UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FíSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

ADDRESSING PROBLEM SIZE IN STACKELBERG SECURITY GAMES

TESIS PARA OPTAR AL GRADO DE
DOCTOR EN SISTEMAS DE INGENIERÍA

VÍCTOR DANIEL SIMÓN BUCAREY LÓPEZ

PROFESOR GUÍA:
FERNANDO ORDOÑEZ PIZARRO

MIEMBROS DE LA COMISIÓN:
JOSÉ CORREA HAUESSLER
EUGENIO DELLA VECCHIA
ALAIN JEAN-MARIE
MILIND TAMBE

Este trabajo ha sido parcialmente financiado por CONICYT

SANTIAGO DE CHILE
2017

ABSTRACT
AUTHOR: VÍCTOR DANIEL SIMÓN BUCAREY LÓPEZ
DATE: 2017
ADVISOR: FERNANDO ORDOÑEZ PIZARRO

ADDRESSING PROBLEM SIZE IN STACKELBERG SECURITY GAMES

In this thesis we present algorithmic and modeling contributions for Stackelberg games that help address challenges due to problem size. Stackelberg games consider that one player, called leader, commits a strategy first and the other player, named follower, observes this strategy and plays a best response.

In this thesis we address the problem size that arises due to large leader action space, or because the interaction between the leader and the follower is evolving over time. In the last case, we model this interaction as a stochastic game.

In Chapter 1, we present a situation where a defender, the leader in the Stackelberg game, has to pair up resources to do patrol labor. In this case the set of pure strategies is exponentially large. We show a mixed integer programming formulation with polynomial number of variables and an exponentially sized set of constraints that can be separated in polynomial time. Moreover, we design sampling methods to retrieve implementable strategies for the defender. We show a case study in border patrolling labor of Carabineros de Chile. We show that our model with the cut-generation scheme outperforms other models, scaling up to large size instances.

In Chapter 2 we show a method to scale up an algorithm to compute optimal strategies in the context of opportunistic crime modelling, via a multi-layer clustering. Inside each cluster the algorithm can scale up in reasonable times. This methodology can be adapted to any problem where geographical aspects are important and the number of targets is large.

In Chapter 3 we face the problem of computing stationary policies that form a strong Stackelberg equilibrium in stochastic games. We find a family of instances where both, Value Iteration and Policy iteration converge to a strong Stackelberg equilibrium. We show via a counterexample that this is not always possible. Also, we show computationally that Value Iteration applied to security games instances seem to always converge to a unique strong Stackelberg equilibrium. Finally, we study mathematical programming formulations to compute a Stackelberg equilibrium in stochastic games.

Finally, in Chapter 4 we study a dynamic game, where a central agency aim to avoid the overexploitation of water, controlling the marginal cost of water extraction in agriculture. We model this situation as a stochastic game where the leader is the central agency and the followers are farmers who seek to maximize an instantaneous reward function at each period. In our setting, the leader maximizes the total discounted reward of the whole set of farmers. We find that we can achieve better levels of water in the steady-state by controlling the marginal cost. Finally, we propose a robust optimization approach to include uncertainty in our model.

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE DOCTOR EN SISTEMAS DE INGENIERÍA
POR: VÍCTOR DANIEL SIMÓN BUCAREY LÓPEZ
FECHA: 2017
PROF. GUÍA: FERNANDO ORDOÑEZ PIZARRO

ADDRESSING PROBLEM SIZE IN STACKELBERG SECURITY GAMES

En esta tesis presentamos contribuciones algorítmicas y de modelación para Juegos de Stackelberg para problemas de gran tamaño. Juegos de Stackelberg es un tipo de juego en el que un jugador, llamado líder, utiliza una estrategia y luego el otro jugador, llamado seguidor, observa esta estrategia y juega una mejor respuesta. En el contexto de seguridad, el líder se le denomina defensor, y al seguidor se le denomina atacante.

En el Capítulo 1, presentamos una situación en donde el defensor tiene que emparejar recursos para poder desarrollar labores de patrullaje. En este caso, el conjunto de estrategias puras es de tamaño exponencial. Presentamos una formulación en programación lineal entera mixta con un número polinomial de variables y un número exponencial de restricciones que pueden ser separadas en tiempo polinomial. Además, se diseñó un método de sampleo para recuperar estrategias implementables para el defensor. Por otra parte, mostramos un caso de estudio basado en el patrullaje fronterizo de Carabineros de Chile. Finalmente mostramos que nuestro modelo con el generación de cortes funciona considerablemente mejor que cualquier modelo en la literatura.

En el Capítulo 2, se muestra una manera de escalar un algoritmo basado en Aprendizaje de Máquinas, realizando multiples capas de clusters de territorios en donde en cada cluster el algoritmo resuelve en tiempos razonables.

En el Capítulo 3 se realiza un estudio sobre existencia de estrategias estacionarias que formen equilibrios de estrategias fuertes. Se detecta una familia de instancias en las cuales algoritmos de programación dinámica pueden encuentran el único equilibrio de Stackelberg fuerte. En este caso, se demuestra que Iteración de políticas y de valores convergen a los únicos valores de equilibrio. Se muestra vía contraejemplo que no siempre programación dinámica puede encontrar equilibrios de Stackelberg. Computacionalmente se encuentra que la estructura de juegos de seguridad presentara características que hace aplicable la teoría de operadores para encontrar equilibrios de Stackelberg. Finalmente se estudian formulaciones basadas en programación matemática para calcular equilibrios de Stackelberg.

Finalmente, en el Capítulo 4 mostramos un juego dinámico, en donde un agente central tiene como objetivo la sobreexplotación de agua en el contexto agrícola, controlando los costos marginales de extracción que enfrentan los agricultores. Modelamos esta situación como un juego estocástico en donde se busca un equilibrio de Stackelberg en donde el líder es la agencia central, y los seguidores son los agricultores. Computacionalmente se encuentra que se pueden alcanzar mejores niveles de agua en el estado estacionario controlando los costos marginales a través de las tasas de descuento del líder, aunque los agricultures sean miopes. Finalmente, nosotros proponemos una formulación de programación robusta para incluir incertidumbre en nuestros modelos.

Dedicado a la memoria de
Adela Godoy Inostroza.
Te extraño.

Acknowledgements

In the first place, I would like to thank my family. To my mother, for being the main source of love and support throughout all these years. Undoubtedly, it was you who taught me to fight for my dreams. To embrace love. To share love. I am so proud of you.

To Juan, who has always supported my crazy adventures. For being such a great companion.

To Daniela, for her love and affection throughout these years, and sharing with me, this beautiful adventure called parenthood. To my little son, Pablo, for being my main and most important motivation in this life.

While doing this Ph.D., I was able to forge and strengthen bonds of friendship, that have been nourishing in every single way. To my dear namesake Víctor Verdugo, for his unconditional support, for being a true friend, for being a counselor and a mentor at times I needed him; To my dear friend Dana, for being a great mate, for her affection and of course, for giving me alfajores; To Angie who came to refresh this last period with her joy and laughter; To Felipe Carrasco for allowing me to be his friend, and for all those pleasant breaks we took; To Carlos Casorrán, for being such a great friend and academic partner; To Andrea M. for all her support in Santiago, Rosario and Los Angeles. I would also like to thank my friends Gaby, Felipe, Rayen and Cristóbal, who supported and helped me in times of need. I feel I became a better person thanks to all of you

I want to thank all the researchers that trained and educated me during these years doing the Ph.D. To Eugenio, for his patience as a teacher, for his advice and tips. It's a real privilege to work with such a valuable and beautiful person like you. To Alain and Mabel for receiving and treating me as another colleague, for their tours and fun talks. When I grow up, I want to be like you! To Milind for letting me be a part of his researching group for a couple of months. How amazing is to share with people of world's first level. I learnt so much from you and your people.

I also want to thank several people I had to discuss and work with during this period. To Chao Zhang, Ayan Mukhopadhyay, Arunesh Sinha, Yundi Qian, Yevgeniy Vorobeychik, Hugo Navarrete, Karla Rosas, Oscar Figueroa, Andreas Wiese, Roberto Cominetti, José Correa, Mario Bravo, Renaud Chicoisne, Renny Márquez, Andrea Canales, Álvaro Brunel, Javier Ledezma, Ricardo de La Paz, Eduardo Lara, Eduardo Zuñiga, Verónica Diaz, Cristiam Gil, Sebastián Davila, Richard Weber. Thank you, for your ideas and contributions.

Thanks to the Industrial Engineering Department of the Universidad de Chile. From the coffee shop people, who always offered me a nice chat about football and music, to the teachers, students and workers, all of you have made of this spot a great place to be. Especially to Paulina and Evelyn for all the interesting scientific outreach project that I have been involved in Comunidad Ingenio, and for all the interesting conversations that we had in this 7 last years.

Finally, I would like to thank Fernando, for being an excellent advisor, an amazing friend and a true mentor. Thank you so much for the advices, the lessons, the patience and for cheering me up during the hard times. Even though your tasks as the Director of the Department consumed lots of your time, you were always willing to help me.

This thesis is dedicated to my grandmother Adela, who was always next to me, spiritually. I know it. I really wish you were here.

Agradecimientos

En primer lugar quiero agradecer a mi familia. A mi mamá por ser la fuente principal de amor y apoyo durante estos años. Sin duda, de ti aprendí a luchar por lo que uno quiere, a entregar amor. Estoy muy orgulloso de ti. A Juan por apoyarme en todas mis locuras y ser una gran compañía. A Daniela, por el cariño de todos estos años, y por compartir conmigo esta hermosa aventura de ser papás. A mi hijo Pablo, porque es el principal motor de mi vida.

Esta etapa de doctorado hizo forjar y reforzar amistades que han sido muy nutritivas en todo sentido. A mi tocayo Víctor Verdugo por siempre estar ahí, por ser un verdadero amigo, consejero y guía en muchas ocasiones; a Dana por ser una gran compañera y por entregarme mucho cariño y alfajores; a María Angélica que vino a refrescar esta última etapa con su alegría y risa; a Felipe Carrasco por permitirme ser su amigo y los gratos breaks que nos tomamos; a Carlos Casorrán por ser un gran amigo y hermano académico; a Andrea M. por todo su apoyo y cariño en Los Angeles, Santiago y Rosario. También quiero agradecer a mis amigos Gaby, Cristobal y Felipe, que me apoyaron y entregaron contención en momentos importantes. Siento que soy una mejor persona gracias a ustedes.

También quiero agradecer a los investigadores que en estos años de doctorado me formaron. A Eugenio, por la paciencia al enseñar y sus consejos, es un lujo trabajar con gente tan valiosa y linda como tú. A Alain y Mabel por recibirme como un colega más, por sus paseos y entretenidas charlas. Cuando grande quiero ser como ustedes. A Milind por dejarme ser parte de su grupo de investigación un par de meses. Es genial compartir con gente de primer nivel mundial. Aprendí mucho contigo y con tu gente.

Quiero agradecer también a distintas personas con las que me tocó discutir y trabajar en este periodo. A Chao Zhang, Ayan Mukhopadhyay, Arunesh Sinha, Yundi Qian, Yevgeniy Vorobeychik, Hugo Navarrete, Karla Rosas, Oscar Figueroa, Andreas Wiese, Roberto Cominetti, José Correa, Mario Bravo, Renaud Chicoisne, Renny Márquez, Andrea Canales, Álvaro Brunel, Javier Ledezma, Ricardo de La Paz, Eduardo Lara, Eduardo Zuñiga, Verónica Diaz, Cristiam Gil, Sebastián Davila, Richard Weber. Gracias por sus ideas y sus aportes.

Muchas gracias al Departamento de Ingeniería Industrial de la Universidad de Chile. Desde la gente de la cafetería que siempre me ofreció una grata discusión de fútbol y música, hasta los profesores, alumnos y funcionarios con los que hacen que este lugar sea un excelente lugar. También no puedo dejar de agradecer a Comunidad Ingenio con quienes he participado de proyectos apasionantes como lo es la divulgación científica. En particular agradecer a Paulina y Evelyn por sus gratas e interesantes conversaciones.

Finalmente quiero agradecer a Fernando, por ser un excelente guía, un gran amigo y un verdadero mentor. Muchas gracias por tus consejos, enseñanzas, paciencia y subirme el ánimo en los momentos más duros. A pesar de que tus labores de director de departamento no te dejaban mucho tiempo, tuviste siempre la disposición de ayudarme.

Esta tesis va dedicada a mi abuela Adela, que sé que siempre estuvo conmigo en espíritu. Como me encantaría que estuvieras acá.

Table of Contents

List of Tables

List of Figures

Introduction

Many real life situations involve the interaction between the decisions of different agents. Game Theory is a way of modeling situations with multiple self-interested decision makers and in the last 90 years it has become a prolific research field with applications in economy, biology, public policy among others. Stackelberg games [60] are a particular class of games where a leader player moves first and then the follower observes the strategy committed by the leader and decides their own actions, both maximizing their own utility.

Stackelberg games are widely used in security applications [57]. In these applications the leader takes the role of the defender, e. g. the security forces, and the follower takes the role of an attacker, e. g. a terrorist adversary. This way of modeling interactions in security takes into account the ability of an attacker to gather information about the defense strategy before planning an attack while the defender always has to implement a strategy [44]. Many real-world applications were developed following this methodology. Namely, ARMOR in LAX, the airport of Los Angeles, California, [48]; IRIS, a software scheduling assistant for the Federal Air Marshals (FAMs) [59]; or TRUST, a system to schedule randomized patrols for fare inspection in transit Systems [71].

There are many extensions of this basic model. One of them arises when there is uncertainty in the type of follower (attacker) that the leader (defender) has to face, or equivalently the uncertainty about the payoffs in the game, addressing the concept of Bayesian Stackelberg Game [18]. This approach assumes no relationship among the different types of follower. Another approach is to suppose that followers can interact among them and play a Nash equilibrium given a fixed strategy commited by the leader. We use both concepts of multi-follower games along this thesis.

Another way to extend this model is to consider stochastic games. Stochastic games are a generalization of Markov Decision Processes (MDPs) to the case where at each state of the process there are two or more controllers [27]. This concept can be used to extend the security model to cases where the interaction between the defender and the attacker is performed in different steps of times and in different circumstances. For instance, in [71] the authors model the interaction between the defender and the possible attackers as a stochastic game to represent failures in the execution of strategies for the defender. In [69, 70] authors model games where the followers are not completely rational, that is, they do not always play a best response. In this cases bounded rationality is modeled with a probability function response, assigning smaller probability to actions returning lower expected rewards.

Problem size due to large scale is another important feature. For example in the case

where a defender has a set of m resources to cover n targets, the set of strategies for the defender has a size of $\binom{n}{m}$. In [37] this problem is tackled by representing the set of strategies with variables that models the frequency under which each objective is covered, reducing the size of the optimization problem to a one with a polynomial number of variables. Another example of algorithm to aim scalability in large size instances is presented in [32] where a branch and price method is presented, exploiting the special structure of the schedules in the context of FAMS. In this thesis we address problem size due to large scale set of actions for the defender, because either the set of targets is developed in a wide geographical area, or the interaction is evolving through the time.

This thesis is divided in four chapters. In Chapter 1 we develop a Bayesian Stackelberg game where the leader can team up resources to do patrolling work. In this case the strategy space is exponentially large. We develop a compact model with polynomial number of variables and exponential number of constraints and an algorithm to separate this exponentially large set of constraints in polynomial time. This method substantially improves solution running times and the scalability of this application. We also applied this methodology in a real case to protect the borders in the Chilean context. We propose a methodology to estimate the payoff matrix of this game. This chapter is an extended version of the working paper [11].

In Chapter 2 we use a model to do abstractions improving the performance in algorithms to face opportunistic crime. We develop a mixed integer program to do districting in order to create multi-layer geographical clusters. This chapter is part of [73].

In Chapter 3 we face the problem of computing Stackelberg equilibria in stochastic games. To attack the problem, we define a suitable operator and a family of problems where there exists stationary policies forming Stackelberg equilibrium. We also discuss about the applicability of mathematical programming approach and dynamic programming algorithms to find Stackelberg equilibria. We study a stochastic game where a security agent has to patrol a set of locations through the time and an attacker has to decide whether to attack and where. This chapter is an extended version of the working paper [12].

Finally, in Chapter 4, we show a model to apply the methodology of Stackelberg equilibrium in stochastic games in an extraction water game in agriculture. The point is to show the methodology developed in Chapter 3 is useful in domains other than security. This game models a situation when an agency wants to put prices in the extraction water cost and then agents observes this price and plays a Nash equilibrium. The main idea is to achieve a sequence of *good* Nash equilibria to keep robust levels of water and avoid overexploitation of the natural resource. This chapter is an extended version of the working paper [34].

Main Contributions

In Chapter 1 we introduce a new model to combine resources in the context of Stackelberg Security Games. We also develop a compact formulation making use of the matching polyhedron structure. Furthermore, we design an approximate sampling method to retrieve implementable strategies from the optimal solution of our compact formulation. Finally, we propose a new constraint generation algorithm to scale up our formulation.

In Chapter 2 we present a hierarchical algorithm to approximate a large set of targets in a large geographical area by clustering according to capacity constraints. We exhibit some heuristics to solve this problem at scale.

In Chapter 3 we show a family of stochastic games where dynamic programming algorithms find a Strong Stackelberg Equilibrium in stationary policies. We define a Stackelberg operator for this family of instances and for the general case. We use this operator to prove that Value Iteration and Policy Iteration converge for this family of instances. We discuss the applicability of the models and algorithms proposed in security settings and natural resources management (Chapter 4).

Chapter 1

Coordinated defender strategies for border patrols

1.1. Introduction

Security is a worldwide concern that in many situations requires the coordinated use of diverse types of resources. For instance, police patrols a city using different resources: patrol cars, motorbikes, police on horse, on foot, or even using a helicopter. These different types of resources have different capabilities (patrolling range, ability to move through different terrain, ability to interact with the population) and a coordinated use of these specialized resources is used in providing an overall security level. Pooling resources from different units also helps sustain preventive patrolling activity without burdening any single unit.

Securing national borders is a natural concern of a country to defend it from the illegal movement of contraband, drugs and people. The European Union (EU) created the European Border and Coast Guard in October 2016 in response to the recent increase in migrant flows into the EU [19]. In the United States the Department of Homeland Security states as a primary objective that of "protecting [the] borders from the illegal movement of weapons, drugs, contraband, and people, while promoting lawful entry and exit" claiming it is "essential to homeland security, economic prosperity, and national sovereignty" [22].

The task of patrolling the border requires monitoring of vast stretches of land 24/7. A way to help provide this monitoring is to combine resources from different locations in a joint effort to coordinate a global patrol plan. It is crucial to balance the effectiveness of a global border plan with the cost and difficulty of locally coordinating resources in undermanned areas. The European Border and Coast Guard lists as one of its prime objectives "organising joint operations and rapid border interventions to strengthen the capacity of the member states to control the external borders, and to tackle challenges at the external border resulting from illegal immigration or cross-border crime".

In this chapter we consider the problem of patrolling a border in the presence of strategic adversaries that aim to cross the border taking into account the defender patrolling strategies.

We model the situation as a Stackelberg game where the defender acts as the leader executing a preventive border patrol, which is observed prior to the optimal response by the strategic adversary, which acts as the follower. Due to the size of the border patrol problem the defender coordinates local resources to achieve a global defender strategy. Stackelberg Games, introduced by [60], are an example of bilevel optimization programs, [10], where top level decisions are made by a player – the leader – that takes into account an optimal response – of a follower – to a nested optimization problem.

Recent research has used Stackelberg games to model and provide implementable defender strategies in real life security applications. In these applications, a defender aims to defend targets from a strategic adversary by deploying limited resources to protect them. The defender deploys resources to maximize the expected utility, anticipating that the adversary attacks a target that maximizes its own utility. Examples of Stackelberg security games applications include assigning Federal Air Marshals to transatlantic flights, [33], determining U.S. Coast Guard patrols of port infrastructure [55], police patrols to prevent fare evasion in public transport systems [71], as well as protecting endangered wildlife [67].

One of the challenges that has to be addressed in solving these Stackelberg games for real-world security applications is problem size. When the defender action is to allocate limited resources to various targets, the set of possible defender actions can be quite large, i.e the different combinations in which the resources can be assigned to the targets. The size of this set of actions is exponential in the number of resources and targets. In [37] a relaxation of the Stackelberg security game is formulated which determines the frequency with which each target is protected. This polynomial formulation (in the number of targets and security resources) is shown to be exact when there are no constraints on what constitutes a feasible defender action, but it is only an approximation in the general case. Decomposition approaches have been developed for the general case, when feasible defender actions satisfy additional constraints. For instance a column generation approach is introduced in [32] and a cutting plane approach in [68]. A branch and cut approach based on Benders' decomposition is introduced in [72]. A specialized exact decomposition method that exploit specific problem structure was introduced in [31].

In this work we tackle the problem of globally planning pairings between police precincts and given those pairs, locally planning the targets within the corresponding territory that need to be protected. To avoid explicitly enumerating the exponentially many defender pure strategies, we propose a mixed integer formulation for a Stackelberg security game on a network with a polynomial number of variables and exponentially many constraints that can be separated in polynomial time. The network is composed of nodes which represent police precincts and edges that determine whether or not a pairing between two precincts is allowed. Within each precinct, a set of targets needs to be protected but a target within a precinct can only be protected by a joint patrol originating from a precinct pair. The decision variables in our formulation are two coverage distributions, one on the edges of the network and one on the targets that need to be protected. Further, we provide two sampling methods that, given the optimal coverage distribution on edges and targets, recover the defender's implementable strategy, i.e., a valid pairing of precincts, and a set of targets to protect.

In addition, we provide a case study of our proposed methodology in a real border patrol

application we develop in collaboration with the national police force in Chile (Carabineros de Chile). We explain the setting in detail as well as describe a parameter estimation procedure we use to determine payoffs for the players in our game. A sensitivity analysis is conducted to test the validity of our method and thorough computational experiments are conducted to measure the performance of our approach.

The rest of the chapter is as follows. Section 1.2 introduces the notation and problem formulation. In Section 1.3 we present the solution method proposed which is based on a constraint generation approach of an equivalent reformulation of our problem. In this section we also present the sampling methods to retrieve an implementable patrolling strategy from the optimal solution obtained. Section 1.4 is devoted to our case study of border patrol. In section 1.5, we provide computational experiments to measure the performance of our formulation. Finally, we present our conclusion and discuss future work in Section 1.6.

1.2. Problem Formulation and Notation

In this section we first introduce the general framework of Stackelberg games, the notation and a review of benchmark models. Then, we present a Stackelberg game that seeks to select coordinated defender strategies given heterogeneous resources when facing strategic attackers.

1.2.1. Stackelberg Security games

We consider a general Bayesian Stackelberg game, where a leader is facing a set K of followers, as introduced in [46]. In this model the leader knows the probability π_k of facing follower $k \in K$. We denote by I the finite set of pure strategies for the leader and by J be the finite set of pure strategies for each of the followers. A mixed strategy for the leader consists in a vector $\mathbf{x} = (x_i)_{i \in I}$, such that x_i is the probability with which the leader plays pure strategy i. Analogously, a mixed strategy for follower $k \in K$ is a vector $\mathbf{q}^k = (q_j^k)_{j \in J}$ such that q_j^k is the probability with which follower k plays pure strategy j. The payoffs for the agents are represented in the payoff matrices $(R^k, C^k)_{k \in K}$, where $R^k \in \mathbb{R}^{|I| \times |J|}$ gives the leader's reward matrix when facing follower $k \in K$ and $C^k \in \mathbb{R}^{|I| \times |J|}$ is the reward matrix for follower $k \in K$. The R_{ij}^k (C_{ij}^k) entry gives the reward for the leader (follower) of taking the leader action i and the k-th follower action j. With these payoff matrices, given a mixed strategy $\mathbf{x}$ for the leader and strategy $\mathbf{q^k}$ for follower k, the expected utility for follower k is given by $\sum_{i \in I} \sum_{j \in J} C_{ij}^k x_i q_j^k$ while the expected utility for the leader is given by $\sum_{k \in K} \pi_k \sum_{i \in I} \sum_{j \in J} R_{ij}^k x_i q_j^k$.

The objective of the game is for the leader to commit to a payoff-maximizing strategy, anticipating that every follower will best respond by selecting a payoff-maximizing strategy of its own. The solution concept used in these games is the Strong Stackelberg Equilibrium (SSE), introduced in [39]. In an SSE, the leader selects the strategy that maximizes payoff given that every follower selects a best response breaking ties in favor of the leader when the follower is indifferent between several strategies.

The problem of finding the strong Stackelberg equilibrium can be formulated as the following bilevel bilinear problem [46]:

$$\max_{x,q} \quad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} \pi^k R^k_{ij} x_i q^k_j \tag{1.1}$$

$$\text{s.t.} \quad \mathbf{x}^\top \mathbf{1} = 1,\ \mathbf{x} \geq 0, \tag{1.2}$$

$$\mathbf{q}^k \in \arg\max \left\{ \sum_{i\in I}\sum_{j\in J} C^k_{ij} x_i r^k_j \ :\ {\mathbf{r}^k}^\top \mathbf{1} = 1,\ \mathbf{r}^k \geq 0 \right\} \qquad \forall k \in K. \tag{1.3}$$

The first level problem–or leader's problem– optimizes the leader's expected reward in (1.1) by selecting the mixed strategy (1.2) and taking into account the optimal follower response (1.3). The second level problems–or follower problems–in (1.3) requires each follower $k \in K$ to commit to a mixed strategy $\mathbf{q}^k$ which is a best response to the leader's strategy, $\mathbf{x}$, in that it maximizes follower k's payoff. Note that, since the problem in (1.3) is a linear optimization problem on the simplex, for any leader strategy x and any $k \in K$, there is a best response to follower k's problem that is a pure strategy, that is a vector $q^k \in \{0,1\}^{|J|}$ such that $\sum_{j\in J} q^k_j = 1$.

This observation allows us to consider that followers only respond with optimal pure strategies. Therefore, the bilevel program above can be reformulated as the following Mixed Integer Linear Program (MILP), referred to as (D2) [46]:

$$\text{(D2)} \quad \max_{x,q,s,f} \quad \sum_{k\in K} \pi^k f^k \tag{1.4}$$

$$\text{s.t.} \quad \mathbf{x}^\top \mathbf{1} = 1,\ \mathbf{x} \geq 0, \tag{1.5}$$

$${\mathbf{q}^k}^\top \mathbf{1} = 1,\ \mathbf{q}^k \in \{0,1\}^{|J|} \qquad \forall k \in K \tag{1.6}$$

$$f^k \leq \sum_{i\in I} R^k_{ij} x_i + M(1 - q^k_j) \qquad \forall k \in K,\ \forall j \in J \tag{1.7}$$

$$0 \leq s^k - \sum_{i\in I} C^k_{ij} x_i \leq M(1 - q^k_j) \qquad \forall k \in K,\ \forall j \in J, \tag{1.8}$$

Constraints (1.5) and (1.6) indicate that the leader selects a mixed strategy and each follower responds with a pure strategy. The constant M in Constraints (1.7) and (1.8) is a large positive constant relative to the highest payoff value that renders the constraints redundant if $q^k_j = 0$. In Constraint (1.7), f^k is a bound on the leader's reward when facing the follower of type $k \in K$. This bound is tight for the strategy $j \in J$ selected by that follower. In Constraint (1.8), s^k is a bound on follower k's expected payoff. This bound is tight for the best response strategy for that follower. Together, Constraints (1.7) and (1.8) ensure that the leader's strategy and each follower's strategies are mutual best responses. The objective function maximizes the leader's expected reward.

A Stackelberg game in a security setting can represent the interaction between the defender (or leader) that decides how to deploy security resources to protect n targets, and attackers (or followers) that then decide which of the n targets to attack. In these games, which we refer

to as Stackelberg Security Games (SSG), the strategy set for the attackers consists of the n targets that can be attacked $J = \{1, \ldots, n\}$. The strategy set for the defender consists of all possible ways of deploying resources to protect up to $m < n$ targets simultaneously. That is $I = \{K \mid K \subset J,\ |K| \leq m\}$ and therefore $|I| = \sum_{i=1}^{m} \binom{n}{i}$. Furthermore, in these (SSG) we assume that the rewards only depend on whether the attack on a target is successful (if it is unprotected) or not (if the target is protected). Thus, we denote by $D^k(j|c)$ the utility of the defender when facing an attacker of type $k \in K$ on covered target $j \in J$ and by $D^k(j|u)$ the utility of the defender when facing an attacker of type $k \in K$ on unprotected target $j \in J$. Similarly, the utility of an attacker of type $k \in K$ when successfully attacking an unprotected target $j \in J$ is denoted by $A^k(j|u)$ and that attacker's utility when attacking a covered target $j \in J$ is denoted by $A^k(j|c)$. Since a defender strategy $i \in I$ corresponds to a subset of J of cardinality at most m, we express as $j \in i$ the condition that defender strategy i patrols target j. The relationship between the payoffs in a security game to those in a general game are as follows:

$$R_{ij}^k = \begin{cases} D^k(j|c) & \text{if } j \in i \\ D^k(j|u) & \text{if } j \notin i \end{cases} \qquad (1.9) \qquad\qquad C_{ij}^k = \begin{cases} A^k(j|c) & \text{if } j \in i \\ A^k(j|u) & \text{if } j \notin i. \end{cases} \qquad (1.10)$$

This reward matrix structure is exploited in [37] to build a compact representation of the problem. This representation seeks to avoid enumerating the large leader strategy space I by introducing for each target $j \in J$ a coverage variable c_j that represents the frequency of coverage for that target. Therefore each coverage variable satisfies $c_j = \sum_{i \in I: j \in i} x_i$, i.e., the frequency of coverage of a target can be expressed as the sum of probabilities over strategies that assign coverage to that target. A bilinear bilevel formulation for SSGs is the following:

$$\max_{c,q} \quad \sum_{j \in J} \sum_{k \in K} \pi^k q_j^k \left\{ c_j D^k(j|c) + (1 - c_j) D^k(j|u) \right\} \qquad (1.11)$$

$$\text{s.t.} \quad \mathbf{c} \in [0,1]^{|J|}, \qquad (1.12)$$

$$\mathbf{c}^\top \mathbf{1} \leq m, \qquad (1.13)$$

$$q^k = \arg\max \left\{ \sum_{i \in I} \sum_{j \in J} r_j^k \left(c_j A^k(j|c) + (1 - c_j) A^k(j|u) \right) \ : \ {\mathbf{r}^k}^\top \mathbf{1} = 1,\, \mathbf{r}^k \geq 0 \right\} \forall k \in K. \qquad (1.14)$$

The defender maximizes the expected profit in (1.11) by committing to a coverage strategy c that satisfies (1.12) and (1.13), that is c_j is a value between 0 and 1 for each target j and that the total coverage induced by c is bounded by the number of available resources. As before, (1.14) indicates that every attacker $k \in K$ chooses a strategy q^k that maximizes their profit taking into account the coverage strategy c selected by the defender.

