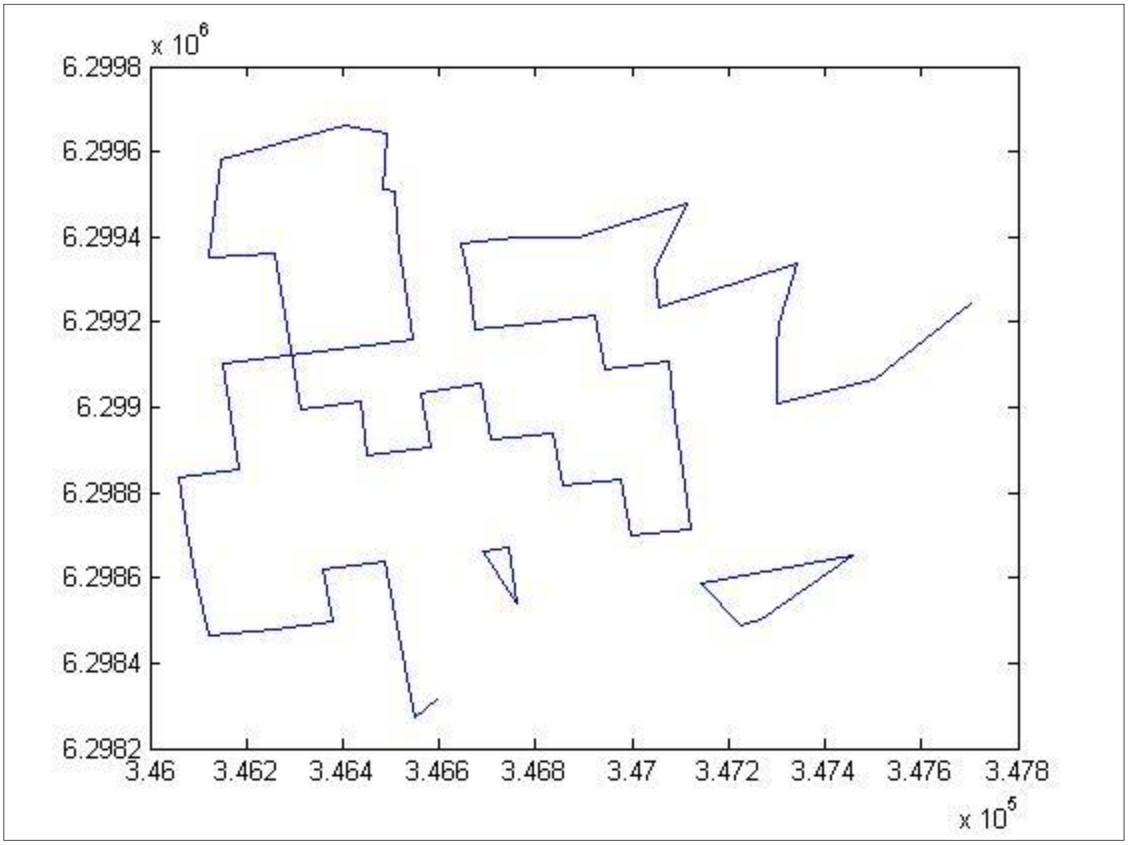
Results obtained as executions of the instance presented in this section are presented on graphs below. Each graph displays an output of the zero-sum scenario implementation of the algorithm. As we recently denoted, we will be considering instances with 1, 2 and 3 police resources each time.



Graph 22

Graph 22: Route delivered by EASSGAR in zero-sum instance, m=1.