Further, the above bilevel program can be reformulated as a MILP, as shown in [37]:

(ERASER)

$$\max_{c,q,s,f} \quad \sum_{k\in K} \pi^k f^k \tag{1.15}$$

$$\text{s.t.} \quad \mathbf{q}^{k\top}\mathbf{1} = 1,\ \mathbf{q}^k \in \{0,1\}^{|J|} \qquad \forall k \in K, \tag{1.16}$$

$$\mathbf{c}^\top \mathbf{1} \le m,\ \mathbf{c} \in [0,1]^{|J|}, \tag{1.17}$$

$$f^k \le D^k(j|c)c_j + D^k(j|u)(1-c_j) + (1-q_j^k)\cdot M \qquad \forall j \in J, \forall k \in K, \tag{1.18}$$

$$0 \le s^k - A^k(j|c)c_j - A^k(j|u)(1-c_j) \le (1-q_j^k)\cdot M \qquad \forall j \in J, \forall k \in K, \tag{1.19}$$

$$s, f \in \mathbb{R}^K.$$

Constraints (1.16) enforce that an attacker of type $k \in K$ attacks a single target $j \in J$. Constraints (1.17) ensure that the total coverage probabilities on the targets does not exceed the number of available resources. In Constraints (1.19) and (1.18), M is a large positive constant relative to the highest payoff. For the target $j \in J$ attacked by attacker $k \in K$, Constraint (1.18) provides a tight bound on the defender's expected payoff when facing an attacker of type $k \in K$. For any other target, the RHS of Constraint (1.18) is arbitrarily large. Similarly, Constraint (1.19) ensures that for each attacker $k \in K$, s^k is a lower bound on that attacker's payoff and provides the optimal payoff for that attacker for the target that attacker attacks. The objective function maximizes the defender's expected utility. The authors in [37] further show that every feasible coverage probability vector corresponds to a mixed strategy, *i.e.*, a deployment of resources to targets, and an optimal coverage probability vector and an optimal attack vector for each attacker type corresponds to an SSE of the game.

1.2.2. Resource Combination in a SSG

We now formulate a problem were the defender first teams up the resources from different precincts to form m combined patrols and then decides where to deploy these combined patrols. Let V be the set of police precincts. We let $E \subset V \times V$ be the set of edges representing the set of possible precinct pairings, forming an adjacency graph $G = (V, E)$. We denote by $\delta(v) \subset E$ the set of edges incident to precinct $v \in V$, similarly for any $U \subset V$, $\delta(U) \subset E$ denotes the edges between U and $V \setminus U$, and $E(U) \subset E$ denotes the edges between precincts in U. We can then represent the possible combinations of m precincts pairs as the set of matchings of size m, which is given by:

$$\mathcal{M}_m := \left\{ \mathbf{y} \in \{0,1\}^{|E|} : \sum_{e\in E} y_e = m,\ \sum_{e\in\delta(v)} y_e \le 1 \quad \forall v \in V \right\}. \tag{1.20}$$

For every precinct $v \in V$, let J_v be the set of targets to patrol that are inside that precinct. Note that $\{J_v\}_{v\in V}$ is a partition of the set of targets J, *i.e.*, $\cup_{v\in V} J_v = J$ and $J_u \cap J_v = \emptyset$ for all $u \neq v$. The set of defender strategies selects the m precinct pairings and also the target where each resource team is deployed. The combined patrol from the pairing of precincts u

and v can only be deployed to a target in $J_u \cup J_v$. For each edge $e = (u, v) \in E$ we define $J_e = J_u \cup J_v$. It follows that the set I of defender strategies can be expressed as

$$I = \left\{ (\mathbf{y}, \mathbf{w}) \in \{0,1\}^{|E|} \times \{0,1\}^{|J|} : \mathbf{y} \in \mathcal{M}_m, \sum_{j \in \cup_{v \in U} J_v} w_j \leq \sum_{e \in E(U) \cup \delta(U)} y_e \, \forall U \subseteq V, \sum_{j \in J} w_j = m \right\}. \tag{1.21}$$

For $(y, w) \in I$, the variable y_e indicates whether edge e is selected for a precinct pairing and w_j indicates whether target j is patrolled. The first condition indicates that the coverage provided to any subset of targets is bounded by the coverage on all incident edges to this subset of targets. The second condition enforces that total coverage is equal to the required number of resources.

For any pairing of resources and deployment strategy decided by the defender $(y, w) \in I$, an attacker of type $k \in K$, appearing with probability π^k, responds optimally, that is, attacks the target $j \in J$ that yields the largest payoff for the attacker. The defender therefore has to select the pairing and deployment strategy $(y, w) \in I$ that maximizes its own payoff, taking into account this best attacker response. This Stackelberg security game can be represented by explicitly enumerating all strategies and formulating the MILP (D2) whose solution gives the optimal mixed strategy for the defender. Unfortunately, this approach is computationally challenging as the resulting MILP considers one variable $x_{(y,w)}$ for each $(y, w) \in I$, which is an exponential number of variables in terms of targets and edges. Below we introduce a compact MILP formulation with a polynomial number of variables and exponentially many constraints. We show the equivalence and the correctness of a decomposition algorithm for this new formulation in the next section.

The compact formulation is derived similarly to (ERASER) and is based on the observation that if rewards are given by (1.9) and (1.10) then the utility of each player only depends on c_j, the coverage at a target $j \in J$. Where this coverage is determined by the sum of $x_{(y,w)}$ the probability of selecting defender strategy $(y, w) \in I$ for all strategies that cover target j. Similarly we can express the coverage probability of edge $e \in E$ by z_e, the sum of $x_{(y,w)}$ over defender strategies that cover edge e. That is, we define variables c_j and z_e as follows

$$c_j = \sum_{(y,w) \in I \,:\, w_j = 1} x_{(y,w)} \qquad j \in J \tag{1.22}$$

$$z_e = \sum_{(y,w) \in I \,:\, y_e = 1} x_{(y,w)} \qquad e \in E \; . \tag{1.23}$$

Given a graph $G = (V, E)$ with V the set of precincts and E the feasible pairings between

precincts, we propose the following formulation:

(COMB)

$$\max_{c,z,q,s,f,g} \quad \sum_{k\in K} \pi^k f^k \tag{1.24}$$

$$\text{s.t.} \quad \mathbf{q}^{k\top}\mathbf{1} = 1,\ \mathbf{q}^k \in \{0,1\}^{|J|} \qquad \forall k \in K, \tag{1.25}$$

$$\sum_{j\in J} c_j = \sum_{e\in E} z_e = m \qquad \forall v \in V, \tag{1.26}$$

$$\sum_{e\in\delta(v)} z_e \le 1 \qquad \forall v \in V, \tag{1.27}$$

$$\sum_{e\in E(U)} z_e \le \frac{|U|-1}{2} \qquad \forall U \subseteq V,\ |U| \ge 3, |U| \text{ odd} \tag{1.28}$$

$$\sum_{e\in E: j\in J_e} g_{e,j} = c_j \qquad \forall j \in J \tag{1.29}$$

$$\sum_{j\in J_e} g_{e,j} = z_e \qquad \forall e \in E \tag{1.30}$$

$$f^k \le D^k(j|c)c_j + D^k(j|u)(1-c_j) + (1-q_j^k)\cdot M \qquad \forall j \in J, \forall k \in K, \tag{1.31}$$

$$0 \le s^k - A^k(j|c)c_j - A^k(j|u)(1-c_j) \le (1-q_j^k)\cdot M \qquad \forall j \in J, \forall k \in K \tag{1.32}$$

$$\mathbf{c} \in [0,1]^{|J|},\ \mathbf{z} \in [0,1]^{|E|},\ \mathbf{s},\mathbf{f} \in \mathbb{R}^K,\ \mathbf{g} \in [0,1]^{|E||J|}\ . \tag{1.33}$$

Constraints (1.25) ensure that the each attacker $k \in K$ attacks a single target $j \in J$ with probability 1. Constraint (1.26) indicates that the defender uses all his resources in a feasible solution and that in order to form his resources he pairs up precincts without exceeding the number of resources he wants to deploy. Constraint (1.27) indicates that a precinct's contribution to a pairing cannot exceed 1. Constraints (1.28) correspond to the *Odd Set Inequalities*, as introduced in [24], and together with (1.26) and (1.27) enforce that the coverage probabilities on the edges belong to the convex hull of the matching polytope of size m. Constraints (1.29) and (1.30) enforce the conservation between marginal coverages in nodes and edges. Finally, Constraints (1.31) and (1.32) are the same as in (ERASER) and ensure that c and q are mutual best responses. The objective function in our formulation, maximizes the defender's expected utility.

For the correctness of problem (COMB) we need to be able to recover the variables $x_{(y,w)}$ for $(y,w) \in I$ that represent the defender probability distribution over its set of feasible ac-

tions from the variables $\mathbf{c},\ \mathbf{z},\ \mathbf{q},\ \mathbf{s},\ \mathbf{f},\ \mathbf{g}$ in (COMB). In particular, we need to find variables $\mathbf{x} \in [0,1]^{|I|}$ that satisfy constraint (1.23). We now show that the odd set inequalities (1.28) are necessary, giving an example of z variables for which there does not exist a probability distribution over I that would satisfy (1.23). Consider the example in Figure 1.1, which shows non-negative values for z variables that satisfy constraints (1.26) and (1.27) for $m = 2$ but that violate the odd set inequalities (with the set $U = \{3, 4, 5\}$). We observe that this solution cannot be expressed as a convex combination of pure matchings of size 2, making it impossible to retrieve an implementable defender strategy $\mathbf{x}$.

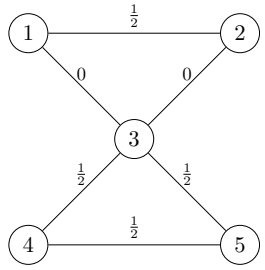


Figure 1.1: Variables $\mathbf{z} \in [0,1]^{|E|}$ that satisfy (1.26) and (1.27) but violate (1.28) for $m = 2$.

Similarly, Constraints (1.29) and (1.30) also play a vital role in that they establish a link between the coverage variables on the edges and on the targets. This becomes much more apparent if one applies Farkas' Lemma [25] on the linear system defined by (1.29), (1.30) and $g \geq 0$ to understand which conditions on c and z guarantee feasibility of the system. Applying Farkas provides the following conditions on c and z which offer a more direct interpretation:

$$\sum_{e \in E : e \in E(U) \cup \delta(U)} z_e \geq \sum_{j \in \cup_{u \in U} J_u} c_j \quad \forall U \subseteq V, \tag{1.34}$$

$$\sum_{j \in J : j \in \cup_{u \in V : \delta(u) \cap E' \neq \emptyset} J_u} c_j \geq \sum_{e \in E'} z_e \quad \forall E' \subseteq E. \tag{1.35}$$

Constraint (1.34) states that given a subset of nodes, the coverage provided on all targets inside these nodes cannot exceed the weight of the edges incident to these nodes. Constraint (1.35) indicates that given a fixed set of edges E', the weights on those edges is a lowerbound on the coverage of the targets in nodes to which those edges are incident. Figure 1.2 shows an example of variables $\mathbf{z} \in [0,1]^{|E|}$ and $\mathbf{c} \in [0,1]^{|J|}$ that satisfy all constraints in (COMB) but (1.29) and (1.30). The numbers on the nodes represent total coverage on targets in that node, $\sum_{j \in J_u} c_j$ and the numbers on the edges represent the coverage probabilities on the edges, z_e. The solution in Figure 1.2 violates (1.34) for $U = \{1, 2\}$. It is also not possible to find in this example an implementable defender strategy $\mathbf{x} \in [0,1]^{|I|}$ related to these variables $\mathbf{z}$ and $\mathbf{c}$.

1.3. Decomposition Model

In this section we show the equivalence of problem (COMB) to the (D2) formulation for Stackelberg security games. Furthermore we present a decomposition algorithm to solve (COMB) and we finish by presenting methods to generate implementable defender strategies $\mathbf{x} \in [0,1]^{|I|}$ by sampling solutions from (COMB).

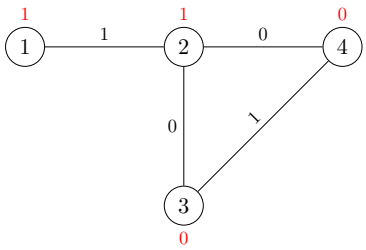


Figure 1.2: Variables $\mathbf{z} \in [0,1]^{|E|}$ and $\mathbf{c} \in [0,1]^{|J|}$ that do not satisfy (1.29) and (1.30) with $m = 2$.

1.3.1. (COMB) is an equivalent SSG formulation

To set the notation, we consider a combined resources SSG problem with defender action set given by (1.21) and defender and attacker utilities given by (1.9) and (1.10), respectively. We show that (COMB) is equivalent to (D2) for a SSG with resource combination by showing that a feasible solution from one leads to a feasible solution in the other with same objective value and viceversa. We first prove the following technical lemma.

Lemma 1.1 Given $(y, w) \in I$ then the following holds

$$w_j = 1 \Rightarrow \sum_{e \in E: j \in J_e} y_e = 1 \tag{1.36}$$

$$y_e = 1 \Rightarrow \sum_{j \in J_e} w_j = 1. \tag{1.37}$$

Proof. Since $(y, w) \in I$ with $w_j = 1$, if we denote $v(j)$ the precinct that contains $j \in J_{v(j)}$, we have

$$1 = w_j \leq \sum_{k \in J_{v(j)}} w_k \leq \sum_{e \in \delta(v(j))} y_e = \sum_{e \in E: j \in J_e} y_e \ ,$$

which proves (1.36). Here the first inequality comes from $w_k \in \{0,1\}$, the second from the definition of set I, and the last equality represents two equivalent ways of expressing the sum over the edges incident on $v(j)$. For the second implication, let $y_{e_1} = y_{e_2} = \cdots = y_{e_m} = 1$ be the set of arcs that are set to one in the matching y. Let $A(y)$ be the set of precincts v that are not paired by $y_{e_1} \ldots y_{e_m}$ (i.e. $A(y) = \{v \ : \ \delta(v) \notin \cup_{k=1}^{m} e_k\}$). Then, for any $v \in A(y)$ the definition of I implies $\sum_{k \in J_v} w_k \leq \sum_{e \in \delta(v)} y_e = 0$. Similarly, for any arc $e_k = (u_k, v_k)$ we have, from the definition of I, that $\sum_{j \in J_{e_k}} w_j \leq y_{e_k} + \sum_{e \in \delta(\{u_k, v_k\})} y_e = y_{e_k} = 1$. Here the sum in the second term is zero because $e \in \delta(\{u_k, v_k\})$ are not in matching y. With these inequalities we can separate

$$m = \sum_{j \in J} w_j = \sum_{v \in A(y)} \sum_{j \in J_v} w_j + \sum_{k=1}^{m} \sum_{j \in J_{e_k}} w_j \leq \sum_{k=1}^{m} y_{e_k} = m \ ,$$

which completes the proof because the m terms $\sum_{j \in J_{e_k}} w_j$ are both less than 1 and sum to m. □

Theorem 1.2 Problem (COMB) given by (1.24)-(1.33) is equivalent to (D2) given by (1.4)-(1.8).

PROOF. We begin with a solution (x, q, s, f) feasible for (D2). Given this x we define $\mathbf{c} \in [0,1]^{|J|}$ and $\mathbf{z} \in [0,1]^{|E|}$ as in (1.22) and (1.23), respectively. Furthermore we define $g_{e,j}$ as follows:

$$g_{e,j} = \begin{cases} \sum_{(y,w)\in I: y_e=1, w_j=1} x_{(y,w)} & \forall j \in J, \forall e \in E \text{ such that } j \in J_e \\ 0 & \text{o/w} \ . \end{cases} \tag{1.38}$$

We now show that this (c, z, q, s, f, g) is feasible for (COMB) and has the same objective value. Indeed, the objective function (1.24) and constraints (1.25) in (COMB) are the same as (1.4) and (1.6) in (D2). In addition constraints (1.33) are satisfied from the definition of $\mathbf{c}$, $\mathbf{z}$ and $\mathbf{g}$ and the fact that $\mathbf{x}$ is a probability distribution.

Since the utilities are given by (1.9) and (1.10), we have that

$$\begin{aligned} \sum_{(y,w)\in I} R^k_{(y,w)j} x_{(y,w)} &= \sum_{(y,w)\in I: w_j=1} R^k_{(y,w)j} x_{(y,w)} + \sum_{(y,w)\in I: w_j=0} R^k_{(y,w)j} x_{(y,w)} \\ &= D^k(j|c) \sum_{(y,w)\in I: w_j=1} x_{(y,w)} + D^k(j|u) \left(1 - \sum_{(y,w)\in I: w_j=1} x_{(y,w)}\right) \\ &= D^k(j|c) c_j + D^k(j|u)\,(1 - c_j) \ . \end{aligned}$$

The last equality uses the definition of c_j (1.22). This equality is used to say that (1.7) implies (1.31). Analogously for the attackers utility we have that (1.8) implies that (1.32) also holds.

From the definition of $g_{e,j}$ we can write

$$\begin{aligned} \sum_{j\in J_e} g_{e,j} &= \sum_{j\in J_e} \sum_{\substack{(y,\,w)\,\in\, I \\ y_e\,=\,1,\, w_j\,=\,1}} x_{(y,w)} = \sum_{j\in J_e} \sum_{(y,w)\in I} x_{(y,w)} y_e w_j \\ &= \sum_{(y,w)\in I} x_{(y,w)} y_e \sum_{j\in J_e} w_j = \sum_{(y,w)\in I} x_{(y,w)} y_e = z_e \ . \end{aligned}$$

Where the next to last equality follows from (1.37) in Lemma (1.1) and the last equality is the definition of z_e in (1.23). This shows that (1.30) is satisfied. Similarly simplifying the $\sum_{e\in E: j\in J_e} g_{e,j}$ and now using (1.36) in Lemma (1.1) and definition (1.22) gives that (1.29) is satisfied. In the same logic it follows that

$$\sum_{e\in E} z_e = \sum_{j\in J} c_j = \sum_{e\in E} \sum_{j\in J_e} g_{e,j} = \sum_{e\in E} \sum_{(y,w)\in I} x_{(y,w)} y_e = \sum_{(y,w)\in I} x_{(y,w)} \sum_{e\in E} y_e = m \ .$$

This proves that (1.26) is satisfied. Finally we observe above that the vector $z \in [0,1]^{|E|}$ satisfies $\mathbf{z} = \sum_{(y,w)\in I} x_{(y,w)} \mathbf{y}$ where $\mathbf{x}$ is a probability distribution over I and the vectors $\mathbf{y}$ correspond to m-matchings in the set V. Therefore $\mathbf{z} \in$ convex hull$(\mathcal{M}_m)$, and therefore $\mathbf{z}$ satisfies the constraints (1.27) and (1.28) that characterize the m-matching polytope. This completes this first part of the proof.

We now consider a solution (c, z, q, s, f, g) feasible for (COMB) and construct a solution (x, q, s, f) that is feasible for (D2) with the same objective function. Variables q, s and f are the same, this shows that the objective (1.4) and constraint (1.6) in (D2) are satisfied as they correspond to (1.24) and (1.25) respectively. To build $\mathbf{x}$ we first recognize that by constraints (1.27) and (1.28), the vector $\mathbf{z} \in$ convex hull$(\mathcal{M}_m)$. Therefore, there is a finite set of integer m-matchings $M_z \subseteq \mathcal{M}_m$ such that we can express $\mathbf{z} = \sum_{y \in \mathcal{M}_z} \lambda_y \mathbf{y}$, where λ_y are the convex combination weights of the integer m-matchings $y \in M_z$. That is $\lambda_y \geq 0$ for all $y \in M_z$ and $\sum_{y \in M_z} \lambda_y = 1$. Given this decomposition of $\mathbf{z}$, we perform the following construction, illustrated in Figure 1.3.

Step 1. For every edge $e \in E$ consider a column of height 1. Divide this $1 \times |E|$ rectangle in horizontal segments, one for each integer matching $y \in M_z$, with each segment of width λ_y. For each segment, corresponding to matching y, block out the edges that are not used in matching y (marked 'NO' in Figure 1.3).

Step 2. For every edge $e \in E$, further subdivide the area in its column and matchings that contain e using the values of g_{ej} for all $j \in J_e$. Since $\sum_{j \in J_e} g_{ej} = z_e = \sum_{y \in M_z} \lambda_y y_e$, from (1.30) and the decomposition of $\mathbf{z}$, this partition uses all the area in the column of e that was not blocked out.

Step 3. Define $\mathbf{x}$ by identifying the largest width horizontal lines formed in this diagram, ignoring blocked out squares. These horizontal lines are contained in a matching y and for each edge e in the matching ($y_e = 1$), include part of some g_{ej_e}. The matching y indicates the precinct pairings and the targets $j_e \in J_e$ identify the protected targets of the strategy. Let the width of this horizontal line be $x_{(y,w)}$, where $w \in \{0,1\}^{|J|}$ is the indicator vector of the protected targets $\{j_e\}_{\{e:y_e=1\}}$.

For example, in Figure 1.3, we notice that matching y_2 includes edges $e_1, e_3, \ldots, e_{|E|}$. Furthermore a constant width horizontal line (shaded) can be identified using portions of $g_{14}, g_{38}, \ldots, g_{|E|11}$. This shaded horizontal line corresponds to using precinct parings according to matching y_2 and protecting targets $w = \{4, 8 \ldots, 11\}$, whose width we label as $x_{(y_2,w)}$.

We now show that all the (y, w) identified in Step 3 satisfy $(y, w) \in I$. First we have by construction that $y \in \mathcal{M}_m$. In addition for each edge $e = \{u_e, v_e\}$ used in y we identify a single target $j_e \in J_e = J_{u_e} \cup J_{v_e}$ so that $w_{j_e} = 1$. Therefore the vector w considers exactly m targets selected. For any $U \subseteq V$, we have

$$\sum_{j \in \cup_{v \in U} J_v} w_j \leq \sum_{\substack{j_e \in J_{u_e} \cup J_{v_e} \\ \{u_e, v_e\} \cap U \neq \emptyset}} w_{j_e} = \sum_{e \,:\, \{u_e, v_e\} \cap U \neq \emptyset} y_e = \sum_{e \in E(U) \cup \delta(U)} y_e \,.$$

This proves that $(y, w) \in I$. Also, by construction, the horizontal lines corresponding to defender strategies cover the box completely, since the entire area of the $1 \times |E|$ that is not blocked out is covered by some $g_{ej} > 0$. Therefore Step 3 can be performed at any height of the box. This implies that the values $x_{(y,w)}$ constructed in Step 3 are non-negative and sum to one, therefore they form a probability distribution over $(y, w) \in I$. This means that the vector $\mathbf{x}$ constructed satisfies (1.5). To show that constraints (1.7) and (1.8) are satisfied just we need to show that the $\mathbf{x}$ constructed satisfies $c_j = \sum_{(y,w) \in I \,:\, w_j = 1} x_{(y,w)}$. This result is then used in a similar derivation than in the first part of the proof to show that (1.31) and (1.32) implies (1.7) and (1.8).

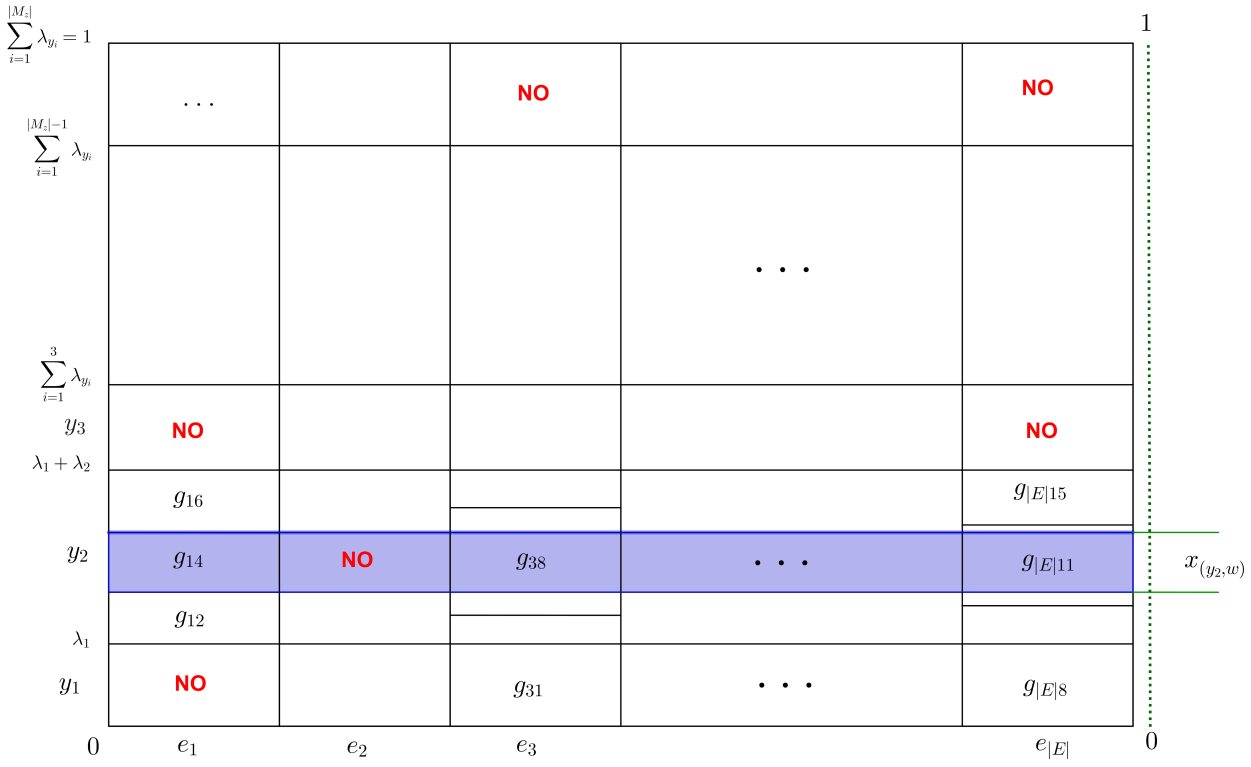


Figure 1.3: Construction to identify implementable mixed strategy $\mathbf{x}$ given vectors $\mathbf{z}, \mathbf{c}, \mathbf{g}$ feasible for (COMB).

Consider a target $j \in J$ and e such that $j \in J_e$ with $g_{ej} > 0$. We have from Step 2 in the construction above that the full amount of g_{ej} is assigned to some area in the column e intersected with matchings that use that column. That area can be partitioned into many strategies (y, w) but each of them has $y_e = 1$ and $w_j = 1$. Therefore we have that $g_{ej} = \sum_{(y,w)\in I} x_{(y,w)} w_j y_e$. We have then

$$c_j = \sum_{e:j\in J_e} g_{ej} = \sum_{e:j\in J_e} \sum_{(y,w)\in I} x_{(y,w)} w_j y_e = \sum_{(y,w)\in I} x_{(y,w)} w_j \sum_{e:j\in J_e} y_e = \sum_{(y,w)\in I} x_{(y,w)} w_j \ .$$

Here the first equality comes from (1.29), the second from the observation above, and the last from (1.36). □

1.3.2. Cutting Odd-Set Constraints

While the (COMB) formulation requires far fewer variables than the (D2) formulation for problems that combine resources, it still considers an exponential number of odd set inequalities (1.28). In this subsection we show how to separate the odd set inequalities in (COMB) following the methodology in [45].

We begin by noting that the odd set inequalities (1.28) are equivalent to the following

constraint:

$$\sum_{v \in U} s_v + \sum_{e \in \delta(U)} z_e \geq 1, \quad |U| \geq 3, \, |U| \text{ odd}. \tag{1.39}$$

This can be shown by adding slack variables s_v, $v \in V$ for every Constraint (1.27). Summing Constraints (1.27) over a fixed subset of nodes U yields:

$$\sum_{v \in U} \sum_{e \in \delta(v)} z_e + \sum_{v \in U} s_v = |U|, \quad s_v \geq 0 \quad \forall v \in V. \tag{1.40}$$

Using the fact that

$$\sum_{u \in U} \sum_{e \in \delta(u)} z_e = 2 \sum_{e \in E(U)} z_e + \sum_{e \in \delta(U)} z_e$$

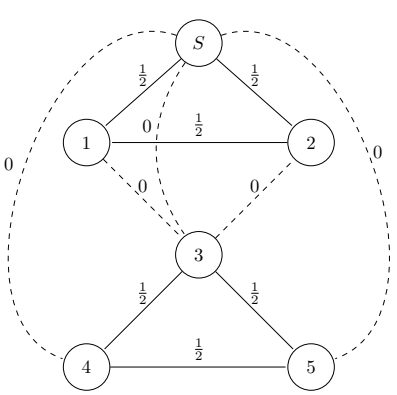
and (1.40) we can simplify the odd set inequalities (1.28) to obtain (1.39).

Thus, a solution $(z^*, \{s_v^* = 1 - \sum_{e \in \delta(v)} z_e^*\}_{v \in V})$ violates an odd set inequality if there exists a subset of nodes U with odd cardinality such that $\sum_{v \in U} s_v^* + \sum_{e \in \delta(U)} z_e^* < 1$. To detect such condition is equivalent to detect a global min cut of odd cardinality in a suitable graph $G' = (V', E')$ where $V' = V \cup \{\mathcal{S}\}$, being $\mathcal{S}$ a dummy node and $E' = E \cup \{(v, \mathcal{S}) | v \in V\}$. The capacities are set z_e for each $e \in E$ and s_v for each of the form $(v, \mathcal{S})$. The minimum cut of this network can be computed with the Gomory Hu algorithm and finding the odd cardinality set with minimum capacity. If the capacity of the minimum odd cardinality set is smaller than one then the set of nodes U not containing $\mathcal{S}$ violates constraint 1.39. A simple implementation of the Gomory Hu algorithm is given by [29]. This procedure is described in Algorithm 1.