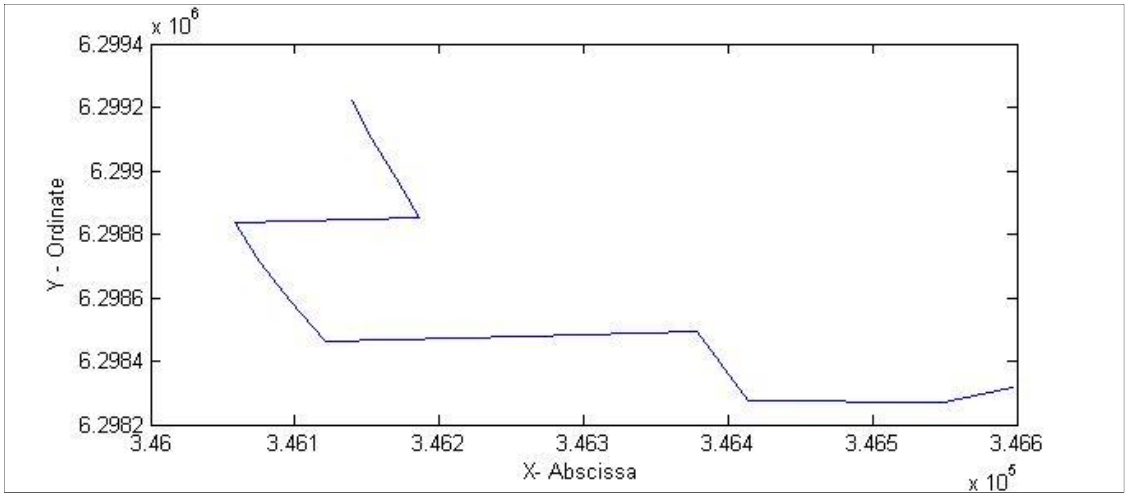
The returned solution for the m=1 instance consisted on 1 main route and 2 circular ones. The amount of iterations needed was 238, total user elapsed time was 4198.19 seconds and system elapsed time was 24.56 seconds; utility for the defender and attacker was -4455000 and 4455000 respectively.



Graph 23

Graph 23: Route delivered by EASSGAR in zero-sum instance, m=2.

Solution for the m=2 instance consisted on 1 main route and 2 circular. Amount of iterations needed was 515, total user elapsed time was 27751.7 seconds and system elapsed time was 90.6352 seconds; utility for the defender and attacker was -1,529,080 and +1,529,080 respectively.

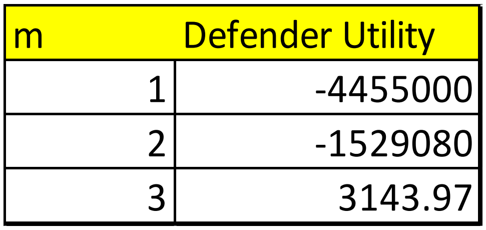


Graph 24

Graph 24: Route delivered by EASSGAR in zero-sum instance, m=3.

Solution for m=3 instance consisted on 1 main route and no circular ones. Amount of iterations needed was 915, total user elapsed time was 41056.9 seconds and system elapsed time was 363.232 seconds; utility for the defender and attacker was 3143.97 and -3143.97 respectively.

Table 6, below, summarizes total utility for the defender (the police) for each case. As can be appreciated, more resources increase utilities for the defender.

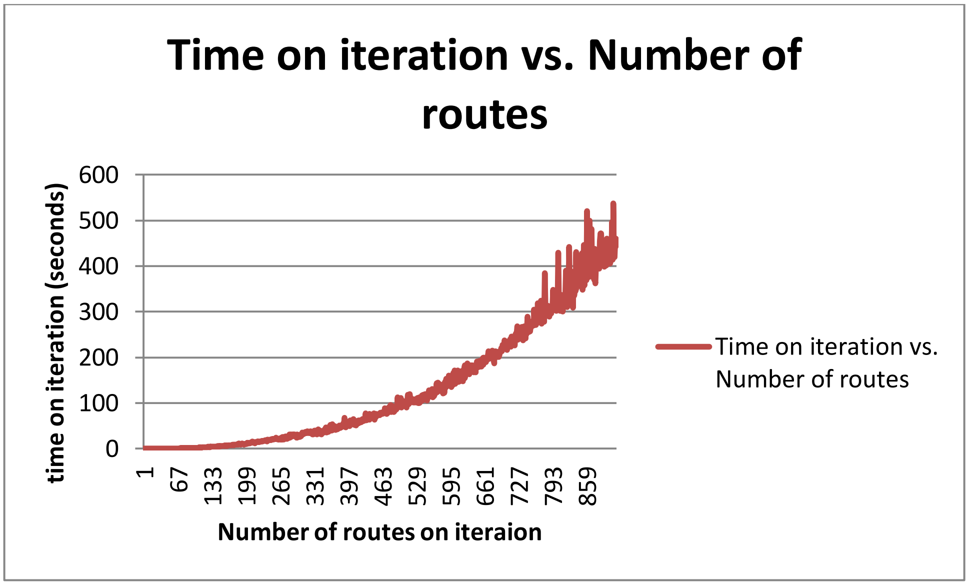


m	Defender Utility
1	-4455000
2	-1529080
3	3143.97

Table 6

Table 6: Defender´s final utility, m=1, 2 and 3.

Graph 25, below, indicates amount of time required to process each iteration depending on the number of routes present at that time.



Graph 25

Graph 25: Time to process iterations depending on the number of routes.

Marginal time on processing each iteration have polynomial dependency number of routes. With this, is clear that trying to solve the problem for all possible routes with no intelligent approach would take a huge amount of time and, probably, crush before it could finish.

6.3 Comparing Results

We generated an instance of our problem on a 8x8 grid (64 nodes). Utilities were randomly generated considering zero-sum conditions and a total of 3 security resources were made available. Each route is defined as sequences of consecutive nodes were attackers can only move right or up. More specifically, attackers begin on node (1, 1) and finish on node (8, 8) where only increments of 1 on a single coordinate are allowed for subsequent nodes. The amount of routes with this characteristic is:

$$\binom{14}{7} = 3432\ Routes$$

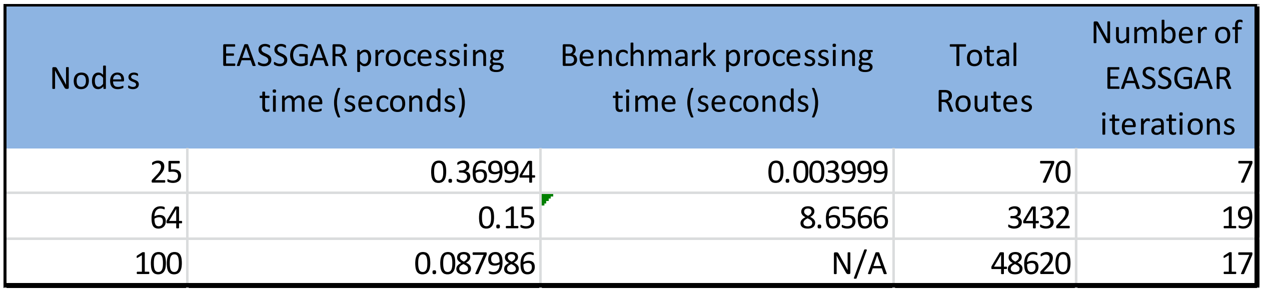
This instance was solved with two different methods; EASSGAR and via standard linear programming on a problem with 3432 restrictions. The complete problem with the whole set of restrictions is equivalent to solving the problem with the ERASER-C algorithm presented on chapter 5 on this same document.

The complete problem took 8.6566 seconds of system processing time versus 0.15 seconds registered by EASSGAR on the same instance. As expected, solutions were the same on both methods (6.14607 and -6.14607 for the attacker and defender respectively). Also, EASSGAR needed 19 iterations before finding the optimal solution; we solved a problem of 3432 restrictions but instead, we never needed the computing resources for solving a problem more complex than 19 restrictions. This yields that both time and computing resources required can be reduced when using the EASSGAR algorithm on problems of a bigger scale.

It fits to note that we tried the same example for an instance of 100 nodes. EASSGAR took 0.036994 seconds; in contrast, the complete problem was kept running for more than one week looking for the solution when this document was finished but the process has not delivered a solution yet.