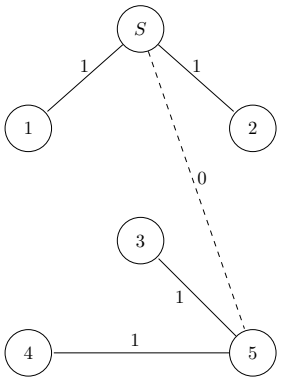
Algorithm 1 Min odd-cut set

1: Graph $G = (V, E)$, $z \in \mathbb{R}_+^{|E|}$
2: Create a graph $G' = (V', E')$ where $V' = V \cup \{\mathcal{S}\}$ and $E' = E \cup \{(v, \mathcal{S}) | v \in V\}$, and capacities $\bar{r}_{e'} = z_{e'}$ for edges $e' \in E$ and $\bar{r}_{e'} = s_v$ for edges $e' = (v, \mathcal{S}) \in E' \backslash E$.
3: Compute a Gomory - Hu tree $G_T = (V_T, E_T)$ in the graph G' and set $r = +\infty$
4: **for** $e \in E_T$ **do**
5: Calculate the cut U induced by deleting e in E_T.
6: **if** $|U|$ is odd and $r_e < r$ **then**
7: $X \leftarrow U$ and $r \leftarrow r_e$
8: **end if**
9: **end for**
10: **return** X the odd minimum cut-set and r its capacity.

Figure 1.4 shows an example of how the algorithm works cutting the infeasible solution exposed in Figure 1.1. We represent the extended graph on the left and the min-cut set in dashed lines and show, on the right, how the Gomory-Hu tree computes this minimum cut. Note that in the example the min-cut odd set is formed by the set of nodes $\{3, 4, 5\}$ with cost 0 and the associated constraint is $z_{\{3,4\}} + z_{\{3,5\}} + z_{\{4,5\}} \leq 1$ which cuts the solution in Figure 1.1.



a) Extended graph of solution in Fig. 1.1



b) Gomory-Hu Tree

Figure 1.4: Representation of the algorithm which detects the Odd min cut set. The dashed edges shows the min cut of this graph.

We use Algorithm 1 to implement a generic cut-generation algorithm to face large-sized instances. This procedure is described in Algorithm 2. We denote by $\mathcal{U}$ the set of nodes of odd cardinality in constraints (1.28), and by $D(\mathcal{U})$ the optimization scheme that includes these sets in the model. The main idea is that given an optimal solution of $D(\mathcal{U})$, z^*, at some iteration, we check if any of the odd set inequalities, not already in $\mathcal{U}$, is violated. If there are not any then the algorithm obtains an optimal solution. Otherwise, the procedure adds the violated inequality to $D(\mathcal{U})$ and re-optimizes.

Algorithm 2 Constraint Generation Algorithm

Set $\mathcal{U} = \emptyset$
for $i = 1, \cdots, MAX\_IT$ **do**
  $z^* \leftarrow \mathrm{D}(\mathcal{U})$
  $costMinOddCut, U \leftarrow$ Min odd-cut set(G,z^*)
  if $costMinOddCut \geq 1$ **then**
    break
  else
    $\mathcal{U} \leftarrow \mathcal{U} \cup \{U\}$
  end if
end for
return The optimal coverage (z^*, c^*)

1.3.3. Recovering an implementable strategy

In this section we consider two different ways of recovering an implementable mixed strategy x^* which respects the optimal probabilities on targets and edges $(\mathbf{z}^*, \mathbf{c}^*)$ returned by (COMB). In Section 1.3.1 we propose a box method to recover the implementable mixed strategy. This method relies heavily on being able to decompose a fractional matching of size m as convex combination of pure matchings of size m. Let us take a closer look at how this

can be achieved.

Decomposing $\mathbf{z}^*$ in pure matchings of size m

Given a vector $(\mathbf{z}^*, \mathbf{c}^*)$ we want to build a set $M_z \subseteq \mathcal{M}_m$ of matchings of size m and a set of weights $\{\lambda_y\}_{y \in \mathcal{M}_m}$ such that the vector $\mathbf{z}^*$ can be written as a convex combination of elements in M_z. This is possible because the cardinality constrained matching polytope is integral, [52]. Then the problem is reduced to distributing the weights λ such that $\mathbf{z}^*$ is respected.

We propose two methodologies to achieve this goal. First we propose a mixed integer program considering $|E| + 1$ matchings are neccesary to create the decomposition, and a Dantzig-Wolfe approach where the matchings are created on-the-run. The mixed integer programming formulation is the following:

$$\min \quad \sum_{y \in M_z} ord(y)\lambda_y \tag{1.41}$$

$$\sum_{e \in \delta(v)} \theta_{ye} \leq 1 \qquad \forall y \in M_z \tag{1.42}$$

$$\sum_{e \in E} \theta_{ye} = m \qquad \forall y \in M_z \tag{1.43}$$

$$\sum_{y \in M_z} \lambda_y = 1 \tag{1.44}$$

$$\sum_{y \in M_z} \delta_{ye} = z_e \qquad \forall e \in E \tag{1.45}$$

$$\delta_{ye} \leq \theta_{ye} \qquad \forall e \in E, \forall y \in M_z \tag{1.46}$$

$$\lambda_y - 1 + \theta_{ye} \leq \delta_{ye} \leq \lambda_y \qquad \forall e \in E, \forall y \in M_z \tag{1.47}$$

$$\lambda_y \geq 0 \qquad \forall y \in M_z \tag{1.48}$$

$$\theta_{ye},\ \delta_{ye} \in \{0, 1\} \qquad \forall e \in E,\ y \in M_z \tag{1.49}$$

where the variables θ_{ye} are binary decisions taking value 1 if the edge e belongs to the matching $y \in \mathcal{M}_m$ and 0 otherwise, λ_y is the weight of the matching $y \in \mathcal{M}_m$ and variables δ_{ye} represents the product between λ_y and θ_{ye}. Constraints (1.42) and (1.43) define that the variables θ_{ie} are matchings of size m. Constraints (1.44) and (1.45) ensure that the matchings retrieved can build the vector $\mathbf{z}$ as a convex combination of these matchings. The linearization of the product $\lambda_y\theta_{ye}$ is ensured by (1.47) and (1.48), and the fact that θ_{ye} is a binary variable. The objective function pretends to get a set of matchings with the minimum cardinality, giving more weight to the matchings with highest $ord(y)$. We note that any objective function is useful due to the focus on this part is feasibility and not optimality.

This problem can also be solved with a Dantzig-Wolfe approach, [20]. Let M_z^t the set of matchings considered at the step t of the procedure. The master problem can be stated as:

$$\text{(MP)} \quad \min \quad \sum_{e \in E} Y_e \tag{1.50}$$

$$Y_e + \sum_{i \in M_z^t} \theta_{ye} \lambda_y = z_e \qquad \forall e \in E \tag{1.51}$$

$$\sum_{y \in M_z^t} \lambda_y = 1 \tag{1.52}$$

$$\lambda_y \geq 0 \qquad \forall y \in M_z^t \tag{1.53}$$

where as in mixed integer formulation, λ_y is the weight of matching y. Y_e variables are auxiliary variables to be minimized. Notice that in this model θ_{ye} is a parameter which has the same meaning than the last model. Let π_e and σ be the optimal dual variable associated to constraint (1.51) and (1.52) respectively. Then, the reduced costs of any new column/matching generated is given by:

$$r_\theta = 0 - \sum_{e \in E} \pi_e \theta_e - \sigma. \tag{1.54}$$

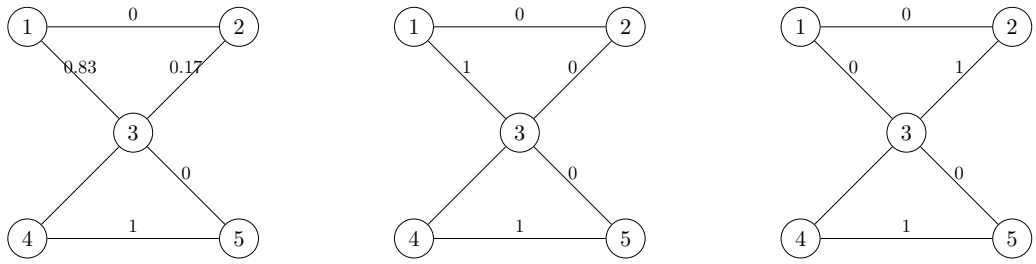
The problem of adding a new column can be stated as a maximum weight matching of size m with weights $\{\pi_e\}_{e \in E}$. If the optimal cost is greater than $-\sigma$ a new matching θ is added to M_z^{t+1}. The algorithm stops when there is no new column to be added or the objective function of (MP) is equal to zero, and z can be written as a convex combination of matchings $\{\theta_y\}_{y \in M_z^t}$ with weights $\{\lambda_y\}_{y \in M_z^t}$.

We implement a warm start using a greedy algorithm. We describe this procedure in Algorithm 3. The algorithm first is initialized with an empty set of matchings, and the problem of maximum matching of size m is solved considering z as weights of the matching. Then, the matching retrieved is assigned a weight λ equal to the minimum edge weight selected by the matching, and this λ is subtracted to the edges belonging to that matching. The algorithm continues until there is no more weight to allocate to matchings. This condition can be achieved if all the weight has been assigned to the corresponding matchings, or if at any iteration the value of λ retrieved is equal to 0.

Algorithm 3 Warm Start (G,z^*, m)

```
c = z*, M_z^0 = ∅, λ = null, it = 0.
while Σ_{e∈E} c_e > 0 do
    M[it] = max_matching_size(N, E, c, m)
    λ[it] = min_{e∈M[it]} c_e
    for e ∈ M[it] do
        c_e = c_e − λ[it]
    end for
    it = it + 1
end while
return M_z^0, λ
```

An example of how this algorithm works is represented in Figure 1.5. Figure 1.5.a shows a solution z for a problem with $m = 2$. The maximum weight matching of size 2 in this graph has weight 0.83 (Fig.1.5.b). Then subtracting this amount to the edges belonging to this matching there are only two edges with positive weight, that conforms the second matching with weight 0.17. In this case, the warm start returns a pair of matchings and weights such that z is a convex combination of them. This is not always possible. Anyway, this algorithm can always be used to accelerate the Dantzig Wolfe procedure described above.



a) Fractional matching b) $\lambda = 0{,}83$ c) $\lambda = 0{,}17$

Figure 1.5: Example of the warm start algorithm.

Next, we propose an alternative sampling method which allows to recover an implementable mixed strategy x^* given optimal coverage probabilities (c^*, z^*).

An alternative sampling method

The recovery of an implementable strategy through the box method described in Section 1.3.1 can be computationally challenging. In particular, as seen in the previous section, decomposing our optimal z into a convex combination of pure matchings of size m is not an easy task. In this section, we propose a more *naïve* sampling method in two stages that recovers an implementable defender strategy. In Section 1.5, we study the accuracy of this sampling method.

Given z^*, the coverage vector over $|E|$ of size m, we can discard all the edges in E such that $z_e = 0$. We can then select m of the remaining edges according to a uniform random variable $U(0, m)$. This, in itself, could provide edges that do not form a matching. Hence, we need to do something a bit more subtle. Let M be the set of m edges we have sampled. Now, let us solve the following optimization problem:

$$\begin{aligned} \text{Max} \quad & \sum_{e \in M} z_e^* x_e \\ & \text{s.t.} \mathbf{x} \in \mathcal{M}_{\mathbf{m}}. \end{aligned} \tag{1.55}$$

Out of all matchings of size m, the objective function guarantees that we pick a maximum weight matching. Solving the above formulation for our initial set M will have two possible outcomes. Either the model will return an optimal solution, in which case the edges in M

generate a matching of size m, or, the problem is infeasible and such a matching cannot be constructed. If the problem returns a matching, we now that this matching agrees with the optimal distribution z^*. If the problem is infeasible, we sample a new edge which we add to the set M and we re-optimize. We proceed in this iterative fashion until we construct a matching of size m. Note that this algorithm will produce a matching in at most $|E| - m$ iterations, as we know that such a matching exists in the original graph.

Finally, given M^*, our optimal matching of size m, we sample an allocation of resources to targets that satisfies the optimal coverage probability returned by our formulation as follows. We discard targets j that belong to precincts which are not paired. For each target j that belongs to a paired pair of precincts, say u and v, we normalize their coverage probability by the weight of the total coverage provided by the optimal coverage vector c^* in the two areas u and v that are paired and denote it by $\bar{c}_j^*$:

$$\bar{c}_j^* = \frac{c_j^*}{\sum_{j \in J_u \cup J_v} c_j^*} \qquad \forall (u, v) = e \in M.$$

This way, we ensure that one resource is available per paired pairs of precincts. The weekly schedule is composed sampling over the newly constructed $\bar{c}^*$.

1.4. Case Study: Carabineros de Chile

In this section, we describe a realistic border patrol problem proposed by Carabineros de Chile. In this problem, Carabineros considers three different types of crime, namely, drug trafficking, contraband and illegal entry. In order to minimize the free flow of these types of crime across their borders, Carabineros organizes both day shift patrols and night shift patrols along the border, following different patterns and satisfying different requirements.

We are concerned with the specific actions that Carabineros can take during night shift patrols. The region is divided into several police precincts. Due to the vast expanses and harsh landscape at the border to patrol and the lack of manpower, for the purpose of the defender actions under consideration, a number of these precincts are paired up when planning the patrol. Furthermore, Carabineros are aware of a fixed number of locations along the border of the region that can serve as vantage points from where to conduct surveillance with technical equipment such as night goggles and heat sensors. A night shift action consists in deploying a joint detail with personnel from two paired precincts to conduct vigilance from 22h00 to 04h00 at the vantage point located within the territory of the paired precincts. Due to logistical constraints, for a given precinct pair, Carabineros deploys a joint detail from every precinct pair to a surveillance location once a week.

Carabineros requires a schedule indicating the optimal deployment of details to vantage points for a given week. Figure 1.8 depicts a defender strategy in a game with $m = 3$ pairings, $|V| = 7$ precincts and $|J| = 10$ locations. Table 1.1 shows a tabular representation of the implemented strategy for that week.

Therefore, we have an adjacency graph $G(V, E)$ where V is the set of police precincts

Figure 1.6: A Carabinero conducts surveillance.

Figure 1.7: Harsh border landscape.

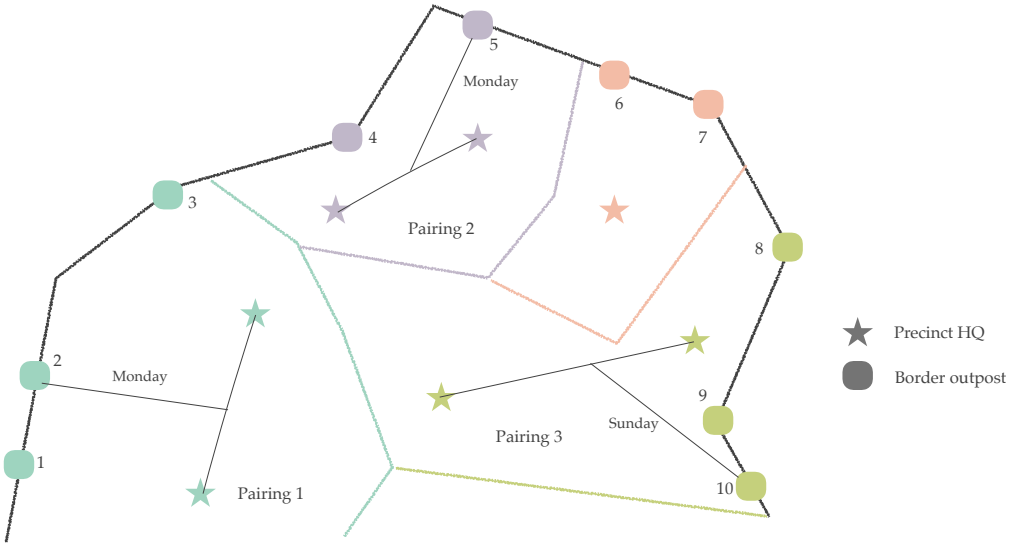


Figure 1.8: Feasible schedule for a week. Elaborated by Carlos Casorrán.

Pairing\Outpost	1	2	3	4	5	6	7	8	9	10
Pairing 1		M								
Pairing 2					M					
Pairing 3										Su

Table 1.1: Tabular representation for the feasible schedule in Figure 1.8.

and E are the edges that represent valid pairing of precincts. Further, the set of vantage points that need to be protected, corresponds to the set of targets J. Furthermore, the vantage points are partitioned among the different vertices of G, such that for a given $u \in V$, J_u contains all the vantage points inside precinct u. The set of attacker types is given by $K = \{\text{Drugs, Contraband, Illegal entry}\}$. In this setting, the pairings among precincts is fixed at the beginning of each month. Therefore, the game is separable into different standard SSG within every pair of paired precincts and one can use a standard SSG formulation such as the one presented in [37] to solve the different subproblems. Within each subproblem, the defender has a single resource to allocate to one of the different vantage points on a given day of the week. Given a coverage strategy over the targets, an adversary of type $k \in K$ plays the game with probability π^k and tries to cross the border through the vantage point $j \in J$ and on the day of the week that maximizes his payoff. It remains to construct the payoffs of the game for the problem described. To that end, Carabineros supplied us with arrest data in the region between 1999 and 2013 as well as other relevant data discussed next. In the following section, we discuss a payoff generation methodology.

1.4.1. Payoff estimation

An accurate estimation of the payoffs for the players is one of the most crucial factors in building a Stackelberg model to solve a real-life problem. For each target in the game, we need to estimate 12 different values corresponding to a reward and penalty for Carabineros and the attacker for each type of crime $k \in K$.

We tackle this problem in several steps. First, we use QGIS [51], an open source geographic information system, to determine what we call action areas around each vantage point provided by Carabineros, based on the visibility range from each outpost. Such an action area represents the range of a detail stationed at a vantage point, *i.e.*, the area within which the detail will be able to observe and intercept a criminal.

Further, consider, for each type of crime $k \in K$, a network $\mathcal{G}^k(\mathcal{V}^k, \mathcal{E}^k)$ that models that type of crime's flow from some nodes outside the border to some nodes inside the border, crossing the border precisely through the action areas previously defined. As nodes of origin for the different types of crime, we consider cross border cities. As destination nodes we consider the locations inside Chile where Carabineros has performed an arrest of that type of crime. In order to have a more manageable sized network, we consider a clustering of these destination nodes. We later show that our methodology is robust versus changes in the number of cluster nodes.

Specifically, for a crime of type $k \in K$, let us define $S^k \subset \mathcal{V}^k$ as the nodes of origin situated outside the borders, $F^k \subset \mathcal{V}^k$ as the nodes of destination and J as the set of action areas along the border. Each destination node, $f \in F^k$, resulting from a clustering procedure is then assigned a demand $b(f)$ which corresponds to the number of destination nodes which are contracted into f. For each $k \in K$, the edge set $\mathcal{E}^k$ is constructed as follows. All nodes of origin are linked to all action areas. These areas are then linked to all of the destination nodes for crime $k \in K$. Figure 1.9 is a representation of such a network. The nodes to the left represent the points of origin of crime and the three nodes to the right are clusters of destination nodes for those crime flows. Note that crime enters the country through the four action areas marked as squares along the border. We propose the following attractivity parameter for a given action area $j \in J$ for a criminal of type $k \in K$ attempting to move from node $s \in S^k$ to node $f \in F^k$ through action area j:

$$U_{sf}^j = \frac{\text{Kilometers of roads inside action area j}}{d_{sj} + d_{jf}},$$

where d_{uv} is the distance in kilometers between nodes $u \in \mathcal{V}^k$ and $v \in \mathcal{V}^k$. This attractivity parameter is proportional to the total length of roads inside a given action area and it is inversely proportional to how much an attacker moving from s^k to f^k has to travel in order to cross the border through area j.

We model the flow of crime $k \in K$ through a single route from $s \in S^k$ to $f \in F^k$ passing

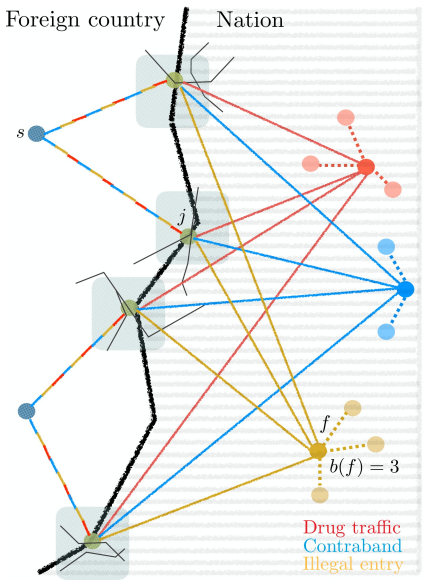


Figure 1.9: Three crime flow networks, one per type of crime. Elaborated by Carlos Casorrán.

through $j \in J$ as follows:

$$x(s,j,f,k) = \frac{e^{\lambda U^j_{sf}}}{\sum_{s' \in S^k} \sum_{j' \in J} e^{\lambda U^{j'}_{s'f}}} \cdot b(f).$$

The flow of crime $k \in K$ through a route (s,j,f) is expressed as a proportion with respect to the flow of crime $k \in K$ through all routes leading into the same destination point $f \in F^k$. The parameter $\lambda \in \mathbb{R}_+$ provides a proxy of how the defender expects crime to behave. A value of $\lambda = 0$ means that crime $k \in K$ between any node of origin and destination distributes itself evenly among the different action areas. A high value of λ, however, is consistent with a flow of that type of crime through those action areas $j \in J$ with a higher attractivity parameter U^j_{sf}. It follows that the total flow of crime of type $k \in K$ through $j \in J$ can be computed by summing over all origin nodes $s \in S^k$ and all destination nodes $f \in F^k$:

$$x(j,k) = \sum_{s \in S^k} \sum_{f \in F^k} \frac{e^{\lambda U^j_{sf}}}{\sum_{s' \in S^k} \sum_{j' \in J} e^{\lambda U^{j'}_{s'f}}} \cdot b(f) \quad \forall j \in J, \forall k \in K.$$

Based on this parameter, we propose the following values for the players' payoff values:

$$\begin{aligned}
A^k(j|u) &= x(j,k) \cdot AG(k) && \forall j \in J, \forall k \in K, \\
A^k(j|c) &= -x(j,k) \cdot OC(k) && \forall j \in J, \forall k \in K, \\
D^k(j|c) &= 0 && \forall j \in J, \forall k \in K, \\
D^k(j|u) &= -x(j,k) \cdot AG(k) && \forall j \in J, \forall k \in K,
\end{aligned}$$

where $AG(k)$ denotes the average gain of successfully committing crime $k \in K$, and $OC(k)$ the opportunity cost of being captured while attempting to perpetrate a crime $k \in K$. Note that the reward Carabineros perceives when capturing a criminal is 0, irrespective of the crime. Carabineros is only penalized when a crime is successfully perpetrated on their watch.

These values were calculated following open source references [16], [1], [42] and where then fully vetted by Carabineros to ensure that our estimates were realistic.

1.4.2. Building software for Carabineros

We have provided Carabineros with a graphical user interface developed in PHP to determine optimal weekly schedules for the night shift actions for a set of border precincts in the XV region of Chile. The software provided for Carabineros is divided into two parts: a first part devoted to the parameter generation of the game according to the indications of the previous section, and a second part, which solves for the optimal deployment of resources. We discuss the two parts separately.

Parameter estimation software

The objective of the parameter estimation software is to construct the payoff matrices for the SSG. This software allows for the matrices to be updated when new criminal arrests are recorded in Carabineros' database. The input for this software is a *csv* data file with arrest data which is uploaded to the software. The main screen of the software shows a map of the region to the left and the following options to the right:

1. *Crimes*: Shows all criminal arrests in the area, color-coded according to the type of crime.
2. *Nodes of origin*: Shows the nodes of origin used in the networks constructed to determine the crime flow through the action areas.
3. *Cluster*: Clusters the criminal arrest points and constructs the crime flow networks joining nodes of origin, action areas and the clustered arrest points, which are the destination nodes for the different types of crime. It displays the payoff matrices for the different action areas.
4. *Input file and update*: Allows to upload a *csv* data file with arrest data. One then reclusters to obtain new destination points and to construct the new crime flow networks that lead to new payoff matrices.

Deployment generation software

The deployment generation software is the part of the software that optimizes the SSGs and returns an implementable patrols strategy for Carabineros. The user is faced to a screen that on the left shows a map of the region where the different action areas are color-coded along the border, and on the right shows different available user options. Clicking on an action area reveals the payoff values for that area. The values can be modified on-screen although this is discouraged. The user can additionally select the number of resources in a given paired pair of precincts. Increasing the number of resources can be used to model that a joint detail can perform a night-shift patrol as many times during a week as the number of

resources he has. Further, the user can select the number of weekly schedules that are to be sampled from the optimal target coverage distribution, allowing him to change the weekly schedule to a monthly schedule. Once all parameters are set, clicking on solve returns the desired patrol schedule such as the one shown in Figure 1.1.

Once a patrol strategy has been returned, the user can perform several actions. If the patrol is not to the planner's liking, he can re-sample based on the optimal coverage distribution returned by the optimization. This produces a different patrol strategy that still complies with the same coverage distribution over targets. The user can further impose different types of constraints on each paired pair of precincts to model different requirements such as forcing a deployment on a given day of the week or to a particular target. Similarly, the user can forbid a deployment on a given day of the week, or forbid deployment to a given target. Further, the user can ensure that at least one of a subset of targets is protected or that deployment to a given target happens on at least one out of a subset of days. Solving the game under these constraints and sampling will produce a deployment strategy that complies with the user's requirements.

1.4.3. Robustness of our approach

We study the robustness of the solutions produced by our software to variations in the payoff matrices. Specifically, we study the robustness of our method against variations of two key parameters in the payoff generation methodology: λ, which models the defender's belief on how crime flows across the border and $b(f)$, which indicates the number of nodes clustered into a given destination node f. Equivalently, one can consider variations in a vector $h = (h_1, h_2, h_3)$ which determines the number of cluster nodes for the three types of crime considered. We study the effects of variations in the parameter λ and in the vector of cluster nodes h separately.

As a base case, we generate payoffs for the players by setting $\overline{\lambda} = 50$ and $\overline{h} = (6, 6, 6)$. This appears reasonable given the size of the problem and distribution and number of arrests per type of crime in the studied region. Let $\lambda \in \Lambda = \{0{,}5\overline{\lambda}, 0{,}75\overline{\lambda}, 1{,}25\overline{\lambda}, 1{,}5\overline{\lambda}\}$ and $h \in H = \{(h_1, h_2, h_3) \in \mathbb{N}^3 : h_t = \overline{h}_t \pm s,\ t \in \{1, 2, 3\},\ s \in \{0, 1, 2, 3\}\}$. We denote by $c(\lambda, h)$, the optimal coverage probabilities on the targets when the payoffs have been defined according to λ and h. Given two vectors $p, q \in \mathbb{R}_+^{|J|}$, we consider the usual euclidean distance function between them:

$$d(p, q) = \sqrt{\sum_{j \in J} (p_j - q_j)^2}.$$

We identify $\lambda^* \in \operatorname{argmax}\{d(c(\overline{\lambda}, \overline{h}), c(\lambda, \overline{h}))\}$ and $h^* \in \operatorname{argmax}\{d(c(\overline{\lambda}, \overline{h}), c(\overline{\lambda}, h))\}$ and plot $c(\overline{\lambda}, \overline{h})$, $c(\lambda^*, \overline{h})$ and $c(\overline{\lambda}, h^*)$.

Figure 1.10 shows the optimal coverage probabilities $c(\overline{\lambda}, \overline{h})$, $c(\lambda^*, \overline{h})$ and $c(\overline{\lambda}, h^*)$ for a game with five paired police precincts and twenty targets. One can see that the optimal probabilities are very robust towards variations in the number of clusters. As one could expect, they are less robust to variations in the parameter λ. Recall that a low value of λ constructs the payoff matrices under the assumption that crime distributes itself uniformly

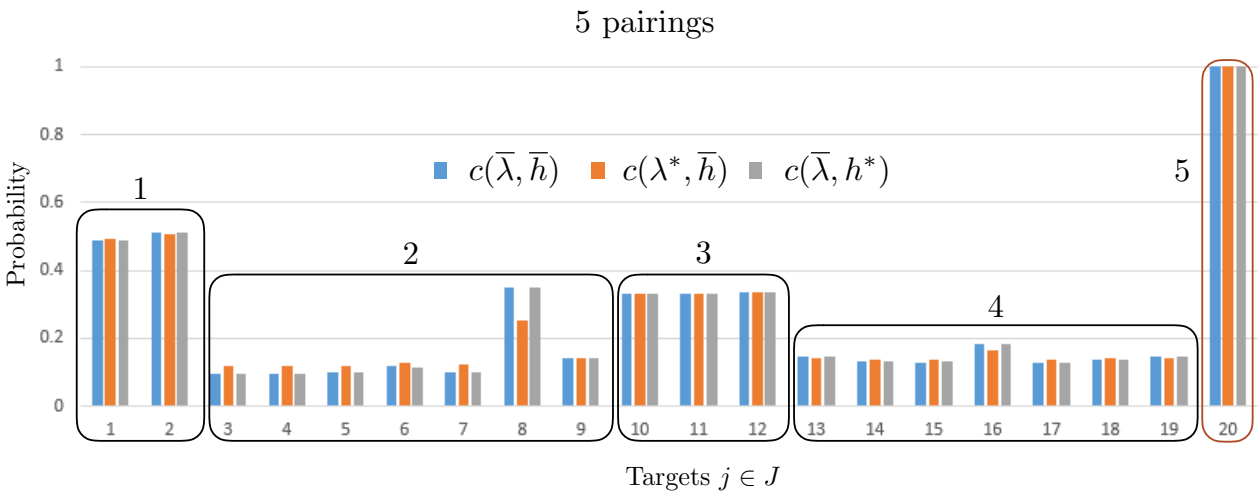


Figure 1.10: Robustness of the solution method to variations in the parameters λ and h

among the different action areas $j \in J$. It is therefore understandable that the optimal coverage probabilities reflect this by trying to cover the targets uniformly. On the other hand the optimal coverage probabilities tend to be more robust for higher values of λ.