Another instance with 25 nodes was also tried but in this time processing time was longer for EASSGAR rather than solving the complete problem. In this case the complete problem consisted on 70 routes taking 0.003999 seconds to reach the solution where EASSGAR made 7 iterations reaching the same solution but in 0.036994 seconds.

Table 7 below summarizes presented results:



Nodes	EASSGAR processing time (seconds)	Benchmark processing time (seconds)	Total Routes	Number of EASSGAR iterations
25	0.36994	0.003999	70	7
64	0.15	8.6566	3432	19
100	0.087986	N/A	48620	17

Table 7

Table 7: Performing results for EASSGAR and the benchmark.

6.4 Hardware Details

Our tests and executions for EASSGAR algorithm were programmed in AMPL using the CPLEX solver. The full license for the AMPL software was located on University of Southern California USA, Los Angeles, California. The working environment consisted of two Dell Workstations where we could remotely log into: feet.usc.edu and pies.usc.edu. Logging into either gave us access to our account and the optimization software. The description of each machine was as follows:

Hardware	Dell Workstation
CPU	Dual Intel(R) Xeon(TM) CPU 3.20GHz
Memory	2 GB RDRAM
Operating System	Red Hat Enterprise Linux 3

7 Conclusion and Future Work

Optimally scheduling defender resources in a network-based environment is an important and challenging problem. Especially in times where security in urban road networks, computer networks, and other transportation networks is of growing concern, requiring the development of novel scalable approaches. Domains similar in size as the one we presented in this work have extremely large strategy spaces; a graph with just 50 nodes and 6 resources can have millions of pure strategies for both players. As a result of this work, we successfully developed an algorithm (EASSGAR) which proposes an efficient approach at solving large scale security problems. Our work started characterizing structural properties of the domain in order to identify scenarios with exact results. Further study of the structural properties for utility functions could provide better criteria on the iteration process and, therefore, have a better margin of error from the optimal solution than our algorithm and lower processing times. We also believe that modeling the attackers differently and considering time windows in the optimization could further improve the similarities between the model and reality in this police/terrorist context. This form of approach can represent that people move differently depending the time of day and the day of week, which considers that interests and people have different behavior at different times of the day.

Further research should consider the points of view just mentioned when analyzing optimality conditions. Also studies of sensibility of solution when payoffs change, this could give the algorithm a stronger theoretic backup and, consequently find more properties to support the algorithm.

Our studies successfully solved scenarios for up to 2500 nodes. We noted an increase in processing times as the number of police resources and nodes increases; this is expected because the feasible strategy space for the attacker is directly related to number of resources, with this, more routes are necessary to be considered before stopping conditions are met.

EASSGAR can greatly reduce amount of resources needed for solving problems with a lot of restrictions. It easily reduced problems to 10% of their size but in some cases EASSGAR takes longer to compute the solution than the benchmark. Future studies are required in order to exploit these properties and keep on developing method that allow us to improve the use of computational resources and be able to determine in which cases is more efficient to use special solving methods or when to solve the standard problem.

Finally, algorithm calibration is one of the bigger issues when implementing the algorithm on real life instances. Wrong valorizations in the form of inaccurate utilities fail to represent the way attackers evaluate their attacking routes, thus, providing misleading solutions. This way found solutions probably correspond to an inefficient allocation strategy. Additional analysis of data could be realized by considering qualitative attributes with tools like data mining or logistic regressions.

Further studies could aim to find SSE’s in any kind of data structure and not only on zero-sum situations. There are no reasons to believe that terrorists and police forces have exactly opposite motivations for crime. Also, considering non zero-sum scenarios could allow us to model crime in a more accurate way.

8 Bibliography

[1] Desaulniers, Desrosiers y Solomon, Column Generation, New York: Springer, 2005.

[2] T. von Stengel, Game Theory, 2001.

[3] Osborne, «An introduction to Game Theory,» *Oxford University Pres USA,* 2003.

[4] G. Leitmann, «On Generalized Stackelberg Strategies,» *Optimization Theory and Applications,* pp. 637-643, 1978.

[5] A. a. H. Breton, Sequential Stackelberg equilibria in two-person games, 1988.

[6] Oldser y Basar, Dynamic noncooperative Game Theory, 1995.

[7] v. S. a. Zamir, «Leadership Commitment to mixed strategies,» *CDAM Research Report LSE-CDAM-2004-01,* 2004.

[8] C. a. Sandholm, «A Technique for Reducing Normal-Form Games to Compute a Nash Equilibrium,» AAMAS, pp. 537-544, 2006.

[9] O. a. Rubinstein, A Course in Game Theory, MIT Press, 1994.

[10] Paruchuri, Pearce, Marecki, Tambe, Ordoñez y Kraus, «Efficient Algorithms to Solve Bayesian Stackelberg Games for Security applications,» *AAAI,* Vols. 1 of %21559-1562,

2008.

[11] Paruchuri, Pearce, Marecki, Tambe, Ordoñez y Kraus, «Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games,» *AAMAS*, Vols. %1 of %2895-902, 2008.

[12] Jain, Tsai, Pita, Kiekintveld, Rathi y Ordoñez, «Software Assistants for Randomized Patrol Planning for the LAX Airport police and the Federal Air Marshals Service,» Vols. %1 of %2267-290, 2010.

[13] Kiekintveld, Jain, Tsai, Pita, Tambe y Ordonez, «Computing Optimal Randomized Resource Allocations for Massive Security Games,» *AAMAS*, pp. 689-696, 2009.

[14] An, Pita, Shieh, Tambe, Kiekintveld y Marecki, «GUARDS and PROTECT: next generation applications of security games,» Vols. %1 of %231-34, 2011.

[15] Jain, Kardes, Kiekintveld, Tambe y Ordoñez, «Security games with arbitrary schedules: A branch and price approach,» 2010.

[16] Pita, Jain, Ordoñez, Portway, Tambe, Western, Paruchuri y Kraus, «Using Game Theory for Los Angeles Airport Security,» 2009.

[17] Basilico, Gatti y Amigoni, «Leader-Follower Strategies for Robotic Patrolling in Environments with Arbitrary Topologies,» 2009.

[18] An, Jain, Tambe y Kiekintveld, «Mixed-Initiative Optimization in Security Games: A Preliminary Report,» 2011.

[19] Jiang y Leyton-Brown, «A Polynomial-Time Algorithm for Action-Graph Games,» 2006.

[20] Koller y Milch, «Multi-Agent Influence Diagrams for Representing and Solving Games,» vol. 45, nº 181-221, 2003.

[21] M. P. J. Alegría, Análisis del fenómeno delictual utilizando un modelo de regresión logística en base a atributos, Santiago, Chile: Universidad de Chile, 2011.

[22] Luce y Raiffa, «Games and Decisions: Introduction and Critical Survey,» 1957.

[23] Fletcher, «Practical methods of optimization,» 1987.

[24] Gajardo, Introducción al Analisis Convexo, Universidad de Chile, CNRS, 2006.

[25] H. Bargera, «http://www.bgu.ac.il/~bargera/tntp/,» 2011. [En línea]. Available: http://www.bgu.ac.il/~bargera/tntp/. [Último acceso: 15 09 2011].

[26] J. Jara, «Modelo de asignación de recursos policiacos en la vía pública,» University of Chile - Dept. of Industrial Engineering, Santiago, 2011.

[27] Jain, Korzhyk, Vanek, Pechoucek, Conitzer y Tambe, «A Double Oracle Algorithm for Zero-sum security games on graphs,» AAMAS, 2011.

9 Annex 1

Coordinates and original data by node regarding Santiago´s downtown crime scene between 2002 and 2004.