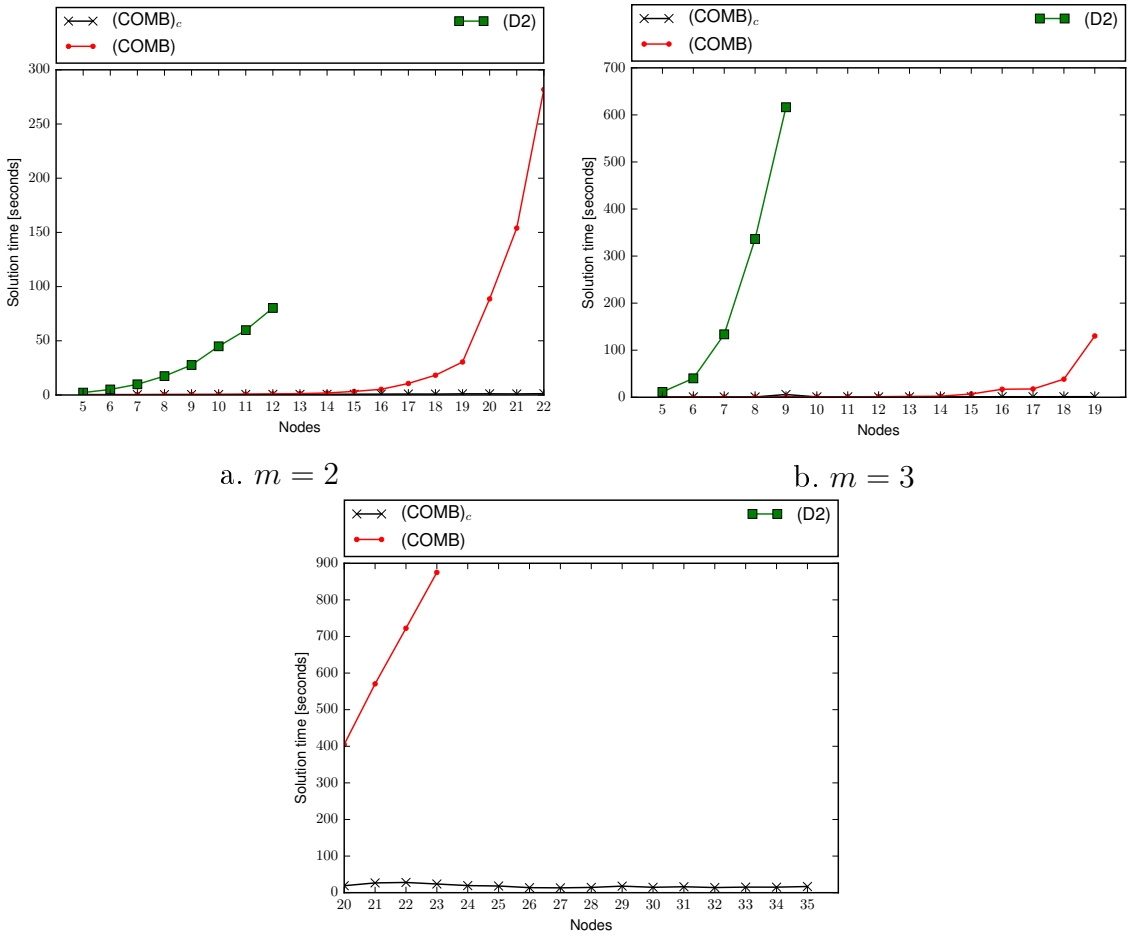
1.5. Computational Experiments

In this section we measure the performance of the compact formulation (COMB) and we study the effectiveness of the alternative sampling procedure presented in Section 1.3.3. The experiments were programmed in Python 3.5 and GUROBI to solve mixed integer programs.

1.5.1. Performance of (COMB)

We present an analysis on the computational efficiency of our compact formulation (COMB) on randomly generated instances. We do so by comparing the solving time of three different methods. The first method is solving the instances directly with (COMB) without separating the exponential family of odd-set constraints (1.28). The second method is solving the instances with (COMB) where we apply the separation procedure described in Section 1.3.2 to the family of odd-set constraints (1.28). We refer to this method as (COMB_C), where the C indicates that we add cuts on the fly. Finally, the third method is solving the instances with (D2) where we explicitly enumerate all leader strategies.
The instances we compare the three methods on are randomly generated as follows. We consider random adjacency graphs where the number of nodes lies in $n = \{5, 6, \ldots, 100\}$ and we generate arcs such that we ensure that the graph is connected and that in average each node has four incident edges. Further, we consider four targets inside each node. In addition, we uniformly generate payoff values for the defender and each attacker type by considering, for each player, rewards $D^k(j|c)$ and $A^k(j|u)$ for all $k \in K$ and $j \in J$ in the range $[0, 100]$ and penalties $D^k(j|u)$ and $A^k(j|c)$ for all $k \in K$ and $j \in J$ in the range [-100,0].

Figure 1.11 a. shows the solving time of the three methods on instances from 5 nodes to 22 nodes. We solve these instances for $m = 2$, *i.e.*, we need four nodes to pair up. For each instance size we plot the average solving time of 30 randomly generated instances. Note that for $m = 2$ only can scale-up until 12 nodes. For greater set of nodes, the process of enumerating the set of pure strategies is not possible for the computational capacity. Figure 1.11 b. shows the solving time of the three methods on instances from 6 to 20 nodes. In this case we solve for $m = 3$. For each instance size we plot the average solving time of 30 randomly generated instances. Note that in this case, (D2) only can solve instances up to 10 nodes. In Figure 1.11 c. we report average solution times of 30 randomly generated instances with $m = 10$, and set of nodes in $\{20, \ldots, 35\}$. In this case, our formulation without the cut-generation procedure only can scale-up until instances of 24 nodes. On the other hand, (COMB_c) does not exceed 20 seconds of solution times.



a. $m = 2$

b. $m = 3$

c. $m = 10$

Figure 1.11: Performance for $m = 2$, $m = 3$ and $m = 10$: Solving time (s.) vs. number of nodes in graph.

As one can see, our compact formulation plus the separation procedure for the odd-set inequalities clearly outperforms the other two methods. The slowest of the methods, as could have been expected, is (D2). The set of leader strategies grows exponentially and (D2) can only explicitly enumerate these strategies for very small graphs of less than 12 nodes. It is also

interesting to note that our full compact formulation, (COMB), behaves remarkably well for graphs of up to 18 nodes. For larger graphs, it is essential to separate the exponential family of odd-set inequalities. Combining this separation procedure with our compact formulation leads to substantial improvements in terms of solving time, consistently solving each of our random instances in less than 20 seconds.

1.5.2. Performance of the alternative sampling method

In Section 1.3.3, we provide a two-stage sampling method to recover an implementable defender strategy, which consists in selecting a matching of size m and then determining the m targets to protect within those paired nodes. In this section we use the Kullback-Leibler distance [38] to determine the similarity between the optimal z^* and our estimated $\hat{z}$. The optimal z^* are those returned by our optimization model (COMB) whereas the estimated $\hat{z}$ are obtained as follows: We sample $i = 1, \ldots, N$ matchings of size m according to the method in Section 1.3.3. Then, for each edge $e \in E$, set $\hat{z}_e = \sum_{i=1}^{N} z_e^i / N$. In our experiments we consider $N = 1000$ and use the Kullback-Leibler distance to measure the similarity of z^* and $\hat{z}$ over instances with n nodes where $n = \{5, 25, 50, 100\}$. For each instance size, we generate 30 estimated $\hat{z}$ and plot the results as box diagrams as shown in Figure 1.12.

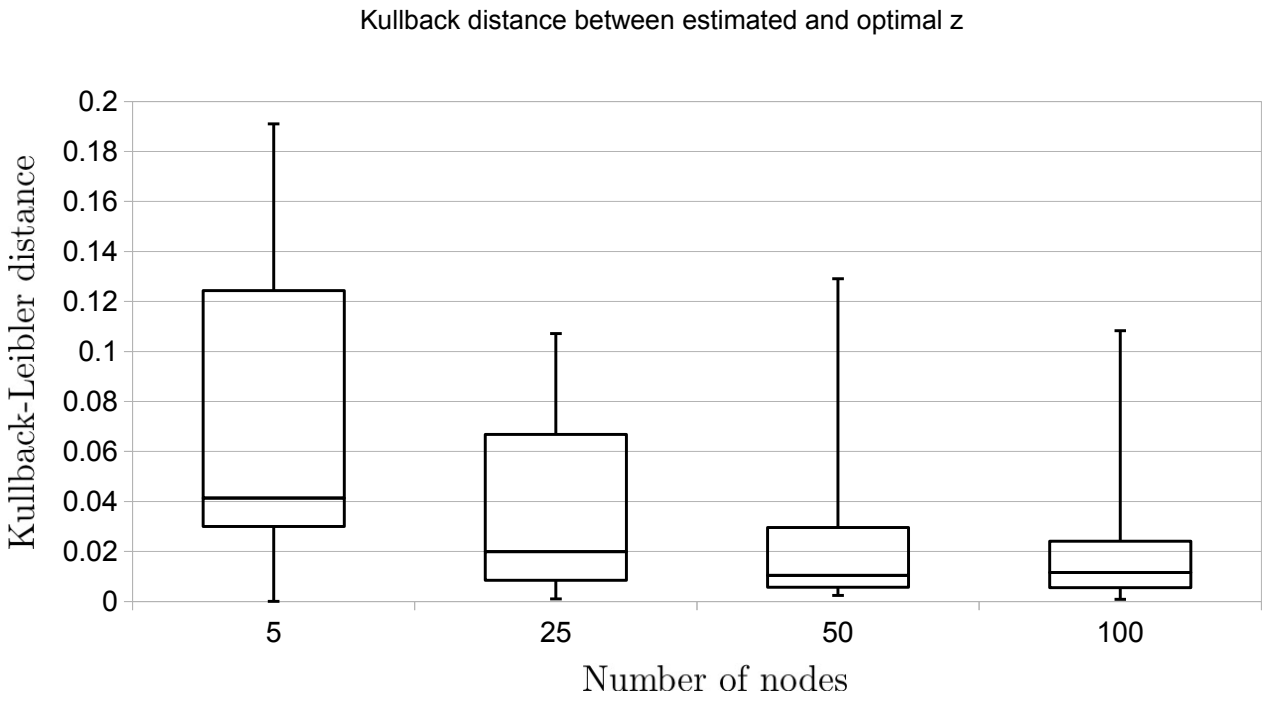


Figure 1.12: Kullback-Leibler distance between z^* and $\hat{z}$ over instances of different size

Observe that the Kullback-Leibler distance between z^* and $\hat{z}$ is very small, below 0,2 over all instances, which is a good indicator that $\hat{z}$ is a good estimator for z^*. in particular we observe that the larger the set of nodes in an instance the better an estimator $\hat{z}$ appears to be. As one can see in Figure 1.12, for instances with 100 nodes, most of the $\hat{z}$ have a Kullback-Leibler distance to z^* which is below 0,02.

1.6. Conclusions and future work

In this chapter, we studied a special type of SSG played on a network. In this game, a defender has to combine resources to do patrol labors in a set of targets. We proposed a novel formulation, (COMB), that represent the set of mixed strategies in a compact form via marginal probabilities in the pairs of resources to team-up and the targets to cove. We proposed a method to retrieve an implementable strategy given the optimal marginals and we prove the validity of our formulation. In this method we proposed two ways to decompose fractional size constrained matchings as convex combination of pure matchings. (COMB) formulation has a exponential family of constraints. To scale-up the instances that solvers are able to optimize, we proposed a cut generation algorithm.

We have further provided an alternative sampling method to recover an implementable defender strategy given the optimal coverage distributions. Computational tests have shown that the two-stage sampling method we describe, provides implementable strategies that do not deviate much from the optimal coverage distributions. Further computational tests have shown (COMB) to have smaller solution times and better scaling capabilities than the extensive formulation (D2) on randomly generated security instances. Both methods are considerably outperformed by the cut generation algorithm.

In addition, we have described a real-life border patrol problem and have presented a parameter generation methodology that takes into account past crime data and geographical and social factors to construct payoffs for the Stackelberg game. Robustness tests have shown that the solutions our software provides are fairly robust to the networks we generate as well as to minor changes in the flow of crime along the border.

In future work we aim to extend our model and methodology to the case where each resource can be paired up with more than one resource, taking into account different capabilities, time schedules, and other considerations. We also are working in improving the parameter estimation for the real case exposed in this chapter.

Chapter 2

Abstractions to handle large scale models.

Besides the Stackelberg Security Games disscused in Chapter 1, data-driven models have been developed to generate strategies against crime. An algorithm based on machine learning techniques is presented in [74] to handle opportunistic crime. This algorithm consists in learning through a Dynamic Bayesian Network (DBN), where the data is the defender strategy in target i at time t, denoted by D_{it}, Y_{it} the number of crimes committed at time t in target i, and X_{it} is a hidden variable representing the number of criminals in target i at time t.

The main idea of the algorithm is to learn the transition probabilities of the number of crimininals in target i at time $t+1$, noted by $\mathbb{P}(X_{i,t+1}|D_t, X_t)$ and the crime output probability $\mathbb{P}(Y_{it}|D_t, X_t)$. The output of this learning phase is used to get a planning strategy, stated as an optimization problem. Experiments shows that this algorithm does not converge for instances of more than $n = 5$ targets in a reasonable time. We call n as the scalability parameter.

In this chapter we use the concept of abstraction to transform large scale urban area problem into a smaller abstract problem and then apply the algorithm mentioned before and solve it hierarchically. Here we use the concept abstraction as in the Artificial Intelligence literature, where we collect many objects with similar features into a single class [9]. In this particular case, targets that have similar characteristic are merged into aggregated targets. This set of aggregated targets is called the abstract layer while the set of original targets is named the original layer. If we are able to solve the problem with the targets aggregated, the algorithm propagates the strategy to the targets inside each cluster of aggregated targets. Given a scalability parameter n of the algorithm that we want to scale-up maybe we need to generate more than two layers. In contrast to traditional clustering methods, in geographical areas the geometry of each cluster is important [13].

Our focus in this work is the methodology to generate those geographical multilayer clusters (or districts) to propagate efficiently mixed strategies from one layer to another, losing the less information as possible.

2.1. Layer Generating Algorithm

We model the layer generation as a Districting Problem [35, 13]. The districting problem is the well known problem of dividing a geographical region into balanced subregions with the notion of balance differing for different applications. For example, police districting problems focus on workload equality [13]. Our layer generation is a districting problem that group targets in the original layer into *aggregated targets* in the abstract layer. However, distinct from the classic Districting problem where the resources are balanced among different *aggregated targets*, in our problem, we try to maximize the similarity of the targets inside the same *aggregated target*. We do so by modeling the similarity of targets within each aggregated target and use this similarity measure as one of the criteria in the optimization formulation of our problem.

When generating the aggregated targets, there are three principles to follow. First, the aggregated targets should follow the geometric constraints in the districting problem such as contiguity, compactness and environmental constraints. Contiguity means that every target is geographically connected; compactness means that all targets in an aggregated target should be close together; and environmental constraints are the constraints for defender's patrol convenience. For example, if two neighboring targets are divided by a highway, they should not be merged together. Second, the dissimilarity within the aggregated targets should be minimized. We consider two properties of target i, the number of crimes per shift with the defender's presence c_1^i and the number without the defender's presence c_0^i. For target i and target j, we define the Dissimilarity distance function as $Dis_{ij} = |c_1^i - c_1^j| + |c_0^i - c_0^j|$.

Third, the algorithm should consider the scalability constraint for a learning algorithm. Let N denote the number of targets in the original layer and n denote the largest scale of problem that the learning and planning algorithms can scale up to. Then there can be no more than n targets inside each *aggregated target* and no more than n *aggregated targets* in the abstract layer. Therefore, $N \leq n^2$ in the original layer. When $N > n^2$, we need multiple layer abstraction that will be introduced later. As we prove next in Lemma 2.1, the more the aggregated targets are in the abstract layer, the less information is lost during the abstraction. Hence, we would want to have as many targets as possible in the abstract layer. Thus, we set n aggregated targets in the abstract layer.

Let $I = \{1, \ldots, N\}$ be a set of targets in the original layer. A partition of size K of this set I is a collection of sets $\{I_k\}_{k=1}^K$ such that $I_k \neq \emptyset$ for all $k \in \{1, \ldots, K\}$, $I_k \cap I_l = \emptyset$ for all $k, l \in \{1, \ldots, K\}, k \neq l$ and $\bigcup_{k=1}^K I_k = I$. $\{I_k\}_{k=1}^K$ is the set of the aggregated targets in the abstract layer. Let $\mathcal{P}_K(I)$ denote the set of all partitions of I of size K. Given $I_k \subset I$ we define its inner Dissimilarity as $Dis(I_k) = \sum_{i,j \in I_k} Dis_{ij} = \sum_{i,j \in I_k} |c_1^i - c_1^j| + |c_0^i - c_0^j|$. Also we define its Inertia as $In(I_k) = \min_{j \in I_k} \sum_{i \in I_k} d_{ij}$, with d_{ij} denoting the physical distance between the geometric centers of targets i, j.

In our districting process we want to find a partition which achieves both low inner Dissimilarity and Inertia over all elements of the partition. Given $\alpha > 0$ as a normalization parameter, we define the information loss function $L_I(K)$ as the lowest cost with a partition of size K, mathematically $L_I(K) = \min_{\{I_k\}_{k=1}^K \in \mathcal{P}_K(I)} \sum_{k=1}^K \alpha In(I_k) + Dis(I_k)$.

Lemma 2.1 The information loss decreases with K, that is $L_I(K+1) \leq L_I(K)$.

PROOF. Assume $K < N$. First, note that $In(\{i\}) = Dis(\{i\}) = 0$ for all $i \in I$. Let j^* be the value of j that achieves the minimum in $In(I_k) = \min_j \sum_{i \in I_k} d_{ij}$. Let $\{I_j^*\}_{j=1}^K$ the optimal partition of I and $k^* \in \text{argmax}_{k=\{1,\cdots,K\}} \alpha In(I_k^*) + Dis(I_k^*)$. Then $|I_{k^*}^*| > 1$ otherwise $L_I(K) = 0$ and $|I_k^*| = 1$ for all k and there is no clusters. Let $i^* \in I_{k^*}^* - \{j^*\}$. Note that:

$$\begin{aligned} In(I_{k^*}^*) &= \sum_{i \in I_{k^*}^*} d_{ij^*} \geq \sum_{i \in I_{k^*}^* - \{i^*\}} d_{ij^*} \geq In(I_{k^*}^* - \{i^*\}) \\ Dis(I_{k^*}^*) &= \sum_{i,j \in I_{k^*}^*} Dis_{ij} \geq \sum_{i,j \in I_{k^*}^* - \{i^*\}} Dis_{ij} = Dis(I_{k^*}^* - \{i^*\}) \end{aligned}$$

Then,

$$\begin{aligned} L_I(K) = & \sum_{k=1}^{K} \alpha In(I_k^*) + Dis(I_k^*) \\ \geq & \sum_{k \neq k^*}^{K} \alpha In(I_k^*) + Dis(I_k^*) \\ & + \alpha In(I_{k^*} - \{i^*\}) + Dis(I_{k^*} - \{i^*\}) \\ & + \underbrace{\alpha In(\{i^*\}) + Dis(\{i^*\})}_{=0} \\ \geq & L_I(K+1) \end{aligned}$$

Then, the partition $\{I_1, \ldots, I_{k^*-1}, I_{k^*} - \{i^*\}, \{i^*\}, I_{k^*+1}, \ldots, I_K\} \in \mathcal{P}_{K+1}(I)$ is feasible for the problem of $K+1$ clusters and has lower loss function value, then the optimal clustering in $\mathcal{P}_{K+1}(I)$ also has the lower objective function. □

Based on these three principles, we propose a mixed integer linear program (MILP) to solve the districting problem. We apply an extension of the capacitated K-median problem with K a parameter such that the set of N targets can be partitioned in K aggregated targets with at most n elements each of them. While the capacitated K-median problem [54] satisfies the scalability constraint by setting a maximum capacity for each aggregated target, it cannot handle the geometric constraints such as contiguity. In this chapter, we handle the geometric constraints by considering the inertia of each aggregated target as part of the information loss function.

Algorithm 4 Constraint Generation Algorithm (I, K)

```
Center ← Location_Problem(I, K); Cuts=∅
for i = 1, ··· , MAX_IT do
    y*, z* ← Allocation_Phase(Center, Cuts)
    i*, j*, k* = argmín_{i,j,k} z*_ik − Dis_ik(y_ij + y_kj − 1)
    if (z*_{i*k*} − Dis_{i*k*}(y_{i*j*} + y_{k*j*} − 1)) ≥ 0 then
        break
    else
        Cuts ← Cuts ∪ {z_{i*k*} ≥ Dis_{i*k*}(y_{i*j*} + y_{k*j*} − 1)}
    end if
end for
return {y*}, objective_function
```

$$
\begin{aligned}
\text{mín}_{x,y,z} \quad & \alpha \textstyle\sum_{i,j} d_{ij} y_{ij} + \sum_{ik} z_{ik} & & \\
s.t. \quad & \textstyle\sum_j y_{ij} = 1 & \forall i \in I & \\
& y_{ij} \leq x_j & \forall i, j \in I & \\
& \textstyle\sum_j x_j = K & & \\
& \textstyle\sum_j y_{ij} \leq n & \forall j \in I & (2.1) \\
& z_{ik} \geq Dis_{ik}(y_{ij} + y_{kj} - 1) & \forall i, k, j \in I & \\
& z_{ik} \geq 0 & \forall i, k \in I & \\
& y_{ij} + y_{kj} \leq 1 & \forall j \in I,\ i, k \text{ cannot be aggregated together} & \\
& y_{ij}, x_j \in \{0, 1\} & \forall i, j \in I &
\end{aligned}
$$

x_j is a binary variable. It is 1 if the target j is the center of an aggregated target and 0 otherwise. The variable y_{ij} takes the value 1 when the target i is allocated to the aggregated target centered in j and 0 otherwise. The variable z_{ik} is a continuous non-negative variable that takes the value Dis_{ik} when target i and target k are allocated to the same aggregated target, otherwise z_{ik} is 0. The objective function is the weighted sum of inertia and dissimilarity. α represents the trade-off between geometric shape and the similarity within each aggregated target.

The first set of constraints ensures that every target is allocated to an aggregated target. The second set of constraints ensures that the center of an aggregated target belongs to this aggregated target. The third expression states that there are K aggregated targets. The fourth set of inequalities ensures the size of every aggregated target to be no greater than n. The fifth and sixth constraint ensures that z_{ik} will take the value Dis_{ik} when target i and target k are allocated to the same aggregated target, otherwise z_{ik} will be 0. The seventh constraint is an example of environmental constraints that target i and target k cannot be in the same aggregated target.

We use the heuristic constraint generation algorithm (Algorithm 4) to approximately solve the problem. The algorithm has two phases: first, the Location_Problem is solved in which

the centers of the aggregated targets is computed. The location problem is solved as a K-median problem. Observe that once the centers are fixed the second and third constraints can be omitted from the MILP, which reduces the size of the problem a lot. In the second phase, sub targets are allocated to the centers obtained in the last step. However, even with the first phase simplification, the resultant MILP is difficult to solve. Thus, in the second phase, given the large number of constraints, we use the constraint generation technique [8] to solve the optimization problem. The iterative constraint generation algorithm is shown as the for loop (line 2-9). To start with, all the constraints $z_{ik} \geq Dis_{ik}(y_{ij} + y_{kj} - 1)$ for i, j, k are removed completely (denoted by the empty set $Cuts$ in line 1), and then in each iteration of the for loop the MILP is solved (line 3) and then we check whether any of the left out constraints are violated (line 4, 5). If yes, then the most violated constraint is added to $Cuts$ or else the loop stops. The maximum number of iterations is limited by MAX_IT. Constraint generation guarantees an optimal solution given large enough MAX_IT.

2.2. Multi-Layer Generating Algorithm

Two layers are not enough to handle instances where the number of targets $N > n^2$. Therefore, we propose the approach to generate K layers in this section. We denote the original layer as Layer 1 and the layer directly generated from Layer k as layer $k+1$. In this notation, the top abstract layer is Layer K. Considering the scale constraints, $K > \log_n N$. When merging targets in layer k to generate targets in layer $k+1$, we also need to figure out the total number of targets in layer $k+1$. In this section, we propose a Dynamic Programming generating algorithm so that the minimum information is lost by abstraction.

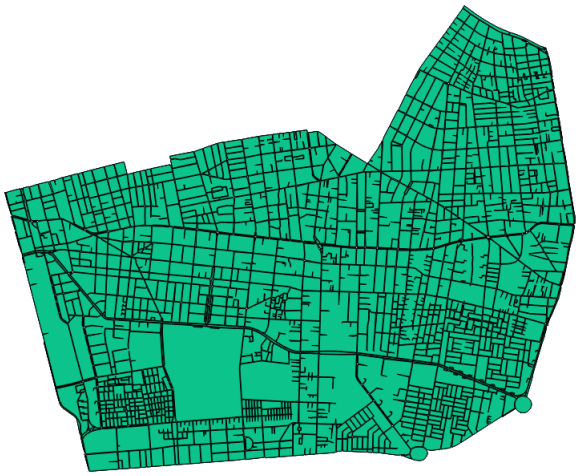
Algorithm 5 Dynamic Programming based Multi-Layer Generating Algorithm

Require: N,n
1: Layers = $\lfloor \log_n N \rfloor + 1$
2: **for** $i = 1, \cdots,$ Layers **do**
3: **for** $j = \lfloor \frac{N}{n^i} \rfloor, \ldots, n^{Layers+1-i}$ **do**
4: optimal_objective(j) = $+\infty$, Targets(j) = $\emptyset$, optimal_path_to(j)= $\emptyset$
5: **for** $k = \lfloor \frac{N}{n^{(i-1)}} \rfloor + 1 : n \cdot j$ **do**
6: I(j), objective(j)= Clustering_Algorithm($I(k), j$)
7: **if** optimal_objective(j) > optimal_objective(k)+objective(j) **then**
8: optimal_objective(j)= optimal_objective(k)+objective(j)
9: optimal_path_to(j) = k
10: Targets(j) = $I(j)$
11: **end if**
12: **end for**
13: **end for**
14: **end for**
15: **return** Targets

This algorithm cannot be solved in a reasonable time for real instances, so for this work

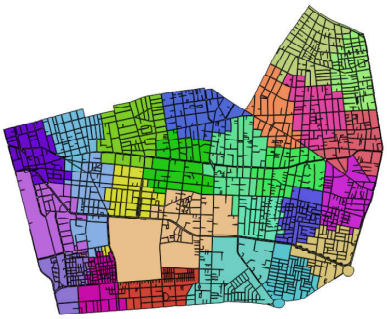
we only use a greedy algorithm that, in each step $i \in \{1, \cdots, \text{Layers}\}$, solves the problem with the maximum number of targets, i.e., $K(i) = n^{Layers+1-i}$.

We show an example where $N = 1266$ and $n = 25$. Then $\lfloor \log_n N \rfloor + 1 = 3$ layers are required in our algorithm. The resulting layers are showed in Figure 2.1.



Original Layer or Layer 1

Layer 2



Layer 3

Figure 2.1: Layers of targets generated by the Greedy Multi-Layer Algorithm algorithm.

2.3. Conclusions

In this chapter we look for efficient ways to approximate algorithms in games developed in large geographical areas inspired by a machine learning algorithm, called DBN, that only scale-up to 5 targets in reasonable time. We propose a territory design problem as K-median with capacity constraints cluster territory units or targets considering geographical aspects and similarity in crime levels. We also design a multi-layer algorithm to do geographical clusters in multiple levels.

The model and algorithm presented in this chapter is quite independent from the algorithm DBN that we aim to scale up. This property can be used for example to scale up any game where the strategies have a relationship with spatial and geographical features, for example in security games in any of their versions that are difficult to solve. In a future work we look for a measure of how much we lose doing this approximation, that is, given an instance and a approximated solution, how far we are to the optimal strategy.

Chapter 3

Solving Stackelberg equilibrium in stochastic games

3.1. Introduction

In this chapter we face the problem of computing a strong Stackelberg equilibrium (SSE) in stochastic games (SG) both in feedback and stationary policies. Given a set of states we model a two player perfect information dynamics where one of them, called *Leader* or player A, observes the current state and commits to a, possibly mixed, strategy f. Then the other player, called *Follower* or player B, observes the strategy of player A and plays his best response denoted by g. Given the selected strategies there is an immediate payoff for each player and a random transition Q^{fg} to another state. This dynamics can be represented by the following scheme:

$$s_0 \rightsquigarrow \begin{array}{c}\text{Player } A\\ \text{chooses } f_0\end{array} \rightsquigarrow \begin{array}{c}\text{Player } B\\ \text{observes } f_0\\ \text{and chooses } g_0\end{array} \underbrace{\rightsquigarrow}_{Q^{f_0 g_0}(s_1|s_0)} s_1 \rightsquigarrow \begin{array}{c}\text{Player } A\\ \text{chooses } f_1\end{array} \rightsquigarrow \begin{array}{c}\text{Player } B\\ \text{observes } f_1\\ \text{and chooses } g_1\end{array} \rightsquigarrow \cdots$$

Formally, let $\mathcal{S}$ represent the set of states of the game. Let $\mathcal{A}, \mathcal{B}$ denote the set of actions of players A and B respectively, and we denote by $\mathcal{A}_s$ and $\mathcal{B}_s$ the actions available in state $s \in \mathcal{S}$. For a given state $s \in \mathcal{S}$ and actions $a \in \mathcal{A}_s$ and $b \in \mathcal{B}_s$, $Q^{ab}(s'|s)$ represents the transition probabilities of reaching the state $s' \in \mathcal{S}$. The time horizon is given by $\tau \in \mathbb{N} \cup \{+\infty\}$. When $\tau < +\infty$ we say that the game has finite horizon and otherwise we say this is an infinite horizon game. The reward received by each player in state s when selecting actions $a \in \mathcal{A}_s$ and $b \in \mathcal{B}_s$ is referred to as the one-step reward functions and are given by $r_A = r_A^{ab}(s)$ and $r_B = r_B^{ab}(s)$. The constants $\beta_A, \beta_B \in [0, 1)$ are discount factors for player A and B respectively. Therefore we represent a two-person stochastic discrete game $\mathcal{G}$ by

$$\mathcal{G} = (\mathcal{S}, \mathcal{A}, \mathcal{B}, Q, r_A, r_B, \beta_A, \beta_B, \tau) \ .$$

SG have been used to model situations arising in areas like evolutionary biology, interaction in economics [4], computer networks [3], and security [21], among others applications.

In the literature of controlled stochastic processes authors distinguish different types of policies, see [5, ch. 2]. In this chapter we shall work with Feedback policies, which depends on the current state s and time epoch t and Stationary policies defined as feedback policies that do not depend on the time step. There are other type of policies that depend on the whole history of the game called closed-loop policies, or policies that only depend on the initial state of the game referred to as open loop policies.