Node Number	X	Y	Total Declared	Times Attacked	Average Stolen
1	1644,0	1041,0	$ 1.422.302	10	$ 142.230
2	1114,6	276,8	$ 13.322.385	149	$ 89.412
3	1996,7	871,2	$ 11.437.488	134	$ 85.354
4	1980,0	957,0	$ 2.117.108	23	$ 92.048
5	1284,0	1136,0	$ 1.290.745	17	$ 75.926
6	981,0	231,1	$ 18.128.856	239	$ 75.853
7	1495,0	524,0	$ 10.748.449	121	$ 88.830
8	348,0	1458,2	$ 5.206.209	48	$ 108.463
9	435,0	1438,0	$ 2.747.952	31	$ 88.644
10	1188,8	659,0	$ 6.422.773	26	$ 247.030
11	1242,9	804,8	$ 4.676.387	37	$ 126.389
12	1085,4	385,1	$ 3.554.842	49	$ 72.548
13	1032,0	771,9	$ 3.038.149	34	$ 89.357
14	1048,1	646,5	$ 3.847.691	36	$ 106.880
15	1016,4	906,1	$ 1.281.509	14	$ 91.536
16	1876,5	828,5	$ 12.960.093	144	$ 90.001
17	1399,0	451,0	$ 2.327.934	18	$ 129.330
18	596,0	101,3	$ 8.053.155	114	$ 70.642
19	1167,0	283,0	$ 2.213.830	17	$ 130.225
20	1147,0	924,2	$ 3.372.773	20	$ 168.639
21	726,0	1287,0	$ 4.509.178	50	$ 90.184
22	1056,2	1275,2	$ 1.046.659	12	$ 87.222
23	988,7	1122,0	$ 2.296.466	27	$ 85.054
24	488,0	957,1	$ 6.897.970	80	$ 86.225
25	1240,2	937,9	$ 1.840.500	10	$ 184.050
26	1847,0	905,0	$ 2.642.802	18	$ 146.822
27	1492,8	904,2	$ 2.072.792	12	$ 172.733
28	491,7	67,8	$ 4.244.766	52	$ 81.630
29	905,5	197,2	$ 5.099.086	56	$ 91.055
30	1065,3	511,0	$ 1.868.210	21	$ 88.962
31	734,4	159,2	$ 6.540.703	81	$ 80.749
32	854,8	195,0	$ 18.984.510	211	$ 89.974
33	1247,0	993,0	$ 1.280.500	11	$ 116.409
34	1207,0	298,0	$ 2.214.075	13	$ 170.313
35	378,0	810,1	$ 6.029.764	67	$ 89.996
36	999,0	1031,0	$ 1.984.762	14	$ 141.769
37	1444,6	863,3	$ 3.522.009	27	$ 130.445
38	1311,6	726,3	$ 988.453	10	$ 98.845
39	1987,0	913,0	$ 6.900.691	91	$ 75.832
40	504,9	830,4	$ 14.024.302	119	$ 117.851