Recent work on computing Stackelberg equilibria in SGs include [21], which introduced a model and algorithm to do resource allocation to detect fare evasion, finding Bayesian Stackelberg equilibrium and [62], which uses a non linear optimization model to find Stackelberg equilibrium in adversarial patrolling games. To the best of our knowledge there is no prior work on efficient algorithms to find stationary strategies in Stackelberg games when $\tau \to +\infty$, in case they exist.

In the security applications that motivate this work, such as patrolling the streets to prevent crime, the decision of where to patrol next should not depend on the history of previous patrolling actions and a time independent policy is easy to communicate to real world security agents. Therefore, the focus of this work is computing Stackelberg equilibria in feedback or Markovian policies. Nevertheless, stationary policies are not always optimal (see the counterexample in the corrected version of [62]).

For the Leader in a Stackelberg dynamic setting, feedback policies can be seen as a collection of functions $\pi = \{f_0, f_1, \ldots, f_\tau\}$, where for each $t \in \{0, \ldots, \tau\}$ the function $f_t : \mathcal{S} \to \mathbb{P}(\mathcal{A}_s)$ maps a state $s \in \mathcal{S}$ into a probability distribution over the set of actions $\mathcal{A}_s$, denoted by $f_t(s)$. The follower feedback policies will be a collection of functions $\gamma = \{g_0, g_1, \ldots, g_\tau\}$ where each function g_t is a mapping from the set of states $\mathcal{S}$ into $\mathbb{P}(\mathcal{B}_s)$. The sets of all possible feedback strategies are denoted by Π and Γ, for player A and B respectively. With this notation, stationary policies are $\pi = \{f, f, f, \ldots\}$, $\gamma = \{g, g, \ldots\}$, for the leader and follower respectively. In particular, we are interested in computing SSE policies. That is, a Stackelberg equilibrium where if the follower is indifferent between actions, the strategy that leads to the best utility for the leader is selected.

To determine the performance of a pair of policies we write down the discounted reward for each player. To do so, we notice that given an initial state $s \in \mathcal{S}$ and a pair of strategies (π, γ) there exists a unique probability measure $\mathbb{P}_s^{\pi,\gamma}$ for the stochastic processes $\{S_t\}$, $\{A_t\}$ and $\{B_t\}$ in the spaces $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{B}$ respectively. Let $\mathbb{E}_s^{\pi,\gamma}$ be the expectation operator with respect to $\mathbb{P}_s^{\pi,\gamma}$. See [30, Appendix C]. For any horizon $\tau \in \mathbb{N} \cup \{+\infty\}$, the value functions for both players are given by

$$\begin{aligned} v_A^{\pi,\gamma}(s) &= \mathbb{E}_s^{\pi,\gamma}\left[\sum_{t=0}^{\tau} \beta_A^t r_A^{A_t,B_t}(S_t)\right] \\ v_B^{\pi,\gamma}(s) &= \mathbb{E}_s^{\pi,\gamma}\left[\sum_{t=0}^{\tau} \beta_B^t r_B^{A_t,B_t}(S_t)\right] . \end{aligned}$$

Given a state $s_0 \in \mathcal{S}$, a Stackelberg equilibrium is a pair of policies (π^*, γ^*) satisfying:

$$\pi^* \in \text{argmax}_{\pi \in \Pi} \quad v_A^{\pi, \gamma^*(\pi)}(s_0) \tag{3.1}$$

$$s.t. \quad \gamma^* \in \text{argmax}_{\gamma \in \Gamma} v_B^{\pi, \gamma}(s_0) \ , \tag{3.2}$$

where Π and Γ will be the set of feedback or stationary policies depending on the case we are studying. We define a pair of Stackelberg policies as a pair of policies (π, γ), forming a Stackelberg equilibrium. Notice that in this case it would be convenient to write, for a policy π of player A, the corresponding policy γ of player B as $\gamma = \gamma(\pi)$. Nevertheless, unless it is necessary we shall continue writing the pair of Stackelberg strategies as (π, γ) instead of $(\pi, \gamma(\pi))$. Similarly, a pair of Stackelberg stationary strategies is a pair of stationary strategies (f, g) which forms a Stackelberg equilibrium.

We summarize the contributions of this work as follows:

1. We prove via a counterexample that it is not always possible to achieve a SSE in stationary policies via dynamic programming.
2. We provide sufficient conditions for a stochastic game to satisfy existence of a unique SSE in stationary strategies.
3. We provide a theoretical framework for algorithms to compute SSE in feedback strategies in both finite and infinite horizon via a definition of a suitable operator.
4. We prove the convergence of Value Iteration and Policy Iteration to find SSE in stationary policies for a family of stochastic games.
5. We experimentally compare the scalability of mathematical programming approaches and algorithms based on dynamic programming.

The rest of this chapter is as follows: In Section 3.2 we provide a Literature review with emphasis on algorithms and math programming approaches to compute Stackelberg Equilibrium. In Section 3.3 we present an application to a security game in a graph. A numerical example to show how a value function iteration algorithm works in a simple SG it is shown in Section 3.4. In section 3.5 we introduce the Stackelberg Operator and discuss the convergence of Value Iteration (VI), Policy Iteration (PI) and Rolling Horizon (RH) for a particular family of stochastic games. In section 3.6 we extend algorithms and operators to the general case and we discuss why dynamic programming algorithms cannot converge in the general case. In Section 3.7 we provide a mathematical programming approach to compute SSE in stationary policies. In Section 3.8 we present a computational comparison of the performance of the algorithms exposed. Finally we present our conclusions and future work in Section 3.9.

3.2. Literature Review

We divide the literature review in two parts. We first describe general results and applications of SG focusing on algorithms and stationary policies, for both Nash and Stackelberg equilibrium. In the second part, we focus on mathematical programming models that aimed to calculate equilibria in SG.

3.2.1. General results

SG were first introduced by Shapley [53] and who also gives the first algorithm to find Nash Equilibrium in zero-sum SG (when $r_A^{ab}(s) = -r_B^{ab}(s) = r^{ab}(s)$) based on a dynamic programming algorithm. The algorithm strongly relies on the fact that there is a unique value of the game. This is not the case for Nash equilibrium in general sum games. There are some articles that extend Shapley's seminal work to general sum games. In [36] a Value Iteration algorithm is used that converges to a Nash equilibrium when the SG horizon is finite. The algorithm uses an arbitrary "Nash selection function" that maps local choices between multiple Nash equilibrium into the selection of a single global Nash equilibrium. For general-sum games a pair of stationary policies that is a Nash equilibrium does not always exists. In [28], the authors discuss about a 3-person game in which there is no Nash equilibrium in stationary policies. In [75] it is demonstrated that there exist SG for which no value iteration algorithm based on dynamic programming can converge to a stationary policy.

Another approach to find Nash equilibrium uses the Fictitious play paradigm where each player updates their beliefs in a simulated game, (see [65]). A complete review about Nash equilibria computation and Learning in SG can be found in [56].

In [40] the authors study the relationship between Stackelberg strategies and correlated equilibrium in stochastic games.They also show that it is NP-hard to find Stackelberg equilibrium in stochastic games. In contrast to MDP settings, Stackelberg equilibrium in stationary policies can be arbitrarily suboptimal as shown in [63]. They also provide a mixed integer non linear program to compute SSE in general SG when players are restricted to stationary policies. They extend this formulation in [62] to policies that depend on history of bounded length.

Some applications of SG in security can be found in [2], [43] and the solution concept used is the Nash equilibrium in zero-sum games. Authors named these games as Stochastic or Markov security games.

Some applications of Stackelberg equilibrium in SG are the following:

- The problem of coordinating a group of robots for planet exploration is presented in [14]. They model it as a multi-objective stochastic game and the solution concept used is the Stackelberg equilibrium.

- Adversarial patrolling games [61] and robotic patrolling games where a robot has to detect an intruder [6].

- Optimal policies to detect fare evasion under execution uncertainty are presented in [21] as solutions of Bayesian Stackelberg SG.

- In [15] the authors present a partially observed stochastic game and define algorithms to generate policies using as an example a manager of a liquid egg production process. The leader aim to maximize expected process productivity while mitigating the risk of an attacker who seeks to contaminate the process with a chemical or biological toxin.

3.2.2. Math programming to solve Stochastic Games

There are well known mathematical programming models that compute equilibria in games. Probably, the most famous case is the linear program to compute the Nash equilibrium in zero-sum games where the duality in linear programming has direct relationship with the calculus of equilibria in zero-sum games (see [47, ch. 12]). There is no linear model to calculate Nash equilibrium in general sum games.

The problem of computing SSE in general sum games can be modeled as a mixed integer linear program, using typical tools of bi-level optimization, as it is mentioned in Chapter 1, with the model (D2).

On the other hand, there are some linear optimization models to find optimal stationary policies in Markov Decisions Process, also referred as MDPs (see [50, ch. 6]). As SG can be seen as a generalization of MDPs when there more than one controller [27], there is the natural question of whether equilibrium policies in SG can also be modeled as linear or mixed integer programs. In [27, ch. 3] the authors name three cases where linear programming can find optimal stationary policies in zero-sum SG:

1. Single-Controller Discounted Games: Games where

$$\text{(SC)} \quad Q^{ab}(s'|s) = Q^{a}(s'|s)$$

   is satisfied, that is only the actions of one player affects the transition probabilities. The model proposed by them is as follows:

$$\begin{aligned}
(\text{FV}_{SC}) \quad & \min \sum_{s\in\mathcal{S}} \alpha_s v(s) \\
& v(s) \geq \sum_{b\in\mathcal{B}} r^{ab}(s) g_{sb} + \beta_A \sum_{s'\in\mathcal{S}} Q^{a}(s'|s) v_A(s') && \forall a \in \mathcal{A},\ \forall s \in \mathcal{S} \\
& \sum_{b\in\mathcal{B}} g_{sb} = 1 && \forall s \in \mathcal{S} \\
& g_{sb} \geq 0 && \forall b \in \mathcal{B},\ \forall s \in \mathcal{S},
\end{aligned}$$

   and its dual:

$$\begin{aligned}
(\text{FV}_{SC})^{*} \quad & \max \sum_{s\in\mathcal{S}} z(s) \\
& \sum_{a\in\mathcal{A}} f_{sa} - \beta \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} Q^{a}(s|s') f_{s'a} = \alpha_s && \forall s \in \mathcal{S} \\
& z(s) \leq \sum_{a\in\mathcal{A}} r^{ab}(s) f_{s,a} && \forall b \in \mathcal{B},\ \forall s \in \mathcal{S} \\
& f_{sa} \geq 0 && \forall a \in \mathcal{A},\ \forall s \in \mathcal{S}.
\end{aligned}$$

The optimal stationary policies can be retrieved setting $f^* = \{f^*(s,a) = \frac{f_{sa}}{\sum_{a' \in \mathcal{A}} f_{sa'}}\}$ to the leader and $g = \{g_{sb}\}$ for player B. The value of the game is given by vector v.

2. Separable Reward State (SER) - Independent Transition Discounted Stochastic games (SIT): This game satisfies the following conditions:

$$\begin{array}{ll} \text{(SER)} & r^{ab}(s) = r_1(s) + r_2(a,b) \\ \text{(SIT)} & Q^{ab}(s'|s) = Q^{ab}(s') \end{array} .$$

The problem can be solved by constructing the following matrix:

$$R = \left[r_2(a,b) + \beta \sum_{s' \in \mathcal{S}} Q^{ab}(s') r_1(s) \right]_{a \in \mathcal{A},\, b \in \mathcal{B}},$$

and calculating the value of this matrix game. The optimal strategies obtained in this matrix game correspond to optimal stationary policies in the (SER)-(SIT) game and the optimal value can be calculated as:

$$v = r_1 + \frac{val(R)}{1-\beta}\mathbf{1}$$

where $val(R)$ is the value of the matrix game R and $\beta = \beta_A = \beta_B$.

3. Switching Controller discounted Stochastic Games: In this type of games the state space is partitioned in S_A and S_B and in each partition one player is a single-controller, that is:

$$\text{(SW)} \quad Q^{ab}(z|s) = \begin{cases} Q^a(z|s) & \text{if } s \in \mathcal{S}_A, \\ Q^b(z|s) & \text{if } s \in \mathcal{S}_B. \end{cases}$$

The authors provide a finite algorithm based on the resolution of multiple Linear programs [64].

Nevertheless, they also provide a counterexample showing that not every stochastic zero-sum game equilibria can be calculated with an LP. Given that Strong Stackelberg Equilibrium and Nash equilibrium are equivalent in zero-sum games (see [17]), the result can be extended to SSE in SG.

In [63] the authors provide Mixed Integer Non Linear Programs (MINLP) to compute general sum Stackelberg equilibria in stationary policies. The model is

$$
\begin{aligned}
\text{(MINLP)} \quad & \text{máx} \quad \sum_{s\in\mathcal{S}} \alpha_s v_A(s) && && (3.3)\\
\text{s.t.} \quad & \sum_{a\in\mathcal{A}} f_{sa} = 1 && \forall s\in\mathcal{S} && (3.4)\\
& \sum_{b\in\mathcal{B}} g_{sb} = 1 && \forall s\in\mathcal{S} && (3.5)\\
& v_A(s) - R_A(s,f,b) \le M_A(1-g_{sb}) && \forall s\in\mathcal{S},\ \forall b\in\mathcal{B} && (3.6)\\
& 0 \le v_B(s) - R_B(s,f,b) \le M_B(1-g_{sb}) && \forall s\in\mathcal{S},\ \forall b\in\mathcal{B} && (3.7)\\
& f_{sa}\ge 0,\ g_{sb}\in\{0,1\} && \forall s\in\mathcal{S},\ a\in\mathcal{A},\ \forall b\in\mathcal{B}, && (3.8)
\end{aligned}
$$

where $R_A(s,f,b) = \sum_{a\in\mathcal{A}} f_{sa}\left[r_A^{ab}(s) + \beta_A \sum_{s'\in\mathcal{S}} Q^{ab}(s'|s)v_A(s')\right]$ and $R_B(s,f,b)$ defined similarly; α_s are positive constants; and M_A, $M_B \gg 1$ are upper bounds to the respective value functions.

Despite their not provide any upper bound to M_A, M_B it suffices to take $M_A = \frac{||r_A||_\infty}{1-\beta_A}$ and $M_B\frac{||r_A||_\infty}{1-\beta_A}$. Variables f_{sa} and g_{sb} represent the probability of taking action a and b respectively in state s, assuming deterministic policies for the follower.

They also provide a linear approximation doing a discretization in the probability space. In [62] the same way of reasoning to an Adversarial Patrolling Game extending (MINLP) to history-dependent policies of fixed length is applied.

In the context of doing surveillance labor on a transit system to detect fare evasion, [21] provide a mixed integer quadratic program for general sum game and a LP in the zero-sum case to find Bayesian Stackelberg Equilibria (the term Bayesian refers to have multiple adversaries, K appearing with a known probability π^k). They model the patrol labor that a set $\{1,\ldots,\gamma\}$ of agents has to do using γ MDPs. The space of states is in the form of (l_i,t) representing the location of agent i and time epoch. Similarly as the game described in Section 3.3, there exists a probability of failing in the actions that each agent takes. Taking s_i^+ and s_i^- as initial and final states for each agent i, the quadratic model can be stated as follows:

$$\text{(DFQ)} \quad \text{máx} \quad \sum_{k\in K} \pi_k x^T U_k^d g_k + \sum_{i=1}^{\gamma} x_i(s_i, a_i, s_i') R_i(s_i, a_i, s_i') \tag{3.9}$$

$$\text{s.t.} \quad x_i(s_i, a_i, s_i') = f^i_{s_i,a_i} Q_i^{a_i}(s_i'|s_i) \qquad \forall s_i \in \mathcal{S} \tag{3.10}$$

$$\sum_{s_i', a_i'} x_i(s_i', a_i', s_i') = \sum_{a_i} f^i_{s_i,a_i} \qquad \forall s_i \in \mathcal{S} \tag{3.11}$$

$$\sum_{a_i'} f^i_{s_i^+,a_i'} = \sum_{s_i',a_i'} x_i(s_i', a_i', s_i^-) = 1 \tag{3.12}$$

$$\sum_{\alpha} g_k(\alpha) = 1 \qquad \forall k \in K \tag{3.13}$$

$$g_k \in \text{argmax}_{\ g_k'} x^T U_k^a g_\lambda \qquad \forall k \in K \tag{3.14}$$

$$x_{sa} \geq 0,\ g_{sb} \in \{0,1\} \qquad \forall s \in \mathcal{S},\ a \in \mathcal{A},\ \forall b \in \mathcal{B}, \tag{3.15}$$

where R represents a trajectory reward which is independent of the interaction with the other players, and U represents the payoff structure due to the interaction with the followers. Variables x and f represents the probability of taking an action and pass through a state to another. Binary variables g_k represent the deterministic best response of each player. In this model rewards are divided in two terms. The first one represents the interaction of a game with payoffs U_k^d and U_k^a. The second term represents the expected utility of doing patrol labor in state s_i take action a_i and arrive to state s_i'. This rewards is given by parameter $R_i(s_i, a_i, s_i')$. Note that the follower response is independent of the future actions, and the transition matrix only depends on the leader decisions. This type of structure as we will see has nice properties in the calculation of SSE.

3.3. Motivational Example

In this section we present a stochastic game example modeling a patrol domain that helps motivate this work. In this game a defender has to patrol a set of locations and an attacker wants to perform an attack in one of these locations, both maximizing their expected rewards. The rewards mainly depend on the place where the attack is performed and whether the location is being covered or not. The effectiveness of the defender's movements are influenced by aleatory factors, so when he decides to patrols any location and try to move to another location, by external factor he could fail, and remain in the same location. We will suppose the attacker always succeeds moving between different locations. In order to represent this situation as a stochastic game in the formalism of Section 3.1, in the following paragraph we define the set of states $\mathcal{S}$, the set of actions for the defender and the attacker $\mathcal{A}$ and $\mathcal{B}$

respectively, the transition probabilities between states Q and the expected rewards for both players r_A and r_B.

Formally, we consider a set of locations to patrol/targets $\mathcal{L} = \{\ell_1, \ell_2, ..., \ell_n\}$. There are some connections allowed between locations represented by edges (denoted by $\mathcal{E}$), so the game can be represented in a graph $(\mathcal{L}, \mathcal{E})$. A typical state $s = (s_A, (s_B, \alpha)) \in \mathcal{S}$ represents the defender's location ($s_A \in \mathcal{L}$), the attacker's location ($s_B \in \mathcal{L}$). The parameter $\alpha \in \{a, \bar{a}\}$ takes the value $\alpha = a$ if the attacker is committing a crime or $\alpha = \bar{a}$ if he is going unnoticed at that period. There are also two special fictitious states $\perp_a$, $\perp_{\bar{a}}$ representing the state of the game once the attack was performed. $\perp_a$ represents the case where the attack is successful and $\perp_{\bar{a}}$ when the attacker is caught.

The action space $\mathcal{A}_s$ for the leader are all the possible location that he can achieve from its current position (given by the state s). For the follower, $\mathcal{B}_s$ represents all the possible locations that the attacker can achieve in the state s with and the decision whether to attack or stay unnoticed. We use the notation $\ell \in \mathcal{L}$ to represent the action of "*move to* ℓ" and $\alpha \in \{a, \bar{a}\}$ to represent the action of *"attacking"* or *"stay unnoticed"* respectively.

Now we can represent the transition probabilities between states taking into account the notation introduced above, as

$$Q^{ab}(z|s) = Q^{s'_A,(s'_B,\alpha')}(z_A, (s'_B, \alpha')|s_A, (s_B, \alpha))),$$

representing the fact that the leader can fail attempting a change in his current position but the attacker has a 100 % of effectiveness moving from one place to another. We also assume independence of the transitions of the defender respect with the attacker's movements.

We define these probabilities noting $q^{\ell'}(\ell''|\ell)$ as the probability of the defender passing from location ℓ to ℓ'' when the defender decides to move to ℓ'. The function q captures the execution uncertainty of the leader. Then Q can be represented as follows:

$$Q^{s'_A,(s'_B,\alpha')}(z_A, (s'_B, \alpha')|s_A, (s_B, \alpha))) = \begin{cases} 1 & s_A = s_B,\ z_A = s'_B = \perp \text{ and } \alpha = a \\ 1 & s_A \neq s_B,\ z_A = s'_B = \perp \text{ and } \alpha = a \\ 1 & s_A = s_B = z_A = s'_B = \perp_a \text{ or } \perp_{\bar{a}} \\ q^{s'_A}(z_A|s_A) & \text{otherwise} \end{cases} \tag{3.16}$$

Expression in its first component represent the fact that if the defender and the attacker meet at same location, and the attack is performed, then the attacker is caught and both go to the final state $\perp_a$. The second row represent the case where the attack succeeds, and both go to the terminal state $\perp_{\bar{a}}$. The third row states that both, $\perp_a$ and $\perp_{\bar{a}}$, are terminal states. The fourth row, represents the uncertainty in the defender movements.

Instant rewards r_A and r_B are defined as the expected values of the rewards R_A and R_B of the dynamics between players and the transportation costs between players, c_A and c_B, which depends on the current state of the system, the actions performed by the players and

the future state of the system. This technique is fairly standard, as it is shown in [50, Ch. 2, pp.20].

The function

$$R_A = R_A^{s'_A,(s'_B,\alpha')}\,(z_A,(s'_B,\alpha')$$

denotes the reward for the leader if the system arrives at state $(z_A,(s'_B,\alpha')$ after defender and attacker choose actions s'_A and (s'_B,α') respectively. Analogously, the function R_B is defined for the follower. The values that R_A and R_B can take are as follows:

$$R_A^{s'_A,(s'_B,\alpha')}((s_A,(s_B,\alpha)),(z_A,(s'_B,\alpha'))) = \begin{cases} -g(s'_B) & z_A \neq s'_B \text{ and } \alpha' = a \\ h(s'_B) & z_A = s'_B \text{ and } \alpha' = a \\ 0 & \alpha' = \bar{a} \\ \varepsilon_A(\perp_a) & z_A = s'_B = \perp_a \\ -\varepsilon_A(\perp_{\bar{a}}) & z_A = s'_B = \perp_{\bar{a}} \end{cases} \tag{3.17}$$

$$R_B^{s'_A,(s'_B,\alpha')}((s_A,(s_B,\alpha)),(z_A,(s'_B,\alpha'))) = \begin{cases} k(s'_B) & z_A \neq s'B \text{ and } \alpha' = a \\ -M(s'_B) & z_A = s'_B \\ -\varepsilon & \alpha' = \bar{a} \\ -\varepsilon_B(\perp_a) & z_A = s'_B = \perp_a \\ \varepsilon_B(\perp_{\bar{a}}) & z_A = s'_B = \perp_{\bar{a}} \end{cases} \tag{3.18}$$

Expressions (3.17) and (3.18) show the payoffs for the leader and the follower for the four possible cases. First, if the attacker succeeds in attacking location s'_B, the attacker receives a payoff given by $k(s'_B)$ which depends on the location where the attack is performed. The defender receives a penalty (negative reward) of $g(s'_B)$. Second, if he attacker is caught by the defender, then the defender receives a payoff of $h(s'_B)$ and the attacker receives a a penalty of $M(s'_B)$. In the third case, we represent the opportunity cost of not attacking in one period as ε. In this case, the defender receives a payoff of 0. Finally, cases fourth and fifth represent the payoff to stay one period caught or *enjoying* a successful attack.

Also, both players incur costs moving between different locations c_A and c_B respectively. Given that the defender transitions are not deterministic, the transportation cost will be evaluated as the expected value. Taking this into account the instantaneous rewards r_A and r_B can be calculated as:

$$\begin{aligned} r_A^{s'_A,(s'_B,\alpha')}(s_A,(s_B,\alpha)) \;:=\; & \sum_{z\in\mathcal{S}} c_A(s_A,z) q^{s'_A}(z|s_A) \\ & + \sum_{z\in\mathcal{L}} R_A^{s'_A,(s'_B,\alpha')}\Big(z,(s'_B,\alpha')\Big)\; Q^{s'_A,(s'_B,\alpha')}\Big(z,(s'_B,\alpha')|s_A,(s_B,\alpha))\Big) \end{aligned}$$

$$r_B^{s'_A,(s'_B,\alpha')}(s_A,(s_B,\alpha)) := c(s_B,s'_B) + \sum_{z\in\mathcal{L}} R_B^{s'_A,(s'_B,\alpha')}\Big(z,(s'_B,\alpha')\Big)\, Q^{s'_A,(s'_B,\alpha')}\Big(z,(s'_B,\alpha')|s_A,(s_B,\alpha)\Big).$$

The dynamics of the game is described as follow:

1. At the start of epoch $t \in \{1,\dots,\tau\}$, the system is in a state formed by the location and the behavior of the attacker.
2. The defender knowing the state chooses a strategy f_t (probably mixed) over the locations reachable from his current location.
3. The attacker observes the strategy and chooses where to move and whether to attack or not. We denote this action as g_t.
4. The system evolves to the following state influenced by f, g and Q. Both players receive their payoffs.
5. The dynamic starts again at a new state s_{t+1} or is finished if $t = \tau$.

This dynamic fits with the games that we are interested to solve.

3.4. Numerical Example 1

We describe a first numerical example with two states and two actions per player in each state in a Stackelberg dynamic in two stages applying a Dynamic Programming approach. We denote $\mathcal{S} = \{s_1, s_2\}$, $\mathcal{A} = \{a_1, a_2\}$, $\mathcal{B} = \{b_1, b_2\}$ the spaces of states and actions for each player respectively. Consider $\beta_A = \beta_B = 0.9$ the discount factors for each player. The transition matrix Q and payoffs r_A, r_B are specified in the following table:

	b_1	b_2
a_1	$(\frac{1}{2},\ \frac{1}{2})$ (10,-10)	(0, 1) $(-5,6)$
a_2	$(\frac{1}{4},\ \frac{3}{4})$ (-8,4)	(1, 0) $(6,-4)$

State s_1

	b_1	b_2
a_1	$(\frac{1}{2},\ \frac{1}{2})$ (7,-5)	(0, 1) $(-1,6)$
a_2	$(\frac{1}{4},\ \frac{3}{4})$ (-3,10)	(1, 0) $(2,-10)$

State s_2

Table 3.1: Transition matrix and payoffs for each player.

where each entry represents of the table in the state s represents each value as follows:

	b
a	$\big(Q^{a,b}(s_1\|s),\ (Q^{a,b}(s_2\|s)\big)$ $(r_A^{a,b}(s), r_B^{a,b}(s))$

We are going to solve this problem using dynamic programming in two stages ($\tau = 2$) to find an SSE, that is, we find two policies $\pi = \{f_1, f_2\}$ and $\gamma = \{g_1, g_2\}$ which forms a SSE. In the last step players are in each state face the following bi-matrices games:

	b_1	b_2
a_1	(10,-10)	$(-5, 6)$
a_2	(-8,4)	$(6, -4)$

State s_1

	b_1	b_2
a_1	(7,-5)	$(-1, 6)$
a_2	$(-3, 10)$	$(2, -10)$

State s_2

Table 3.2: Bi-matrix games in $\tau = 2$.

Let $f_2(s_1) = (x_1, 1-x_1)$ be a randomized policy for the player A, where x_1 the probability if player A chooses action a_1 in the state s_1. Then we can represent the expected payoff for each player if the player B selects an action b_1 or b_2 as follow:

- $\mathbb{E}_{A|b_1}(s_1) = 10x_1 + (-8)(1 - x_1) = 18x_1 - 8$
- $\mathbb{E}_{A|b_2}(s_1) = -5x_1 + 6(1 - x_1) = -11x_1 + 6$
- $\mathbb{E}_{B|b_1}(s_1) = -10x_1 + 4(1 - x_1) = -14x_1 + 4$
- $\mathbb{E}_{B|b_2}(s_1) = 6x_1 + (-4)(1 - x_1) = 18x_1 - 8$

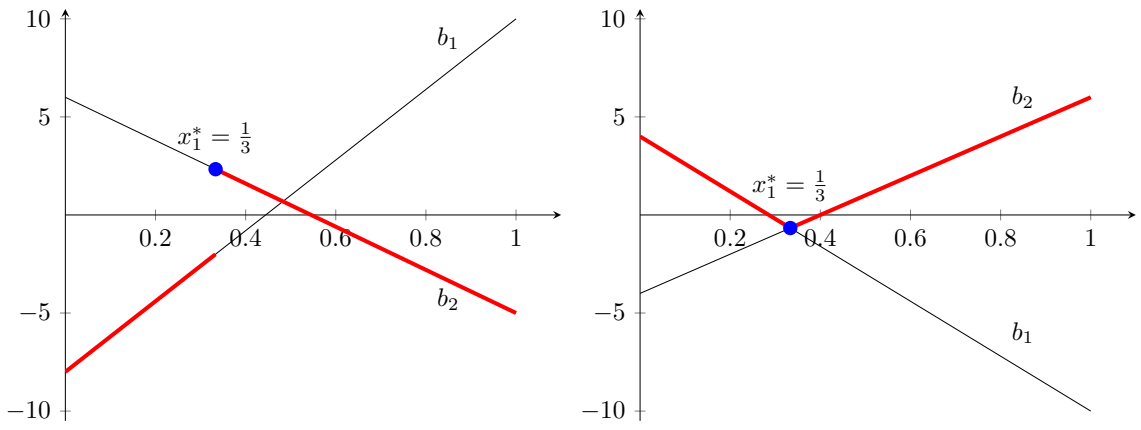
where $\mathbb{E}_{i|b}(s)$ is the expected reward for player $i = A, B$ if the follower choses action b in the state s. In Fig. 3.1 it is shown how a SSE can be found in a $2\times m_B$ bi-matrix game. For the follower, the expected value is plotted in function of x_1. If player A chooses $x_1 < \frac{1}{3}$, then player B will choose b_1 having better expected reward. Analogously, player B will choose b_2 if the player A choose any x_1 greater than $\frac{1}{3}$, and B will have no preference for any action if player A chooses exactly $x_1^* = \frac{1}{3}$. So the expected value for player B taking into account his best response is plotted in red.

On the other hand, player A taking into account the behavior of player B will observe the expected reward function plotted in red, which is not linear and even not continuous. The maximum of this function is in the same 'indifference point' for player B, so the SSE of this game is $f_2(s_1)^* = (\frac{1}{3}, \frac{2}{3})$. Given the indifference of player B and the definition of a SSE, B will choose the action that returns greater expected value to player A, that is b_2, receiving payoffs $v_A^\tau(s_1) = \frac{7}{3}$ and $v_B^\tau(s_1) = -\frac{2}{3}$ respectively.

Similarly, we find a SSE in the state s_2 in the last step. Fig. 3.2 represents the utility function taking into account the best response. The strategies $f^2(s_2)^* = (\frac{20}{31}, \frac{10}{31})$ for player A and $b^* = b_1$ represent a SSE in this stage. The expected values in this last step are $v_A^\tau(s_2) = \frac{107}{31}$ and $v_B^\tau(s_2) = \frac{10}{31}$ respectively.

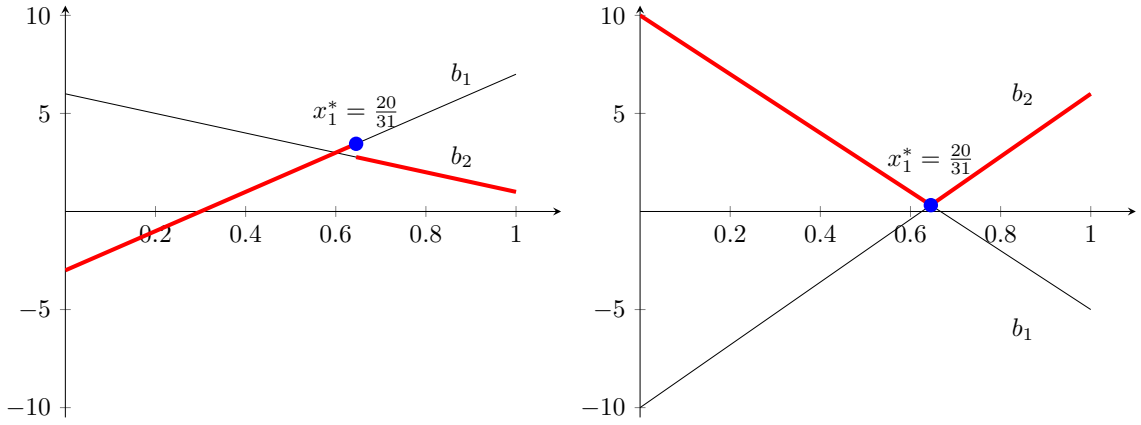
At stage $\tau = 1$, the players knowing the payoffs of the last step can calculate their payoff matrices as the expected value of the instantaneous expected reward added to the value of achieving a specific state. So, for each pair of actions a, b they play a bi-matrix game given by the following payoffs for player i

$$r_i^1(a, b)(s) = r_i^{a,b}(s) + \beta_i \sum_{z \in S} Q^{ab}(z|s) v_i^2(z).$$



Leader Follower

Figure 3.1: Expected reward for each player and SSE in the game played in state s_1 at stage $\tau = 2$.



Leader Follower

Figure 3.2: Expected reward for each player and SSE in the game played in state s_2 at stage $\tau = 2$.

The bi-matrices games that players face in each state are represented in Table 3.3.

	b_1	b_2
a_1	(12.6032, -10.1548)	$(-1.8935,\ 6.2903)$
a_2	(-5.1451, 4.0675)	(8.0999, -4.6)

State s_1

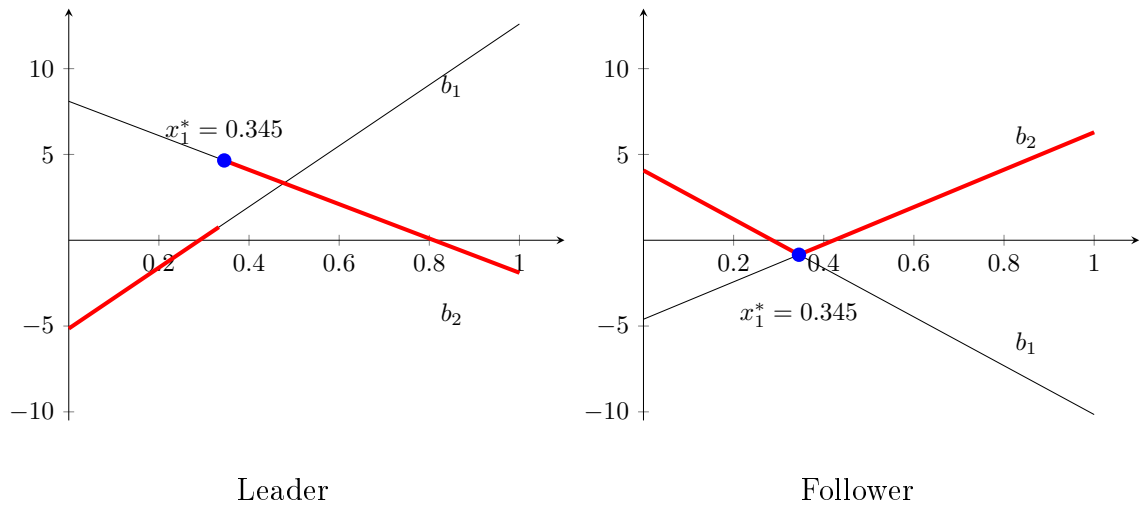
	b_1	b_2
a_1	(9.6032, -5.1548)	(1.0999, 5.4)
a_2	(-0.6484, 9.6226)	$(5.1065,\ -9.7097)$

State s_2

Table 3.3: Bi- matrix games at stage 1.

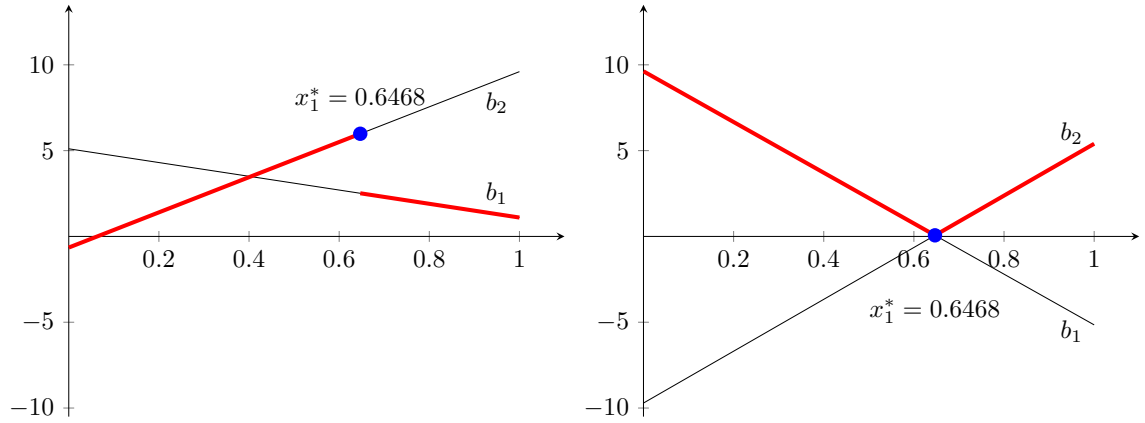
Following the same logic of the games played in stage 2 (see Fig. 3.3 and 3.4), a unique SSE is found at $f^1(s_1)^* = (0.3452, 0.6548)$ with values $v_A^1(s_1) = 4.6507$ and $v_B^1(s_1) = -0.8412$

respectively in state s_1; and $f^1(s_2)^* = (0.6468, 0.3532)$ with values $v_A^1(s_2) = 5.9828$ and $v_B^1(s_2) = 0.0639$ for each player respectively in state s_2



Leader Follower

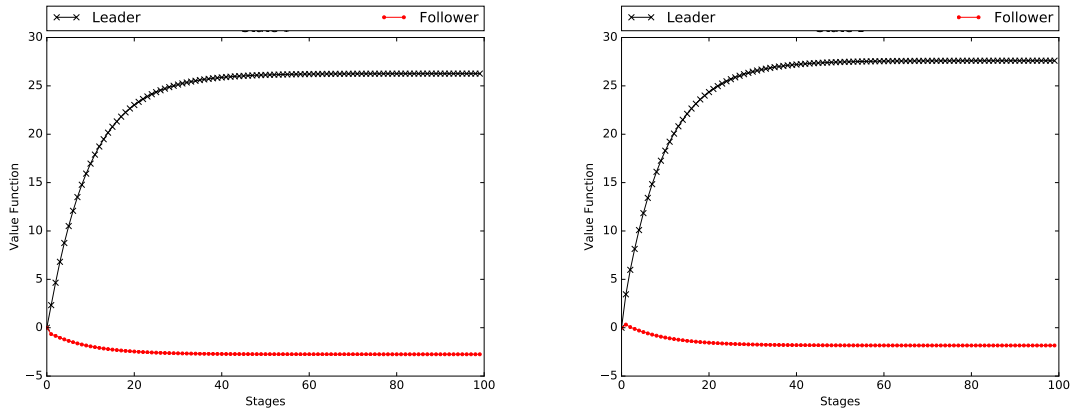
Figure 3.3: Expected reward for each player and SSE in the game played in state s_1 at stage 1.



Leader Follower

Figure 3.4: Expected reward for each player and SSE in the game played in state s_2 at stage 1.

Note that, the policies $\pi^* = \{f^{1*}, f^{2*}\}$ and $\gamma^* = \{b^{1*}, b^{2*}\}$ form a SSE to the SG with horizon $\tau = 2$. Second, if the game has horizon $\tau > 2$, the pair calculated above is exactly the Strong Stackelberg policies corresponding to stages $\tau - 1$ and τ. Third, if the game has a longer horizon ($\tau \gg 1$), the value functions computed through this procedure will stabilize. In Figure 3.5 values are represented for games with $\tau = 100$. Values are stabilized in $v_A^\infty(s_1) = 26.2662$ and $v_A^\infty(s_2) = 27.6012$ for player A and $v_B^\infty(s_1) = -2.7473$ and $v_B^\infty(s_2) = -1.8353$ for player B. We will show in the next sections, that in general it is not always true for all instances.



State s_1 State s_2

Figure 3.5: Value Function vs time in numerical example 1.

3.5. Special case: Myopic Follower Strategies (MFS)

In this section we discuss algorithms for the case where the value function of the follower v_B do not affect in its best response. Let define the functional g of best response as follows:

$$g(f, v_B) = \text{argmax}_{b \in \mathcal{B}} \quad \sum_{a \in \mathcal{A}_s} f(a) \left[r_B^{ab}(s) + \beta_B \sum_{z \in \mathcal{S}} Q^{ab}(z|s) v_B(z) \right] . \tag{3.19}$$

This functional for each policy f of the leader in the state s and value functions v_B, returns the action $b \in \mathcal{B}_s$ that is a best response. For convenience we omit the dependence on s in the policy f. Here we follow the convention that the argmax is unique because in case of indifference between options, we select the one that favors the leader. In case the leader is also indifferent, then, in order for $g(f, v_B)$ to be well defined, the follower selects the action with the lowest index. Note that since $g(f, v_B)$ optimizes (3.19), $g(f, v_B)$ is at least as good as any mixed strategy.

We say that a stochastic game $\mathcal{G}$ has optimal Myopic Follower Strategies (MFS) if at any step of the game the functional g is independent of v_B, that is $g(f, v_B) = g(f)$. That means in other words that optimal responses for the player B are independent of the future values. In particular, we distinguish two important cases of SG with (MFS):

- Myopic follower: We define a game as a myopic follower game if $\beta_B = 0$. Note that in this case the follower at any step of the game does not take into account the future rewards, but only the instantaneous rewards. Clearly, in this case the follower best response is in the form:

$$g(f, v_B) = \arg\max_{b \in \mathcal{B}} \quad \sum_{a \in \mathcal{A}_s} f(a) r_B^{ab}(s) \quad = g(f).$$

- Leader-Controller Discounted Games: This case is a particular case of the Single-controller discounted game described in Section 3.2.2 where the controller is the leader. In other words, the transition law has the form:

$$Q^{ab}(z|s) \;=\; Q^{a}(z|s).$$

In this case best response can be written as:

$$\begin{aligned} g(f, v_B) &= \operatorname{argmax}_{b\in\mathcal{B}} \sum_{a\in\mathcal{A}_s} f(a)\left[r_B^{ab}(s) + \beta_B \sum_{z\in\mathcal{S}} Q^{a}(z|s) v_B(z)\right] \\ &= \operatorname{argmax}_{b\in\mathcal{B}} \sum_{a\in\mathcal{A}_s} f(a) r_B^{ab}(s) + \text{Constant} \\ &= g(f) \quad , \end{aligned}$$

where the Constant term does not depend on b, so the 'argmax' and the best response function does not depend on v_B.

3.5.1. Stackelberg operator and Value function iteration

Let $\mathcal{F}(\mathcal{S})$ be the set of all bounded functions of the finite space state $\mathcal{S}$ into $\mathbb{R}$. Given a SG $\mathcal{G}$ and a strategy f, we define the operator $T_A^f : \mathcal{F}(\mathcal{S}) \to \mathcal{F}(\mathcal{S})$ by the following expression:

$$T_A^f(v_A)(s) \;=\; \sum_{a\in\mathcal{A}_s} f(a)\left[r_A^{ag(f)}(s) + \beta_A \sum_{z\in S} Q^{ag(f)}(z|s) v_A(z)\right]. \tag{3.20}$$

This operator, given a fixed stationary policy of the leader f, computes the value of being in state s and apply this policy. Now we define the Stackelberg operator, $T_A : \mathcal{F}(\mathcal{S}) \to \mathcal{F}(\mathcal{S})$, for the MFS case as follows:

$$\begin{aligned} T_A(v_A)(s) &= \max_{f\in\mathbb{P}(\mathcal{A}_s)} \sum_{a\in\mathcal{A}_s} f(a)\left[r_A^{ag(f)}(s) + \beta_A \sum_{z\in S} Q^{ag(f)}(z|s) v_A(z)\right] \\ &= \max_{f\in\mathbb{P}(\mathcal{A}_s)} T_A^f(v_A)(s). \end{aligned} \tag{3.21}$$

This operator computes in every state the SSE value of being in state s for each $s \in \mathcal{S}$ and the future expected rewards are given by function v_A. Given the value function v_A and a fixed state $s \in \mathcal{S}$ the operator T_A can be computed with (D2) described in Chapter 1, as follows:

$$\text{(D2)} \quad \text{máx} \quad T_A(s) \tag{3.22}$$

$$\text{s.t.} \quad \sum_{a\in\mathcal{A}} f_{sa} = 1 \tag{3.23}$$

$$\sum_{b\in\mathcal{B}} g_{sb} = 1 \tag{3.24}$$

$$T_A(s) \leq \sum_{a\in\mathcal{A}} f_{sa}\left[r_A^{ab}(s) + \beta_A \sum_{z\in\mathcal{S}} Q^{ab}(z|s)v_A(z)\right] + ||r_A||_\infty(1-g_{sb}) \qquad \forall b \in \mathcal{B} \tag{3.25}$$

$$0 \leq T_B(s) - \sum_{a\in\mathcal{A}} f_{sa} r_B^{ab}(s) \leq ||r_B||_\infty(1-g_{sb}) \qquad \forall b \in \mathcal{B} \tag{3.26}$$

$$f_{sa} \geq 0,\ g_{sb} \in \{0,1\} \qquad \forall a \in \mathcal{A},\ \forall b \in \mathcal{B} \tag{3.27}$$

$$T_A(s), T_B(s) \in \mathbb{R} \tag{3.28}$$

where the optimal values of (D2), $T_A(s)$ represent $T_A(v_A)(s)$ for a given state $s \in \mathcal{S}$. Optimal variables $\{f_{sa}\}_{a\in\mathcal{A}}$ and $\{g_{sb}\}_{b\in\mathcal{B}}$ represent a pair of SSE in the state s. Constrains (3.23) and (3.24) define that f and g are probability measures. The objective function and constraint (3.25) states that $T_A(s)$ is the maximum value attainable for the leader considering the optimal response for the follower. The optimality condition for the follower is represented by expression (3.26). Given that variables g_{sb} are binary, the policy implemented for B under this formulation is deterministic.

Given the Stackelberg operator T_A, a generic dynamic programming algorithm is presented in Algorithm 6. This generic algorithm is described as follows:

Algorithm 6 Value function iteration: Finite horizon

1: Initialize with $v_A^{\tau+1}(s) = 0$ for every $s \in \mathcal{S}$
2: **for** $t = \tau, \ldots, 1, 0$, and for every $s \in \mathcal{S}$ **do**
3: Solve

$$v_A^t(s) \;=\; T_A(v_A^{t+1})(s) \tag{3.29}$$

Finding f_t^* and g_t^* SSE strategies at stage $t-1$.
4: **end for**
5: **return** Stackelberg policies $\pi^* = \{f_0^*, \ldots, f_\tau^*\}$ and $\gamma^* = \{g_0^*, \ldots, g_\tau^*\}$

In [5, Theorem 7.4] it is shown that when this algorithm ends, it returns both the value for our game and a pair of Stackelberg feedback policies (π^*, γ^*) for the τ-finite horizon game. In the rest of this subsection, we discuss about the convergence of the sequence of value functions generated for the previous algorithm to the value function (v_A^*) using contraction mapping operators theory and fixed point arguments for the infinite horizon game.

Definition 3.1 *Let $(X, ||\cdot||)$ be a Banach Space (i.e. normed and complete).*

A mapping $\tilde{T} : X \to X$ is said to be contractive, if there exists a positive constant $\beta < 1$, such that

$$||\tilde{T}u - \tilde{T}v|| \leq \beta||u - v|| \ ,$$

for all $u, v \in X$. In this case, the contraction is said to be of modulus β.

In this subsection we work with $X = \mathcal{F}(\mathcal{S})$ and the norm $||v||_\infty = \text{máx}_{s\in\mathcal{S}} |v(s)|$. Given that $(\mathcal{F}(\mathcal{S}), ||v||_\infty)$ is a Banach space, the following theorem states that the operators defined in (3.20) and (3.21) are contractions on $\mathcal{F}(\mathcal{S})$.

Theorem 3.1 Let $\mathcal{G}$ be a SG with MFS, then it is true that:

a) For any stationary strategy f, the operator $T_A^f : \mathcal{F}(\mathcal{S}) \to \mathcal{F}(\mathcal{S})$, defined in (3.20) is a contraction on $(\mathcal{F}(\mathcal{S}), ||\cdot||_\infty)$ of modulus β_A.
b) The operator T_A defined in (3.21) is a contraction on $(\mathcal{F}(\mathcal{S}), ||\cdot||_\infty)$, of modulus β_A.

Proof. To show a), take $v_A, u_A \in \mathcal{F}(\mathcal{S})$, a stationary strategy f, and $s \in \mathcal{S}$. Then,

$$\begin{aligned} T_A^f(v_A)(s) - T_A^f(u_A)(s) &= \beta_A \sum_{a\in\mathcal{A}_s} f(a) \sum_{z\in\mathcal{S}} Q^{ag(f)}(z|s) \left[v_A(z) - u_A(z)\right] \\ &\leq \beta_A \text{max}_{z\in\mathcal{S}} |v_A(z) - u_A(z)|. \end{aligned}$$

By reversing the roles of v_A and u_A and taking the maximum over $s \in \mathcal{S}$ we have that:

$$\begin{aligned} |T_A^f(v_A)(s) - T_A^f(u_A)(s)| &\leq \beta_A \text{max}_{z\in\mathcal{S}} |v_A(z) - u_A(z)| \ , \\ \Rightarrow \ \text{max}_{z\in\mathcal{S}} |T_A^f(v_A)(z) - T_A^f(u_A)(z)| &\leq \beta_A \text{max}_{z\in\mathcal{S}} |v_A(z) - u_A(z)| \ . \end{aligned}$$

Concluding that T_A^f is a contracting map of modulus β_A.

In order to show b), take $v_A, u_A \in \mathcal{F}(\mathcal{S})$, a state $s \in \mathcal{S}$ and f^* the optimal policy of $\text{máx}_{f\in\mathbb{P}(A_s)} T_A^f(v_A)$. Then,

$$\begin{aligned} T_A(v_A)(s) - T_A(u_A)(s) &= \text{max}_{f\in\mathbb{P}(A_s)} T_A^f(v_A) - \text{max}_{f\in\mathbb{P}(A_s)} T_A^f(u_A) \\ &= T_A^{f^*}(v_A) - \text{max}_{f\in\mathbb{P}(A_s)} T_A^f(u_A) \\ &\leq T_A^{f^*}(v_A)(s) - T_A^{f^*}(u_A)(s) \\ &\leq \beta_A \text{max}_{z\in\mathcal{S}} |v_A(z) - u_A(z)|. \end{aligned}$$

Then, by reversing the roles of v_A, u_A and taking the maximum the result follows. □

If T_A is contractive we can use the Banach's fixed point theorem, Theorem 3.2, to conclude that the values converges to the unique fixed point v_A^* of operator T_A.

Theorem 3.2 (*Banach's Fixed Point Theorem*): Let $(X, ||\cdot||)$ a Banach space. If $\tilde{T} : X \to X$ is a contraction mapping, then there exists a unique element $v \in X$ such that

$$\tilde{T}v = v \ .$$

Furthermore, for all $u \in X$,

$$\lim_{n\to\infty} \tilde{T}^n u \to v \ .$$

Given $\varepsilon > 0$, a value function iteration algorithm is proposed in Algorithm 7, that applies T_A repeatedly until the distance between two functions v_A^n and v_A^{n+1} are less or equal to ε. The last two theorems ensure us that the algorithm finishes. Now we use Theorem 3.3 to show that the algorithm converges to the value of the leader in a SSE. Also, this theorem give us a guaranty that at the n-th iteration the values are $\frac{\varepsilon}{1-\beta_A} = \varepsilon'(n)$ far from the equilibrium value.

Algorithm 7 Value function iteration: Infinite horizon

Require: $\varepsilon > 0$

1: Initialize with $n = 1$, $v_A^0(s) = 0$ for every $s \in \mathcal{S}$ and $v_A^1 = T_A(v_A^0)$
2: **while** $||v_A^n - v_A^{n-1}||_\infty > \varepsilon$ **do**
3: Compute v_A^{n+1} by

$$v_A^{n+1}(s) = T_A(v_A^n)(s) \ .$$

Finding f^* and $g^*(f)$ at stage n.
4: $n := n + 1$
5: **end while**
6: **return** Stationary Stackelberg policies $\pi^* = \{f^*, \ldots\}$ and $\gamma^* = \{g^*, \ldots\}$

Theorem 3.3 Let $\mathcal{G}$ be a SG with MFS. Then the sequence of value functions v_A^n converges to v_A^*. Furthermore, v_A^* is the fixed point of T_A, and therefore, for any $n \in \mathbb{N}$,

$$||v_A^* - v_A^n|| \leq \frac{||r_A||_\infty \ \beta_A^n}{1-\beta_A}.$$

Proof. For any pair of policies (π, γ) let $v_{A,\tau}^{\pi,\gamma}$, the value function of A in the problem with horizon τ and the associated policies π y γ, and $v_A^{\pi,\gamma}$ the value function in infinite horizon when the pair (π, γ) is applied.

First we note that for each $s \in \mathcal{S}$,

$$v_A^{\pi,\gamma}(s) - v_{A,\tau}^{\pi,\gamma}(s) \ = \ \mathbb{E}_s^{\pi,\gamma}\left[\sum_{t=\tau+1}^{\infty} \beta_A^t r_A^{A_t,B_t}(S_t)\right].$$

Then, it is true that for each $s \in \mathcal{S}$,

$$|\mathbb{E}_s^{\pi,\gamma}\left[\sum_{t=\tau+1}^{\infty} \beta_A^t r_A^{A_t,B_t}(S_t)\right]| \ \leq \ ||r_A||_\infty \frac{\beta_A^{\tau+1}}{1-\beta_A}.$$

Then,

$$||v_A^{\pi,\gamma} - v_{A,\tau}^{\pi,\gamma}|| \ \leq \ ||r_A||_\infty \frac{\beta_A^{\tau+1}}{1-\beta_A} \longrightarrow 0 \ ,$$

when $\tau \to \infty$. This fact is independent or the pair of strategies π, γ chosen. Given that there exists a unique fixed point v_A^*, using the last inequality we have that

$$||v_A^{\pi,\gamma} - v_{A,\tau}^{\pi,\gamma}|| \leq \sup_{\pi} \sup_{\gamma} ||v_A^{\pi,\gamma} - v_{A,\tau}^{\pi,\gamma}|| \longrightarrow 0 \ ,$$

when $\tau \to \infty$.

Finally, using that

$$v_{A,\tau}^* = T_A^\tau(0) \ ,$$

it is true that

$$||v_A^* - T_A^\tau(0)|| \longrightarrow 0 \ ,$$

or

$$T_A^\tau(0) \longrightarrow v_A^*$$

when $\tau \to \infty$, and the result follows. □

Let us observe that, by the error bound proposed in the Theorem holds, then, for $\varepsilon > 0$, taking

$$n \geq \frac{ln(\varepsilon(1-\beta_A)) - ln(||r_A||_\infty)}{ln(\beta_A)}$$

stationary values are ε-optimal for the leader.

Given that the best response of the follower in this games are independent of the future expected value v_B we have ignored its behavior. Anyway, the stationary pair of policies $(f^*, g(f^*))$ it has been guaranteed to exist and they are enough to calculate the value function for SG with MFS as the fixed point:

$$v_B^* = \sum_{a \in A} f_a^* r_B^{a,g(f^*)} + \beta_B \sum_{z \in \mathcal{S}} Q^{ag(f^*)}(z|\cdot) v_B^*.$$

In the next subsection we show that in this case Policy Iteration also converges for SG with MFS.

3.5.2. Policy Iteration

The Policy Iteration (PI) algorithm directly iterates in the policy space. This algorithm starts with an arbitrary policy f and then finds the optimal infinite discounted horizon values, taking into account the optimal response $g(f)$. These values are then used to compute new policies. These two steps of the algorithm can be defined as *Evaluation Phase* and *Improvement Phase*. This algorithm is described in Algorithm 8.

Algorithm 8 Policy Iteration (PI)

1: Require $varepsilon > 0$.
2: Choose an arbitrary stationary Stackelberg pair $(f_0, g(f_0))$.
3: **while** $||u_{A,n} - u_{A,n+1}|| > \varepsilon$ **do**
4: Evaluation Phase: Find $u_{A,n}$ fixed point of the operator $T_A^{f_n}$.
5: Improvement Phase: Find a distribution f_{n+1} such that

$$T_A^{f_{n+1}}(u_{A,n}) = T_A(u_{A,n}) \ .$$

6: $n := n + 1$
7: **end while**
8: **return** Stationary Stackelberg policies $\pi^* = \{f^*, \ldots\}$ and $\gamma^* = \{g(f^*), \ldots\}$

The Evaluation Phase requires to solve a linear system of size $|\mathcal{S}| \times |\mathcal{S}|$. On the other hand, the Improvement Phase can be implemented by solving $(D2)$ for each state $s \in \mathcal{S}$. Now we prove that PI algorithm converges to the SSE in stationary policies for SG with MFS.

Lemma 3.4 If a function $v_A \in \mathcal{F}(\mathcal{S})$ satisfies $v_A \leq T_A^f(v_A)$, for some $f \in \mathbb{P}(\mathcal{A})$ then $v_A \leq v_A^f$, where v_A^f is the unique fixed point of $T_A^f(v_A)$ in $\mathcal{F}(\mathcal{S})$.

Proof. First we note that the following monoticity property is satisfied by the operators T_A^f:

$$u_A \leq v_A \ \Rightarrow \ T_A^f(u_A) \leq T_A^f(v_A) \ .$$

In fact, for each $s \in \mathcal{S}$

$$T_A^f(v_A)(s) - T_A^f(u_A)(s) = \sum_{a \in \mathcal{A}} f(a) \left[r_A^{ag(f)}(s) + \sum_{z \in \mathcal{S}} Q^{a,g(f)}(z|s) \left[v_A(z) - u_A(z) \right] \right] \geq 0.$$

By hypothesis we have that

$$v_A \leq T_A^f(v_A) \ ,$$

that implies by monoticity

$$T_A^f(v_A) \leq (T_A^f)^2(v_A) \ ,$$

and then

$$v_A \leq (T_A^f)^2(v_A) \ .$$

In the same way, for each n we have

$$v_A \leq (T_A^f)^n(v_A) \ ,$$

and by Theorem 3.1 part a.,

$$(T_A^f)^n(v_A) \longrightarrow (v_A^f),$$

the result follows. □

Theorem 3.5 The sequence of functions $u_{A,n}$ verifies $u_{A,n} \uparrow v_A^*$. Further, if for any $n \in \mathbb{N}$, $u_{A,n} = u_{A,n+1}$, then it is true that $u_{A,n} = v_A^*$.

PROOF. For each $s \in \mathcal{S}$, we have that

$$\begin{aligned} u_{A,0}(s) &= T_A^{f_0}(u_{A,0})(s) \\ &\leq T_A(u_{A,0})(s) = T_A^{f_1}(u_{A,0})(s) \ . \end{aligned}$$

then the value function $u_{A,0}$ satisfies

$$u_{A,0} \leq T_A^{f_1}(u_{A,0}) \ ,$$

and by Lemma 3.4

$$u_{A,0} \leq v_A^{f_1} = u_{A,1} \ .$$

Iterating over n, we have that

$$u_{A,n} \leq T_A(u_{A,n}) \leq u_{A,n+1} \ . \tag{3.30}$$

Now the succession $\{u_{A,n}\}_{n\in\mathbb{N}}$ being non-decreasing and bounded by v_A^* then exists a value function $u_A(s)$ such that for any $s \in \mathcal{S}$

$$u_A(s) = \lim_{n\to\infty} u_{A,n}(s) \ .$$

Taking $n \to \infty$ in (3.30), $u_A \leq T_A(u_A) \leq u_A$ and therefore $u_A = T_A(u_A)$, and by unicity of the fixed point

$$u_A = v_A^* \ ,$$

and we have the first claim of the theorem

$$u_{A,n} \uparrow v_A^* \ .$$

Also, if it is verified for some n that $u_{A,n} = u_{A,n+1}$, then, using (3.30),

$$u_{A,n+1} = u_{A,n} \leq T_A(u_{A,n}) \leq u_{A,n+1} \ ,$$

which implies

$$u_{A,n} = T_A(u_{A,n}) = v_A^* \ ,$$

where the second equality is given by the unicity of the fixed point. The result follows. □

The results exposed in this section strongly relies on the fact that $g(f, v_B)$ is independent on v_B. In the next section we show that MFS is a sufficient condition but all the results here may fail in the general case. We show a case where the value function iteration cycles, not finding a stationary policy forming a SSE.

3.6. General Case

In this section we extend the analysis from the MFS case to the general case. We describe and analyze algorithms for solving SSE in feedback strategies for stochastic games with arbitrary time horizon and general instances, extending the operator T_A to a general operator T.