41	600,9	132,5	$ 37.139.050	442	$ 84.025
42	1298,8	1210,3	$ 975.207	10	$ 97.521
43	63,5	259,1	$ 1.878.096	17	$ 110.476
44	115,3	0,1	$ 5.565.590	69	$ 80.661
45	918,1	629,1	$ 6.305.751	57	$ 110.627
46	361,0	937,8	$ 6.147.820	63	$ 97.584
47	615,4	976,9	$ 3.078.718	27	$ 114.027
48	739,0	991,1	$ 2.624.589	35	$ 74.988
49	940,6	497,0	$ 3.916.562	41	$ 95.526
50	758,0	867,9	$ 4.502.606	52	$ 86.589
51	779,1	734,7	$ 10.967.576	66	$ 166.175
52	567,7	322,4	$ 23.509.606	269	$ 87.396
53	668,9	594,2	$ 5.891.577	65	$ 90.640
54	449,0	309,0	$ 28.581.065	149	$ 191.819
55	554,1	449,8	$ 16.998.535	170	$ 99.991
56	650,7	330,8	$ 4.866.082	39	$ 124.771
57	828,0	354,0	$ 6.665.838	56	$ 119.033
58	797,0	611,8	$ 8.572.416	73	$ 117.430
59	537,9	577,4	$ 22.443.305	240	$ 93.514
60	523,0	703,1	$ 13.115.621	151	$ 86.858
61	409,8	559,8	$ 28.272.543	88	$ 321.279
62	686,9	467,4	$ 4.824.893	51	$ 94.606
63	812,3	481,0	$ 5.289.756	48	$ 110.203
64	431,0	434,0	$ 5.606.353	53	$ 105.780
65	284,8	541,6	$ 4.985.257	41	$ 121.592
66	394,0	684,0	$ 4.560.362	41	$ 111.228
67	356,0	69,2	$ 5.704.428	64	$ 89.132
68	320,0	291,9	$ 1.236.073	15	$ 82.405
69	633,0	460,0	$ 1.758.002	15	$ 117.200
70	267,1	671,0	$ 2.486.791	28	$ 88.814
71	941,4	230,1	$ 1.778.258	24	$ 74.094
72	539,0	113,0	$ 1.979.009	30	$ 65.967
73	704,0	338,5	$ 5.830.204	51	$ 114.318
74	961,0	369,0	$ 3.226.677	43	$ 75.039
75	651,1	722,0	$ 5.152.315	57	$ 90.391
76	630,0	852,8	$ 2.803.753	27	$ 103.843
77	556,5	395,2	$ 2.490.742	17	$ 146.514
78	548,0	495,1	$ 1.103.925	12	$ 91.994
79	900,6	755,0	$ 5.834.678	45	$ 129.660
80	904,0	362,0	$ 1.984.349	13	$ 152.642
81	302,3	416,8	$ 2.625.539	27	$ 97.242
82	528,9	638,8	$ 910.175	11	$ 82.743
83	886,2	885,2	$ 1.668.725	15	$ 111.248
84	244,0	39,0	$ 14.375.211	177	$ 81.216
85	864,6	1009,8	$ 898.745	12	$ 74.895
86	586,0	221,1	$ 1.992.715	16	$ 124.545
87	257,2	792,9	$ 2.160.698	11	$ 196.427
88	201,5	275,0	$ 2.727.167	35	$ 77.919
89	704,9	1194,4	$ 719.669	12	$ 59.972

90	332,0	1396,0	$ 10.961.943	124	$ 88.403
91	601,7	1100,2	$ 1.425.026	26	$ 54.809
92	713,9	1229,0	$ 1.129.975	18	$ 62.776
93	717,8	1139,2	$ 3.594.930	41	$ 87.681
94	472,1	1084,6	$ 14.916.511	185	$ 80.630
95	321,0	1260,0	$ 3.872.171	35	$ 110.633
96	344,0	1062,0	$ 4.661.375	56	$ 83.239
97	460,1	1172,6	$ 6.225.233	83	$ 75.003
98	112,0	772,0	$ 1.536.870	19	$ 80.888
99	585,8	1181,8	$ 2.767.116	31	$ 89.262
100	312,6	1320,0	$ 13.469.926	157	$ 85.796
101	596,1	1137,1	$ 2.171.567	16	$ 135.723
102	544,0	1335,0	$ 2.203.369	33	$ 66.769
103	450,2	1301,0	$ 7.734.142	82	$ 94.319
104	578,0	1265,0	$ 3.891.144	33	$ 117.913
105	81,1	1017,5	$ 5.689.523	14	$ 406.395
106	426,8	1306,0	$ 3.366.018	35	$ 96.172
107	144,1	523,7	$ 6.068.715	55	$ 110.340
108	127,9	649,5	$ 2.737.951	28	$ 97.784
109	332,0	1164,0	$ 4.920.395	65	$ 75.698
110	39,0	383,0	$ 920.739	11	$ 83.704
111	170,9	400,5	$ 3.563.643	27	$ 131.987
112	203,2	1155,2	$ 1.255.284	11	$ 114.117
113	88,1	1379,0	$ 2.062.782	18	$ 114.599
114	65,3	1150,0	$ 713.120	10	$ 71.312
115	671,9	1137,1	$ 933.030	10	$ 93.303
116	831,1	1193,0	$ 949.241	11	$ 86.295
117	0,9	631,4	$ 1.056.467	15	$ 70.431
118	17,9	506,3	$ 2.145.008	29	$ 73.966
119	94,9	899,4	$ 1.320.036	14	$ 94.288