3.6.1. Definition of the Stackelberg Operator in the general case

We generalize the Stackelberg operator T_A defined in (3.21) to calculate SSE in general instances. This is necessary in the general case because the operator T_A is not enough to describe the whole dynamics of the values. In other words, we can not assume that $g(f, v_B) = g(f)$, and given that condition not only the best responses of the follower are affected but also the optimal policies of the leader.

Given a SG $\mathcal{G}$ and a pair of stationary strategies f and g we define the pair of operators $T_A^{f,g}, T_B^{f,g} : \mathcal{F}(\mathcal{S}) \to \mathcal{F}(\mathcal{S})$ by the following expression. For $s \in \mathcal{S}$:

$$T_A^{f,g}(v_A)(s) = \sum_{a \in \mathcal{A}_s} f(a) \sum_{b \in \mathcal{B}_s} g(b) \sum_{z \in \mathcal{S}} Q^{a,b}(z|s) \left[r_A^{a,b}(s) + \beta_A v_A(z) \right], \tag{3.31}$$

$$T_B^{f,g}(v_B)(s) = \sum_{a \in \mathcal{A}_s} f(a) \sum_{b \in \mathcal{B}_s} g(b) \sum_{z \in \mathcal{S}} Q^{a,b}(z|s) \left[r_B^{a,b}(s) + \beta_B v_B(z) \right]. \tag{3.32}$$

We also define for a SG $\mathcal{G}$, the Stackelberg operator for general instances $T : \mathcal{F}(\mathcal{S}) \times \mathcal{F}(\mathcal{S}) \to \mathcal{F}(\mathcal{S}) \times \mathcal{F}(\mathcal{S})$. For each $s \in \mathcal{S}$:

$$\begin{aligned}
(T(v_A, v_B))(s) &= \Big(\max_{f \in \mathbb{P}(\mathcal{A}_s)} \sum_{a \in \mathcal{A}_s} f(a) \sum_{z \in \mathcal{S}} Q^{a,g(f,v_B)}(z|s) \left[r_A^{a,g(f,v_B)}(s) + \beta_A v_A(z) \right] \\
&\quad \sum_{a \in \mathcal{A}_s} f^*(a) \sum_{z \in \mathcal{S}} Q^{a,g(f^*,v_B)}(z|s) \left[r_B^{a,g(f^*,v_B)}(s) + \beta_B v_B(z) \right] \Big) \\
&= \left(\max_{f \in \mathbb{P}(\mathcal{A}_s),} T_A^{f,g(f,v_B)}(v_A)(s),\ T_B^{f^*,g(f^*,v_B)}(v_B)(s) \right) \\
&= (T_A(v_A, v_B),\ T_B(v_A, v_B))(s)
\end{aligned} \tag{3.33}$$

where f^* is the measure which maximizes the first coordinate and the functional $g(f, v_B)$ is defined as in (3.19). This operator returns for each state $s \in \mathcal{S}$ given values $v_A, v_B \in \mathcal{F}(\mathcal{S})$ the strong Stackelberg values in equilibrium of the one-shot game played with payoff matrices:

$$\begin{aligned}
R_A(a, b)(s) &= r_A^{a,b}(s) + \beta_A \sum_{z \in S} Q^{ab}(z|s) v_A(z), \\
R_B(a, b)(s) &= r_B^{a,b}(s) + \beta_B \sum_{z \in S} Q^{ab}(z|s) v_B(z).
\end{aligned}$$

Note that we omit the dependence on $\mathcal{G}$ of $T = T(\mathcal{G})$, unless it is necessary indicate it. Given the pair of value functions (v_A, v_B) and a fixed state $s \in \mathcal{S}$ the operator T can be computed with (D2) as in Section 3.5.1 just changing constrains (3.26) by

$$0 \leq T_B(s) - \sum_{a \in \mathcal{A}} \left[r_A^{ab}(s) + \beta_B \sum_{z \in \mathcal{S}} Q^{ab}(z|s) v_B(z) \right] f_{sa} \leq M_B(1 - g_{sb}) \quad \forall b \in \mathcal{B}_s. \tag{3.34}$$

Now we show a second numerical example that show us that Algorithm VI does not always converges because of the non-contractivity of operator T.

3.6.2. Numerical Example 2

Now we describe a numerical example where VI algorithm do not converge in values. We consider a SG with discount factors $\beta_A = \beta_B = 0.5$. As in the earlier example we consider a set of states $\mathcal{S} = \{s_1, s_2\}$ and actions spaces $\mathcal{A} = \{a_1, a_2\}$, $\mathcal{B} = \{b_1, b_2\}$ for the leader and the follower respectively. Following the earlier notation, we describe the transition and the payoffs in Table 3.4.

	b_1	b_2
a_1	(1, 0) / (1,-1)	(0, 1) / (0, 1)
a_2	(0, 1) / (-1,1)	(0, 1) / $(-1,-1)$

State s_1

	b_1	b_2
a_1	(0, 1) / (-1,0)	(1, 0) / (0, 1)
a_2	(1, 0) / (0,1)	(0, 1) / $(1,-1)$

State s_2

Table 3.4: Transition matrix and payoffs for each player in the numerical example 2.

Beginning with $v_A = v_B = 0 \in \mathbb{R}^{|S|}$, we apply the operator T 13 times. The value functions obtained at this iteration are $v_A^{13} = (0.2714,\ 0.4911)$ and $v_B^{13} = (1.4187,\ 0.6646)$. Using this value function we build the bi-matrices games played in step 14 as follow:

	b_1	b_2
a_1	(1.1357, -0.2910)	(0,2456, 1,3323)
a_2	(-0.7544, 1.3323)	(-0.7544, -0.6677)

State s_1

	b_1	b_2
a_1	(-0.7544, 0.3323)	(0.1357, 1.7093)
a_2	(0.1357, 1.7093)	(1.2456, -0.6677)

State s_2

Table 3.5: Bi- matrix games at stage 14 for numerical example 2.

The SSE in this game is $f_{14}(s_1) = (0.552,\ 0.448)$, $g_{14}(s_1) = (1, 0)$ in state s_1 and $f_{14}(s_2) = (0.633,\ 0.367)$, $g_{14}(s_2) = (1, 0)$ in state s_2. Values obtained at this step are $v_A^{14} = (0.2890,\ 0.5428)$ and $v_B^{14} = (0.4364,\ 0.8374)$.

In the next iteration, the bi-matrix game that is played in each state is the following:

	b_1	b_2
a_1	(1.1445, -0.7818)	(0.2714, 1.4187)
a_2	(-0.7286, 1.4187)	(-0.7286, -0.5813)

State s_1

	b_1	b_2
a_1	(-0.7286, 0.4187)	(0.1445, 1.2182)
a_2	(0.1445, 1.2182)	(1.2174, -0.5813)

State s_2

Table 3.6: Bi- matrix games at stage 15 for numerical example 2.

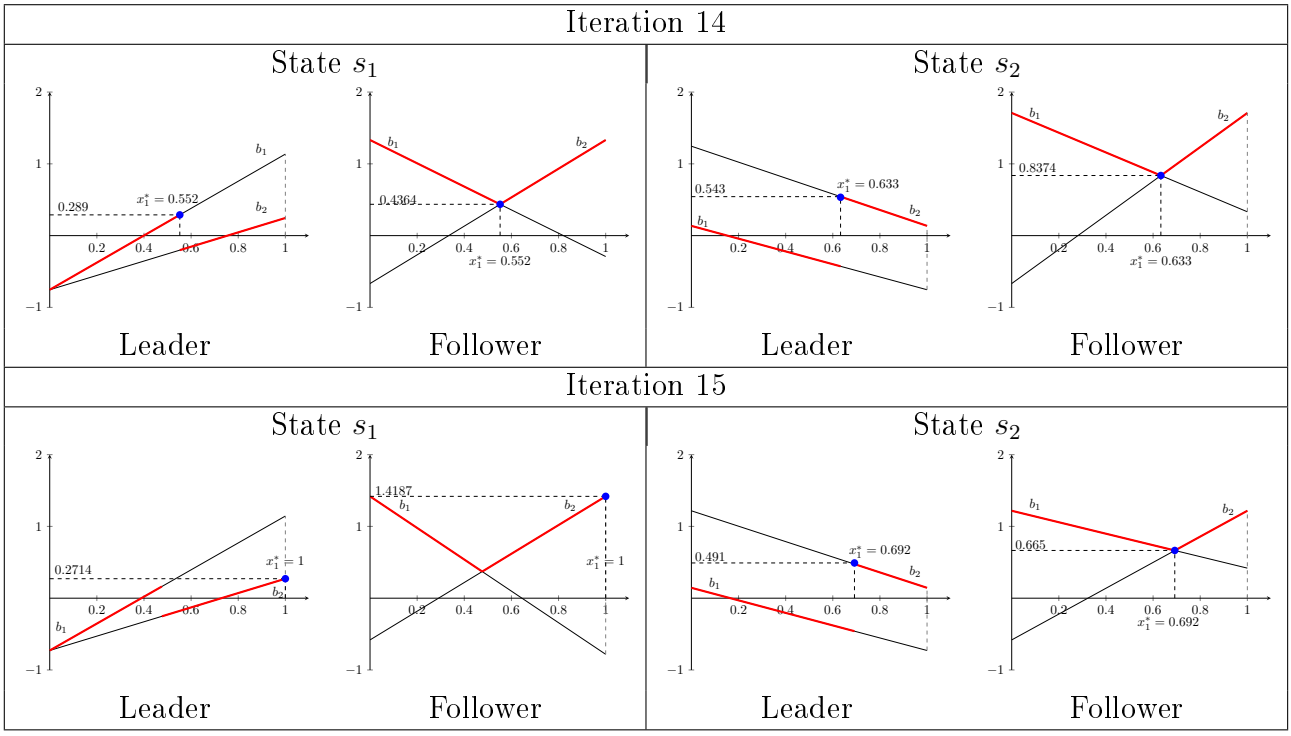


Table 3.7: Iterations 14 and 15 for numerical example 2.

At this step the SSE is given by $f_{15}(s_1) = (1, 0)$ $g_{15}(s_1) = (0, 1)$ in state s_1 and $f_{15}(s_2) = (0.692,\ 0.308)$, $g_{15}(s_2) = (0, 1)$ in state s_2. Values obtained at this step are v_A^{15} =(0.2714, 0.4911) and $v_B^{15} = (1.4187,\ 0.6646)$, the same at the iteration 13. The summary of both iterations is represented in Table 3.7. The algorithm cycles.

Why does it fail? We cannot give an exact answer to this question, but we can say a few words related. First, observe that if we would know that T is a contractive operator, the value sequence will result convergent, but this is not the case. As an example, we take functions the v_A^{14}, v_B^{14} and v_A^{15}, v_B^{15} and observe that

$$|T_A(v_A^{14}(s_1), v_B^{14}(s_1)) - T_A(v_A^{15}(s_1), u_B^{15}(s_1))| > \frac{1}{2}\text{max}_{z\in\mathcal{S}}|v_A^{14}(z) - v_A^{15}(z)|$$

$$|0.5428 - 0.4911| > \frac{1}{2}\text{max}\{|0.2714 - 0.2890|,\ |0.5428 - 0.4911|\},$$

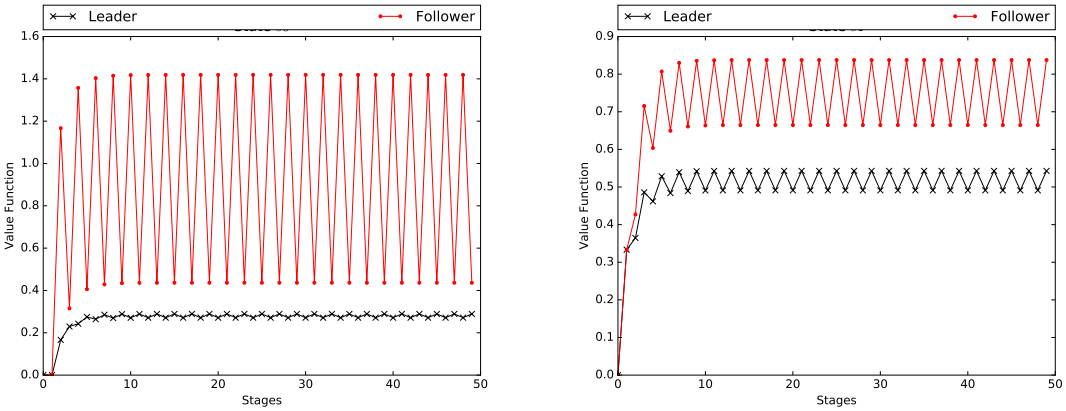
which shows that this instance is not $\frac{1}{2}$-contractive. Even more, if we change β_A to any value between (0,1) the negative result also holds. Then the contractivity, our sufficient condition, is not satisfied for this game and we can not guaranty convergence.

Of course we know that this argument is not enough to assure convergence, since the are convergent instances in the class of non-expansive operators (which includes strictly the class of contractive operators). A non-expansive operator is an operator T verifying, for any u and v,

$$||Tu - Tv|| \leq ||u - v||,$$

which is the case of our example.

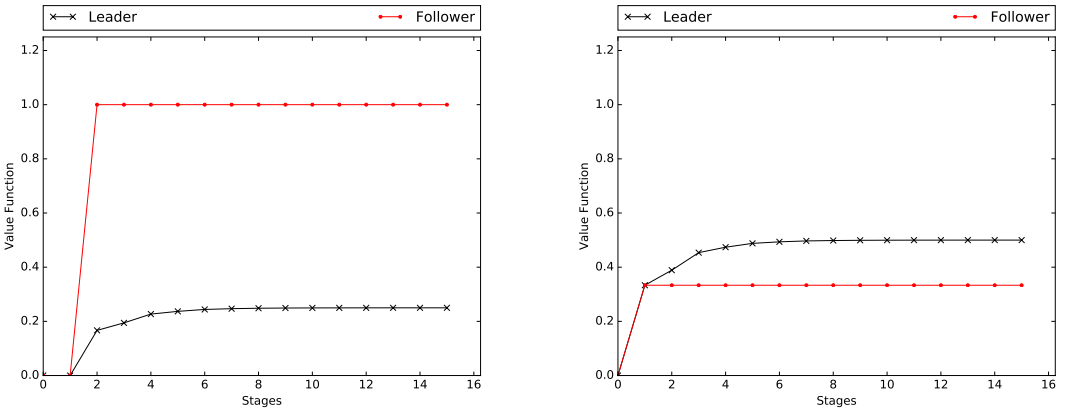
In Figure (3.6) is presented the behavior of the value functions through the application of the operator T. Note that for state s_1, the leader passes from a mixed strategy to a pure one and the follower changes the action b_1 to b_2 in iteration 14 and 15 respectively. That makes values oscillate between two values.



State s_1 State s_2

Figure 3.6: Value Function vs time in numerical example 2.

Note that if we change this games to the MFS case, values converges. First, if we apply value function iteration to the same game but changing to $\beta_B = 0$, Figure 3.7 shows that in few iterations the dynamic programming algorithm converges.



State s_1 State s_2

Figure 3.7: Value Function vs time in numerical example 2 in the myopic case ($\beta_B = 0$).

If we change the instance to a Leader-controller transition, just changing the transition probabilities (see Table 3.8), the algorithm converges in few iterations as we see in Figure 3.8.

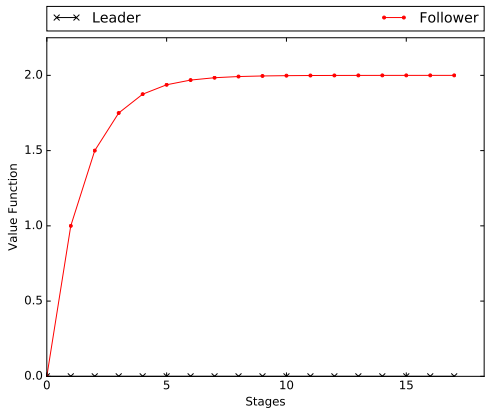
	b_1	b_2
a_1	$(1,0)$	$(1,0)$
a_2	$(0,1)$	$(0,1)$

State s_1

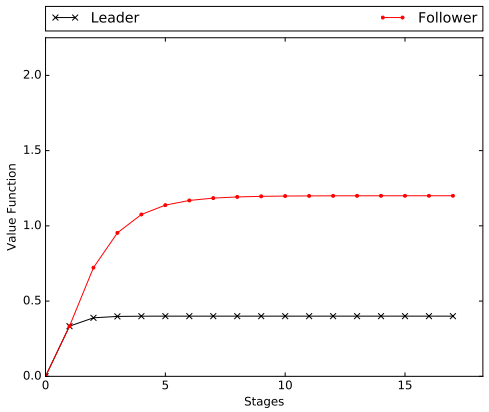
	b_1	b_2
a_1	$(1,0)$	$(1,0)$
a_2	$(0,1)$	$(0,1)$

State s_2

Table 3.8: Transformation of the matrix in numerical example 2 to a Leader-controller case.



State s_1



State s_2

Figure 3.8: Value Function vs time in numerical example 2 in Leader-controller case.

In the next subsection we describe the algorithms of VI in both, finite horizon and infinite horizon using the operator T. We also describe the PI procedure to the general case.

3.6.3. Value Function and Policy Function for the general case

We adapt algorithms exposed in section 3.5 to calculate (when it is possible) strong Stackelberg equilibrium in feedback policies for the finite horizon problem and stationary policies for the infinite horizon problem.

The Algorithm 9 implements dynamic programming to find SSE in feedback strategies for the finite horizon problem. In [5, Theorem 7.4] it is shown that when this algorithm ends, it returns both the value for our game and a pair of Stackelberg feedback policies (π^*, γ^*) for the τ-finite horizon game.

Algorithm 9 Value Iteration (VI): Finite horizon for the general case

1: Initialize with $v_A^{\tau+1}(s) = v_B^{\tau+1}(s) = 0$ for every $s \in \mathcal{S}$
2: **for** $t = \tau, \ldots, 1$, and for every $s \in \mathcal{S}$ **do**
3: Solve

$$\left(v_A^t(s), v_B^t(s)\right) = T(v_A^{t+1}, v_B^{t+1})(s) \tag{3.35}$$

Finding f_t^* and g_t^* SSE strategies at stage t.
4: **end for**
5: **return** Stackelberg policies $\pi^* = \{f_1^*, \ldots, f_\tau^*\}$ and $\gamma^* = \{g_1^*, \ldots, g_\tau^*\}$

Following the same logic as the MFS case, the infinite horizon version of this algorithm is proposed by Algorithm 10. Note that it is necessary to introduce a concept of distance between two pairs of functions (v_A, v_B) (u_A, u_B) to introduce the convergence. We use the following norm in $\mathcal{F}(\mathcal{S}) \times \mathcal{F}(\mathcal{S})$:

$$||(v_A, v_B)|| \;=\; \text{máx}\{\max_{s\in\mathcal{S}} |v_A(s)|, \max_{s\in\mathcal{S}} |v_A(s)|\} \;=\; \text{máx}\{||v_A||_\infty, ||v_B||_\infty\}. \tag{3.36}$$

Now we show, in Lemma 3.6, that this quantity is a norm and makes $(\mathcal{F}(\mathcal{S}) \times \mathcal{F}(\mathcal{S}), ||\cdot||)$ a Banach space.

Lemma 3.6 The quantity $||\cdot||$ defined by expression (3.36) is a norm on $\mathcal{F}(\mathcal{S}) \times \mathcal{F}(\mathcal{S})$, which makes $(\mathcal{F}(\mathcal{S}) \times \mathcal{F}(\mathcal{S}), ||\cdot||)$ a Banach space.

Proof. It is easy to check that $||(v_A, v_B)|| \geq 0$ and it is equal to zero when both v_A and v_B are the null function. Also, it is easy to check that for any $\lambda \in \mathbb{R}$ $||(\lambda v_A, \lambda v_B)|| = |\lambda|||(v_A, v_B)||$. The Triangle inequality also holds. Take $(v_A, v_B), (u_A, u_B) \in F(\mathcal{S}) \times F(\mathcal{S})$:

$$\begin{aligned} ||(v_A, v_B) + (u_A, u_B)|| &= ||(v_A + u_A, v_B + u_B)|| \\ &= \max_{s\in\mathcal{S}} (\text{máx}\{|v_A(s) + u_A(s)|, |v_B(s) + u_B(s)|\}) \end{aligned}$$

W.l.o.g suppose that the maximum is achieved in the first coordinate at $s^* \in \mathcal{S}$, otherwise assume that the maximum is achieved in the second one, and repeat the argument:

$$\begin{aligned} \max_{s\in\mathcal{S}} (\text{máx}\{|v_A(s) + u_A(s)|, |v_B(s) + u_B(s)|\}) &= |v_A(s^*) + u_A(s^*)| \\ &\leq |v_A(s^*)| + |u_A(s^*)| \\ &\leq \max_{s\in\mathcal{S}} \{|v_A(s)|, |v_B(s)|\} + \max_{s\in\mathcal{S}} \{|u_A(s)|, |u_B(s)|\} \\ &= ||(v_A, v_B)|| + ||(u_A, u_B)||. \end{aligned}$$

It remains to check that $(F(\mathcal{S}) \times F(\mathcal{S}), ||\cdot||)$ is a complete normed space. Let $\{(v_A^n, v_B^n)\}_{n\in\mathbb{N}}$ be a Cauchy sequence in $F(\mathcal{S}) \times F(\mathcal{S})$, then:

$$||(v_A^n, v_B^n) - (v_A^m, v_B^m)|| < \varepsilon \quad n, m > N_0.$$

That implies for every $n, m > N_0$

$$||(v_A^n - v_A^m), (v_B^n, v_B^m)|| = \max_{s\in\mathcal{S}}\{|(v_A^n(s) - v_A^m(s))|, |(v_B^n(s) - v_B^m(s))|\} < \varepsilon.$$

Then $\{v_A^n\}_n$ and $\{v_B^n\}$ are Cauchy sequences. Given that $F(\mathcal{S})$ is a complete space, $v_A^n \to v_A^* \in F(\mathcal{S})$ and $v_B^n \to v_B^* \in F(\mathcal{S})$, then:

$$\begin{aligned} ||(v_A^n, v_B^n) - (v_A^*, v_B^*)|| &= ||(v_A^n - v_A^*), (v_B^n - v_B^*)|| \\ &= \max_{s\in\mathcal{S}}\{|(v_A^n(s) - v_A^*)(s)|, |(v_B^n(s) - v_B^*(s))|\} \to 0 \end{aligned}$$

and the result is obtained. □

Algorithm 10 Value Iteration (VI): Infinite horizon for the general case

Require: $\varepsilon > 0$

1: Initialize with $n = 0$, $v_A^0(s) = v_B^0(s) = 0$ for every $s \in \mathcal{S}$
2: **while** $||(v_A^n, v_B^n) - (v_A^{n-1}, v_B^{n-1})|| > \varepsilon$ **do**
3: find the pair (v_A^n, v_B^n) by

$$(v_A^n, v_B^n)(s) = T(v_A^{n-1}, v_B^{n-1})(s)\ .$$

Finding f^* and g^* SSE strategies at stage $n-1$.
4: **end while**
5: **return** Stationary Stackelberg policies $\pi^* = \{f^*, \ldots\}$ and $\gamma^* = \{g^*, \ldots\}$

As we see in the numerical example 2, The algorithm does not always terminate. In the next subsection we discuss first what fails in the analogous for theorems exposed in Section 3.5.

PI is also extended to the general case. As stationary policies forming a SSE are not guaranteed to exists, when this algorithm ends, we are not able to consider the strategies returned to be a Stackelberg equilibrium. Algorithm 11 implements policy iteration for the general case. Note that this algorithm needs to solve at each step 2 linear systems of $|\mathcal{S}| \times |\mathcal{S}|$ in the evaluation phase and (D2) $|\mathcal{S}|$ times in the improvement phase, at each iteration.

Algorithm 11 Policy Iteration (PI): General case

1: Choose a stationary Stackelberg policy f_0 and calculate $g_0 = g(f_0, 0)$.
2: Set $u_{A,-1} = u_{B,-1} = 0$ and $u_{A,0}, u_{B,0} = (T_A^{f_0,g_0}(0,0), T_B^{f_0,g_0}(0,0))$.
3: Set n := 0.
4: **while** $||(u_{A,n}, u_{B,n}) - (u_{A,n-1}, u_{B,n-1})|| > \varepsilon$ **do**
5: Set $n := n + 1$.
6: Evaluation phase: Find $(u_{A,n}, u_{B,n})$ fixed point of the operator $T^{f_n,g_n(f_n,u_{B,n})}$.
7: Improvement phase: Find a pair $(f_n, g_n(f_n))$ such that

$$T^{f_{n+1},g_{n+1}(f_{n+1},u_{B,n})}(u_{A,n}, u_{B,n}) = T(u_{A,n}, u_{B,n})\ .$$

8: **end while**
9: **return** Stationary Stackelberg policies $\pi^* = \{f^*, \ldots\}$ and $\gamma^* = \{g^*, \ldots\}$

In the next section, we give some ideas of why, in general, SSE cannot be computed using VI. We will do that in terms of contractivity properties of the Dynamic Operator propposed.

3.6.4. What does it fail?

To ensure the convergence to a unique fixed point of T as in Theorem 3.1 part b) a sufficient condition is given by the contractivity of the operator T, that is forall (v_A, v_B) and $(u_A, u_B) \in \mathcal{F}(\mathcal{S}) \times \mathcal{F}(\mathcal{S})$:

$$||T(v_A, v_B) - T(u_A, u_B)|| \leq \beta ||(v_A, v_B) - (u_A, u_B)|| \tag{3.37}$$

for some $\beta \in (0, 1)$, which we would have, for instance, if we can prove

$$||T_A(v_A, v_B) - T_A(u_A, u_B)||_\infty \leq \beta_A ||v_A - u_A||_\infty \tag{3.38}$$
$$||T_B(v_A, v_B) - T_B(u_A, u_B)||_\infty \leq \beta_B ||v_B - u_B||_\infty, \tag{3.39}$$

taking $\beta = \text{máx}\{\beta_A, \beta_B\}$.

Unfortunately, Numerical Example 2 shows us that there does not exist such β for all the instances $\mathcal{G}$ (if it would, VI sequence will converge). With this in mind, the challenge is now to impose some restrictions on the parameters of the problems to assure the contractivity of T and then the convergence of VI. On the other hand, even if we could characterize the conditions on $\mathcal{G}$ under which T results a contractive operator, this conditions will not be strictly necessary for the convergence of VI.

In order to ensure that PI converges we need additionally some extra conditions:

- We need an analogous of Theorem 3.1a. for the operator T. This condition can be stated as follows:

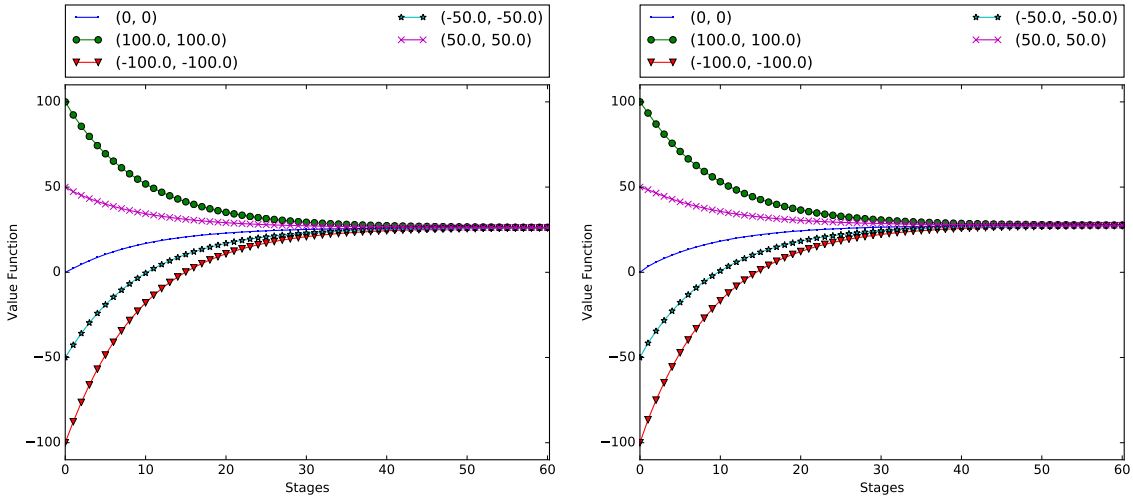
  For any pair of functions $v_{A,0}, v_{B,0}$ and a fixed stationary policy f, the sequence of functions $v_{A,n}, v_{B,n}$ defined by the recurrence

  $$v_{A,n+1} = T_A^{f,g(f,v_{B,n})}(v_{A,n}, v_{B,n}), \quad v_{B,n+1} = T_B^{f,g(f,v_{B,n})}(v_{A,n}, v_{B,n})$$

  converges to a unique v_A^{f*}, v_B^{f*}.
- We also need monotonicity of operator $T(\mathcal{G})$:

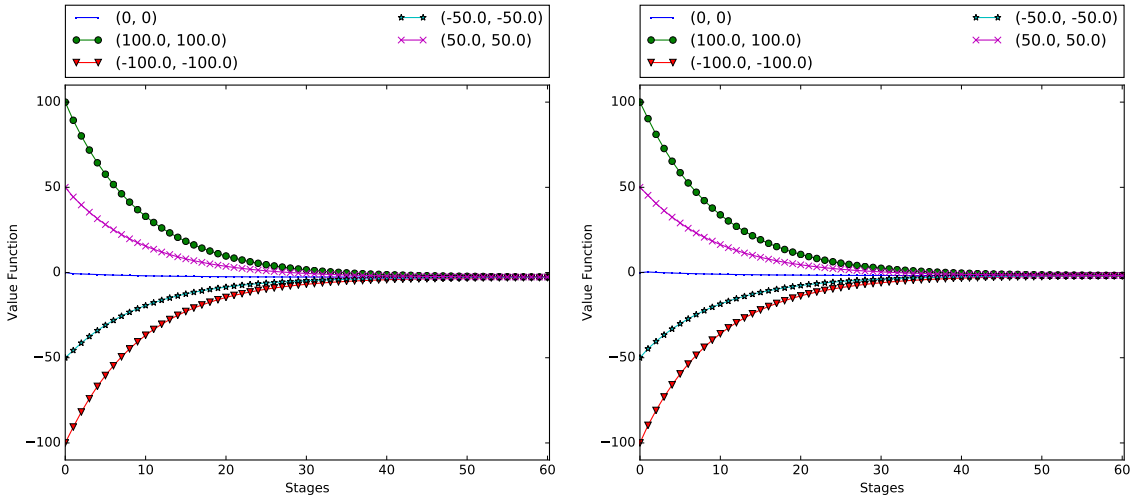
  For (v_A, v_B) and (u_A, u_B), such that $(v_A, v_B) \leq (u_A, u_B)$ then $T(v_A, v_B) \leq T(u_A, u_B)$.

We suspect that not only MFS instances are contractive. For instance, numerical example 1 in Section 3.4, which is not MFS, shows us that the application of operator T starting from $(0, 0) \in \mathcal{F}(\mathcal{S}) \times \mathcal{F}(\mathcal{S})$, the algorithm converges. This is not implying that T is contractive for this game because contractivity is independent of the initial values. However, in Figures 3.9 and 3.10 we test several initial values, all converging to a unique value. That make us suspect that T is contractive for the game described in the numerical example 1.



State s_1 State s_2

Figure 3.9: Convergence of v_A starting from different starting values $(v_A^0,\, v_B^0)$.



State s_1 State s_2

Figure 3.10: Convergence of v_B starting from different starting values $(v_A^0,\, v_B^0)$.

Finally, we can adapt the VI procedure to a finite version of Algorithm 10.

As we have said, if T would be contractive, we will have he existence of a unique (v_A^*, v_B^*) fixed point of T. In that case, the following condition has to be satisfied by the operator at the n-th application of T:

$$||(v_A^n, v_B^n) - (v_A^*, v_B^*)|| \leq \frac{\beta^n}{1-\beta}||(r_A, r_B)||. \tag{3.40}$$

for some $\beta < 1$. Using that we have that:

$$\begin{aligned}||(v_A^n, v_B^n) - (v_A^{n-1}, v_B^{n-1})|| &\leq ||(v_A^n, v_B^n) - (v_A^*, v_B^*)|| + ||(v_A^{n-1}, v_B^{n-1}) - (v_A^*, v_B^*)|| \\ &\leq \frac{\beta^n}{1-\beta}||(r_A, r_B)|| + \frac{\beta^{n-1}}{1-\beta}||(r_A, r_B)|| \\ &\leq 2\frac{\beta^{n-1}}{1-\beta}||(r_A, r_B)||.\end{aligned}$$

Then we have that $||(v_A^n, v_B^n) - (v_A^{n-1}, v_B^{n-1})|| \to 0$ as $n \to \infty$ and if for any n the condition

$$||(v_A^n, v_B^n) - (v_A^{n-1}, v_B^{n-1})|| > 2\frac{\beta^{n-1}}{1-\beta}||(r_A, r_B)||. \tag{3.41}$$

holds, then T is not β-contractive, and we can not assure the geometric convergence of VI of rate β. Nevertheless this does not imply neither T is not contractive with any modulus of contractivity, nor the non-convergence of VI

With these elements we adapt the algorithm of VI to the general case, where if T is contractive returns the strong Stackelberg equilibrium in stationary policies. This algorithm is described in Algorithm 12.

Algorithm 12 Value function iteration modified: Infinite horizon for the general case

Initialize with $n = 0$, $v_A^0(s) = v_B^0(s) = 0$ for every $s \in \mathcal{S}$.
for $n = 1, \cdots, MAX\_IT$ **do**
  Find the pair (v_A^n, v_B^n) by

$$(v_A^n, v_B^n)(s) = T(v_A^{n-1}, v_B^{n-1})(s) \ .$$

  Finding f^* and g^* SSE strategies at stage $n-1$.
  if $(v_A^n, v_B^n) = (v_A^{n-1}, v_B^{n-1})$ **then**
    return (v_A^n, v_B^n) fixed point of T.
  end if
  if $||(v_A^n, v_B^n) - (v_A^{n-1}, v_B^{n-1})|| > 2\frac{\beta^{n-1}}{1-\beta}||(r_A, r_B)||$ **then**
    return UNDEFINED.
  end if
end for
return UNDEFINED.

Let us emphasize that, if the algorithm terminates in a fixed point, then it returns an equilibrium. Otherwise,the algorithm does not define neither the convergence of VI nor the non-convergence of the VI procedure.

3.7. Mathematical Programming approach

In this section we show a Mathematical approach that we compare to the models exposed in Section 3.2.2. We start noting that given a stationary policy for the leader $\pi = \{f, \ldots\}$,

the follower solves the following problem for each state $s \in \mathcal{S}$

$$v_B^f(s) = \max_{b \in \mathcal{B}_s} \sum_{a \in \mathcal{A}} f(s,a) \left[r_B^{ab}(s) + \beta_B \sum_{z \in S} Q^{ab}(z|s) v_B(z) \right], \tag{3.42}$$

$$= \max_{b \in \mathcal{B}_s} \left\{ \bar{r}_B^{fb}(s) + \beta_B \sum_{z \in S} \bar{Q}^{fb}(z|s) v_B(z) \right\}, \tag{3.43}$$

where $\bar{r}_B^{fb}(s) = \sum_{a \in \mathcal{A}_s} f_s(a) r_B^{ab}(s)$ and $\bar{Q}^{fb}(z|s) = \sum_{a \in \mathcal{A}} f_s(a) Q^{ab}(z|s)$. With this policy fixed, the follower solves a Markov decision process (MDP) which can be written as in [50, Ch. 6.9]:

$$(\mathrm{P}_f) \quad \min \sum_{s \in \mathcal{S}} \alpha_s v_B(s) \tag{3.44}$$

$$s.t. \;\; v_B(s) - \beta_B \sum_{z \in S} \bar{Q}^{fb}(z|s) v_B(z) \geq \bar{r}_B^{fb}(s) \qquad \forall s \in \mathcal{S}, \tag{3.45}$$

where $\alpha \in \mathbb{R}_+^{|\mathcal{S}|}$ is a strictly positive vector of coefficients. The dual of this problem is:

$$(\mathrm{D}_f) \quad \max \sum_{s \in \mathcal{S}} \sum_{b \in \mathcal{B}} \bar{r}_B^{fb}(s) y_{sb} \tag{3.46}$$

$$s.t. \sum_{b \in \mathcal{B}} y_{sb} - \beta_B \sum_{z \in S} \sum_{b \in \mathcal{B}} \bar{Q}^{fb}(s|z) y_{zb} = \alpha_s \qquad \forall s \in \mathcal{S} \tag{3.47}$$

$$y_{sb} \geq 0. \tag{3.48}$$

In this case, an optimal solution y^* of (D_f), an optimal stationary policy can be retrieved for the leader as follows:

$$g(s,b) = \begin{cases} \frac{y_{sb}}{\sum_{b \in \mathcal{B}} y_{sb}} & \text{if } \sum_{b \in \mathcal{B}} y_{sb} > 0 \\ \frac{1}{|\mathcal{B}|} & \text{otherwise.} \end{cases} \tag{3.49}$$

By [50, Theorem. 6.9.3] every deterministic policy can be represented as an extreme point of (D_f), in particular, there exists an optimal policy which is deterministic. Then, we can write a more restrictive version of this problem but with the same value. Let us define

$$q_{sb} = \begin{cases} 1 & \text{if } y_{sb} > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{3.50}$$

and or equivalently represented with the following set of linear constraints

$$\sum_{b \in \mathcal{B}} q_{sb} = 1 \qquad \forall s \in \mathcal{S} \tag{3.51}$$

$$q_{sb} \geq y_{sb} \qquad \forall s \in \mathcal{S}, \forall b \in \mathcal{B} \tag{3.52}$$

$$q_{sb} \in \{0,1\} \qquad , \tag{3.53}$$

On the other hand, the solution of (P_f) and (D_f) satisfies strong duality and complementary slackness conditions for linear programs, in addition with feasibility constraints (3.45), (3.47)

and (3.48):

$$\text{(Strong Duality)} \quad \sum_{s\in\mathcal{S}} \alpha_s v_B(s) = \sum_{s\in\mathcal{S}} \sum_{b\in\mathcal{B}_s} \bar{r}_B^{fb}(s) y_{sb} \tag{3.54}$$

$$\text{(Comp. Slackness 1)} \quad y_{sb}\left(v_B(s) - \sum_{z\in S} \bar{Q}^{fb}(z|s) v_B(z) - \bar{r}_B^{fb}(s)\right) = 0 \tag{3.55}$$

$$\text{(Comp. Slackness 2)} \quad v_B(s)\left(\sum_{b\in\mathcal{B}_s} y_{sb} - \beta_B \sum_{z\in S}\sum_{b\in\mathcal{B}_z} \bar{Q}^{fb}(s|z) y_{zb} - \alpha_s\right) = 0 \quad . \tag{3.56}$$

Note that the condition given by (3.56) is satisfied by any feasible solution of (D_f). Given that we only focus on deterministic policies for the follower, condition (3.55) can be represented as:

$$0 \leq v_B(s) - \beta_B \sum_{z\in S} \bar{Q}^{fb}(z|s) v_B(z) - \bar{r}_B^{fb}(s) \leq \frac{||r_B||_\infty}{1-\beta_B}(1-q_{sb}). \tag{3.57}$$

The idea is to use optimality conditions as constrains in an optimization problem to compute the best strategy f for the leader. If T is contractive and monotone, $u \leq v$ implies that $T(u) \leq T(v)$, then finding values that are optimal, implies to find values that are the minimum upper bound of each state.

$$(\mathrm{P}_{SS}) \quad \text{mín} \quad \sum_{s\in\mathcal{S}} \iota_s v_A(s) \tag{3.58}$$

$$s.t. \quad v_A(s) \geq \sum_{a\in\mathcal{A}_s} f_{sa}\left[r_A^{ab} + \beta_A \sum_{z\in S} Q^{ab}(z|s) v_B(z)\right] - \frac{||r_A||_\infty}{1-\beta_A}(1-q_{sb}) \quad \forall s \in \mathcal{S} \tag{3.59}$$

$$\text{Conditions (3.47) - (3.48) - (3.51) - (3.52) - (3.53) - (3.57)}, \tag{3.60}$$

where $\iota \in \mathbb{R}_+^{|\mathcal{S}|}$ is a positive vector of coefficients. The model here is extremely non-linear as the model (MINLP). There is a vast literature of Mathematical programming with equilibrium constrains (see [41]). Solvers like KNITRO [66] or PATH [23], [26] can be used to solve this types of problems in several programming languages (Matlab, Python, GAMS, AMPL, etc).

This type of models can be extended to calculate feedback policies for a fixed horizon τ solving stationary policies for the following game:

$$\bar{\mathcal{G}} = (\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{\mathcal{B}}, \bar{Q}, \bar{r}_A, \bar{r}_B, \beta_A, \beta_B, +\infty),$$

where $\bar{\mathcal{S}} = S \times \{1, \ldots, \tau\}$, for every t the set of actions are in the form $\bar{\mathcal{A}}_{(s,t)} = \mathcal{A}_s$ and $\bar{\mathcal{B}}_{(s,t)} = \mathcal{B}_s$, and the transitions function Q are in the form of

$$Q^{ab}\left((z,t)|(s,t')\right) = \begin{cases} Q^{ab}(z|s) & \text{if } t = t'+1 \\ 0 & \text{otherwise.} \end{cases}$$

This type of transformation is the same that is used in [58] and also used in the model (DFQ) exposed in section 3.2.2. This technique of increasing the space of states is also used to get more complex policies for a fixed time horizon in [61].

Anyway, in terms of computational scalability, for the instances for which we know there exists equilibrium, algorithms based on dynamic programming outperform solvers implementing the model exposed in this section and in Section 3.2.2.

3.8. Computational Experiments

In this section we aimed to compare the performance of the algorithms exposed in Sections 3.5 and 3.6.3 and analyze how the algorithms impact solution times in the different types of instances, how different parameters affect the convergence of algorithms and the existence or not of strong Stackelberg equilibrium. All codes were programmed using Python 3.5 and Gurobi 7.5 to solve optimization problems as subroutines. The experiments were run on a PC with Windows 10 and a Intel i7 processor of 2.4 GHz and 12 GB RAM.

3.8.1. Performance of algorithms: (MFS) case.

We first analyze the performance of the algorithms for instances in which we know it is possible to find a SSE in stationary policies through VI. We denote as n the number of states we consider and m_A and m_B the size of the action space for player A and B respectively. We use $n = m_A = m_B = 10$ as base case. Rewards were generated uniformly between $[-100, 100]$ and transition matrices were generated by Algorithm 13. Given matrix Q we apply a rounding process to avoid numerical issues. Also we randomly generated the values of β_A and β_B uniformly between 0 and 1. It is easy to adapt Algorithm 13 to generate Leader Controller instances. Due to numerical problems we restrict each element of Q to not have more than 2 decimals.

Algorithm 13 Generating Stochastic Matrix

Require: $\mathcal{S}$, $\mathcal{A}$, $\mathcal{B}$.
1: Initialize with $Q = \{\}$.
2: **for** $s \in \mathcal{S}$, $a \in \mathcal{A}$, $b \in \mathcal{B}$, **do**
3: Set $M = 0$
4: **for** $z \in \mathcal{S}$ **do**
5: $Q^{ab}(z|s) = \mathcal{U}(0, 100)$
6: $M = M + Q^{ab}(z|s)$
7: **end for**
8: **for** $z \in S$ **do**
9: $Q^{ab}(z|s) = \frac{Q^{ab}(z|s)}{M}$
10: **end for**
11: **end for**
12: **return** Stochastic matrix Q, randomly generated.

Starting from the base case, we change the values of n, m_A and m_B to take values in $\{2, 25, 50, 100, 200\}$, one at a time. For each set of parameters (n, m_A, m_B) we run $k = 30$ instances. In myopic instances, we generate the instances as before but setting $\beta_B = 0$. Solution times and number of iterations are shown in Table 3.9 and Figure 3.11.

				Time[s]		Iterations [it]	
	n	m_A	m_B	VI	PI	VI	PI
Base Case	10	10	10	12.49	1.10	58.23	2.77
$\Delta\mathcal{S}$	2	10	10	3.08	0.19	74	2.37
	25	10	10	23.80	2.68	40.40	2.30
	50	10	10	128.74	6.53	95.45	2.31
	100	10	10	275.87	22.51	71.53	2.63
	200	10	10	432.09	80.12	40.65	3.2
$\Delta\mathcal{A}$	10	2	10	4.72	0.52	29.60	2.27
	10	25	10	24.19	2.57	77.83	2.90
	10	50	10	18.18	4.73	42.53	3
	10	100	10	46.14	9.86	69.73	3.37
	10	200	10	47.85	21.59	42.97	3.83
$\Delta\mathcal{B}$	10	10	2	6.95	0.45	53.07	2.5
	10	10	25	25.33	3.11	53.1	2.93
	10	10	50	81.84	6.97	72.86	2.90
	10	10	100	105.63	16.92	38.7	3.13
	10	10	200	360.32	49.11	52.12	2.94

Table 3.9: Solution times and iterations performed by algorithms in myopic instances.

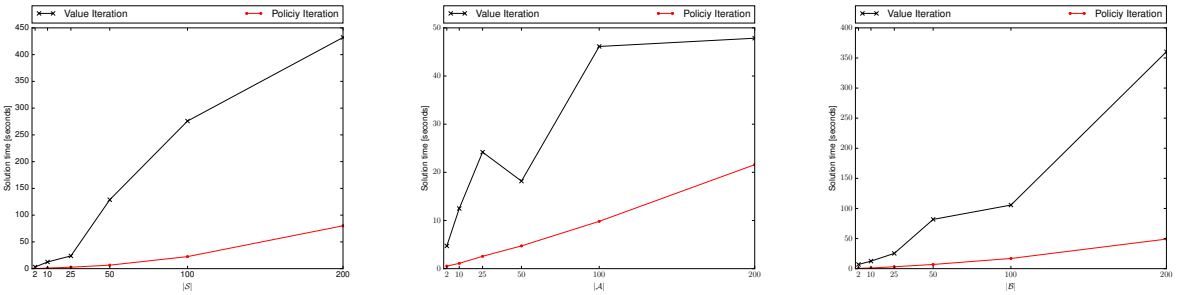


Figure 3.11: Performance of VI and PI in myopic instances.

In Leader controller instances, results are shown in Table 3.10 and Figure 3.12.

				Time[s]		Iterations [it]	
	n	m_A	m_B	VI	PI	VI	PI
Base Case	10	10	10	3.48	1.28	15.72	3.07
$\Delta\mathcal{S}$	2	10	10	2.27	0.19	58.77	2.43
	25	10	10	1.87	2.65	3.17	2.37
	50	10	10	2.69	5.39	2	2
	100	10	10	6.50	14.03	2	2
	200	10	10	19.13	46.18	2	2
$\Delta\mathcal{A}$	10	2	10	2.36	0.52	14.53	2.43
	10	25	10	9.55	3.25	16.43	3.57
	10	50	10	10.71	6.55	18.53	3.77
	10	100	10	34.62	13.97	22.2	3.93
	10	200	10	25.86	20.16	20.73	3.8
$\Delta\mathcal{B}$	10	10	2	1.59	0.44	15.53	3.03
	10	10	25	6.57	2.66	17	3.17
	10	10	50	19.71	7.01	19.77	3.5
	10	10	100	39.89	14.14	17.3	3.33
	10	10	200	156.44	42.41	20.6	3.5

Table 3.10: Resolution time and iterations performed by algorithms in Leader Controller Instances.

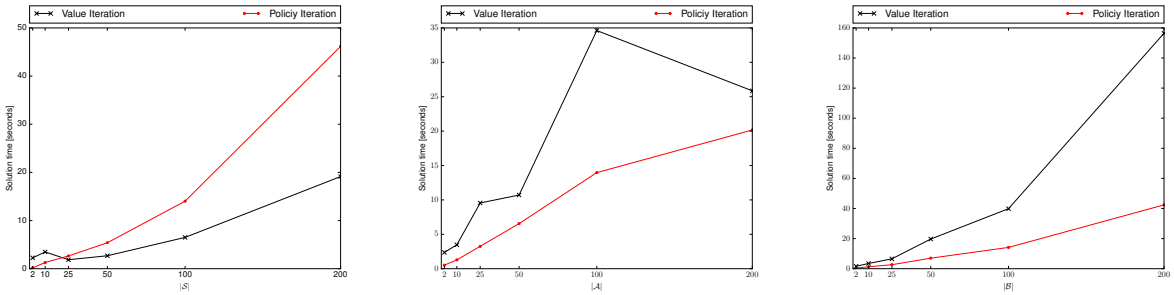


Figure 3.12: Performance of VI and PI in Leader Controller instances.

3.8.2. Stackelberg Security Games

We extend the definition of Stackelberg Security Games to SG, defining instances with the following payoffs structure:

$$r_A^{ab}(s) = \begin{cases} R_A(b) > 0 & \text{if } b = a \\ P_A(b) < 0 & \text{otherwise} \end{cases} \tag{3.61}$$

$$r_B^{ab}(s) = \begin{cases} P_B(b) < 0 & \text{if } b = a \\ R_B(b) > 0 & \text{otherwise} \end{cases} \tag{3.62}$$

This type of game can be interpreted as a game similar to the one proposed in Section 3.3 where a defender has to protect different locations $\mathcal{L}$ and an attacker has to choose a location where to attack. Note that in each state we are analyzing a square game, given by the size of the set of locations $\mathcal{L}$. We denote the size of this set with m.

We use Algorithm 12 to measure the fraction of instances where the algorithm converges. For those instances we compare which algorithm has a better performance. We test square games that is games where in each state both sets of actions have the same dimension m. In our setting, m takes values in $\{10, 25, 50, 75\}$ and we ran 50 experiments for each size of m.

Given the computation accuracy, in all the instances tested, the algorithm terminates by the equality condition, and then in a stationary equilibrium in a number of iterations less than the maximum number of iteration. Since this was the case, we have the geometric decreasing of the values, we conjecture that the operator $T(\mathcal{G})$ is contractive for every game with this payoffs structure. Anyway, we do not provide any proof or counterexample.

The performance of the algorithms in security instances are represented in Figure 3.13, where the behavior is similar to the instances already tested. PI outperforms VI again. Nevertheless, we are not able to affirm that the PI will converge to an equilibrium.

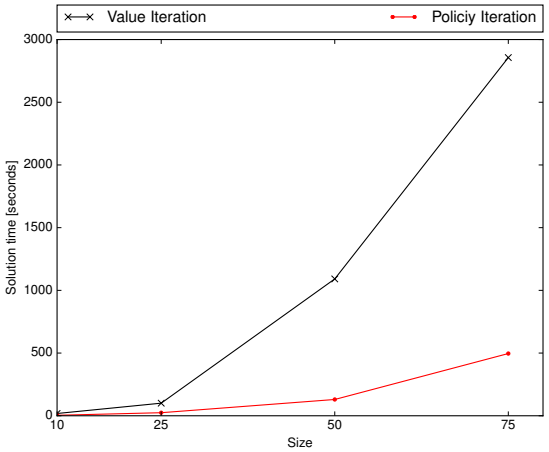


Figure 3.13: Performance of value function iteration and policy iteration in security games instances.

3.8.3. General Case

In general instances, assuming the accuracy of the computer precision, we measure the fraction of instances where VI converges. This fraction gives us an upper bound of how likely it is that a randomly generated instance can find an SSE in stationary policies via VI. We first ran VI (Algorithm 12) in randomly generated instances. If the algorithm does not detect that T is not contractive, then we run PI and compare the solution time of both algorithms. The parameters were generated as in the (MFS) case except that we use $k = 50$ per size.

The percentage of instances that are not solved in general games is represented in 3.14.

Results show that the more the set of states increases, the more likely it is to find a SG whose associated operator T is not contractive. As opposed to increasing the set of actions of the leader or the follower, where the impact seems to be smaller. We guess that the existence of only one matrix with *bad properties* can make T not contractive. If this is true the probability of T not being contractive, given that the cardinality of the space of states is n, is equal to $(1-p)^n$ where p is the probability of a *bad matrix* appearing. In our experiments we calculate that for payoff matrices generated as in the (MFS) case, this probablity for 10 per 10 matrices should be $p = 1.5\,\%$.

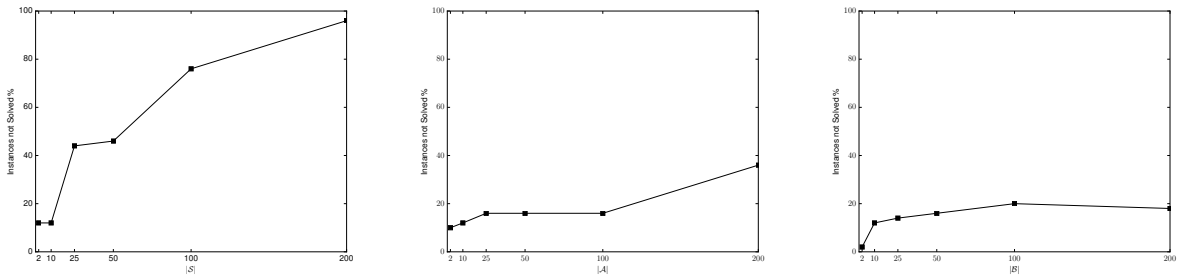


Figure 3.14: Percentage of instances where the Algorithm 12 returns undefined.

Performance of both algorithms in those instances where it is not detected that the operator T is not contractive is shown in Table 3.11 and Figure 3.15.

				Time[s]		Iterations [it]	
	n	m_A	m_B	VI	PI	VI	PI
Base Case	10	10	10	28.45	3.52	102.91	6.93
$\Delta\mathcal{S}$	2	10	10	5	0.63	94.43	6.57
	25	10	10	83.72	4.25	115.32	3.04
	50	10	10	119.93	7.39	74.96	2.22
	100	10	10	502.27	19.97	130.33	2.33
	200	10	10	362.03	61.18	35.5	2.5
$\Delta\mathcal{A}$	10	2	10	17.65	1.94	104	8.62
	10	25	10	36.39	6.15	118.29	7.14
	10	50	10	47.49	14.63	113.88	9.67
	10	100	10	46.14	19.36	71.22	6.83
	10	200	10	60.34	60.98	56.5	11.25
$\Delta\mathcal{B}$	10	10	2	10.71	1.22	69.41	6.02
	10	10	25	44.6	7.95	96.53	7.65
	10	10	50	109.37	16.08	106.02	7.17
	10	10	100	295.77	34.07	156.95	7.5
	10	10	200	978.23	90.57	95	8.41

Table 3.11: Resolution time and iterations performed by algorithms in General instances, for instances where T is not detected to be not contractive.

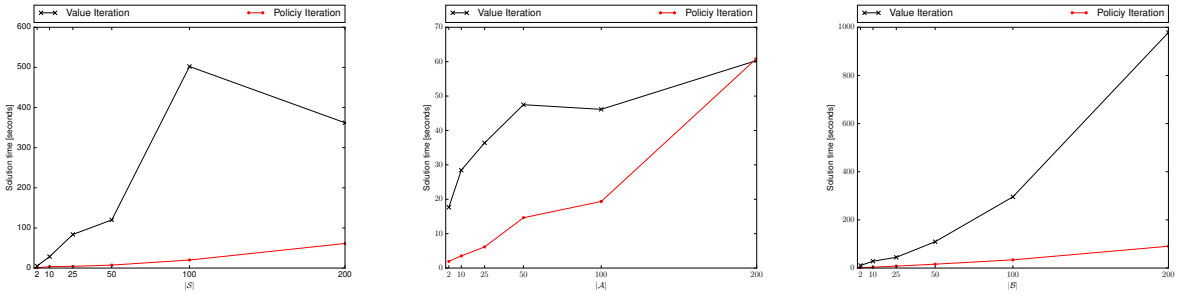


Figure 3.15: Performance of value function iteration and policy iteration in General random instances generated, for instances where it is not detected that the operator T is not contractive.

3.8.4. Sensitivity analysis in β

We analyze the impact of changing β_A on solution times of VI and PI algorithm for MFS instances. To do so, we randomly generate $k = 30$ payoff matrices and transition probabilities, and for each of these instances we consider different values of β_A, taking values in $\{0.1, 0.5, 0.75, 0.9, 0.95\}$. Figures 3.16 and 3.17 show the results of this experiment. As we expected, we note that as β_A increases, VI increases its solution times. This effect is not really clear in PI. Moreover, PI procedure for each β and size instance has constant iterations in each experiment.

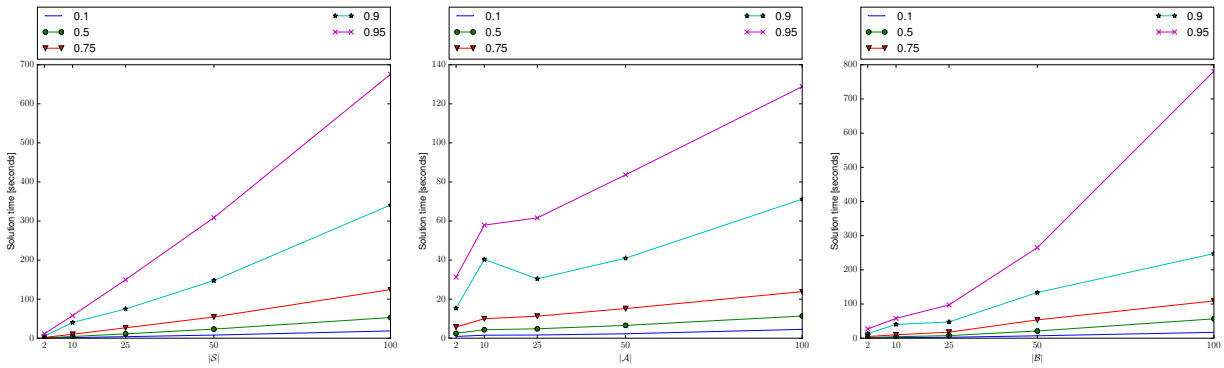


Figure 3.16: Sensitivity analysis in solution times for VI algorithm when β_A changes.

Besides, we study an instance where for some values of $\{\beta_A, \beta_B\}$ VI converges and for others does not. We called this example Numerical Example 3. The instance is described in Table 3.12. We test several values of β_A, $\beta_B \in [0, 1]$ labeling each experiment if it converged or not. Figure 3.18 shows the result of this experiment. As we can see, for $\beta_B \in [0.1, 0.65]$ the associated operator in VI seems to be not contractive. Moreover, β_A seems not to play any role in the fact if VI converges to a equilibrium or not.

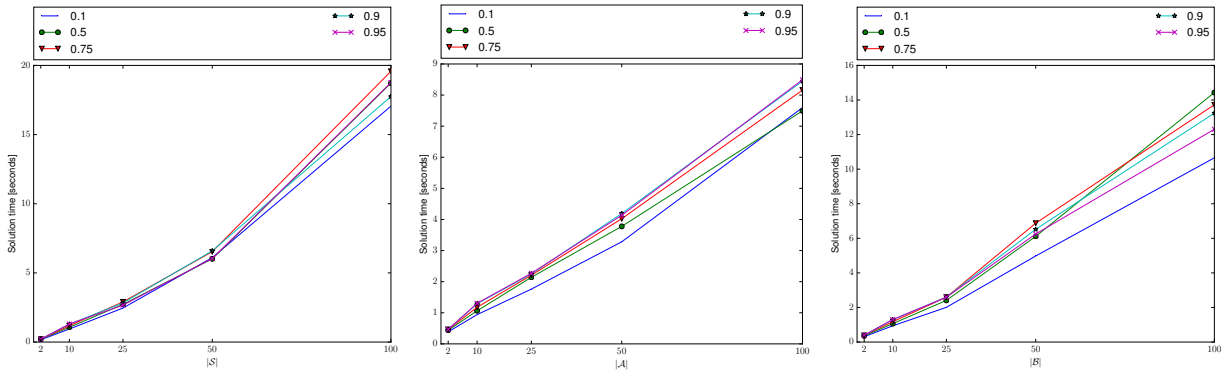


Figure 3.17: Sensitivity analysis in solution times for PI algorithm when β_A changes.

	b_1	b_2
a_1	(0,4, 0,6) / (0,-66)	(0,2, 0,8) / $(-56, -98)$
a_2	(0,3, 0,7) / (29,-81)	(0,8, 0,2) / $(-77, -72)$

State s_1

	b_1	b_2
a_1	(0.8, 0.2) / (-93,-59)	(0,2, 0,8) / $(-33, -65)$
a_2	(0.3, 0.7) / (-61,34)	(0,2, 0,8) / $(-47, 16)$

State s_2

Table 3.12: Transition matrix and payoffs for each player in the Numerical Example 3.

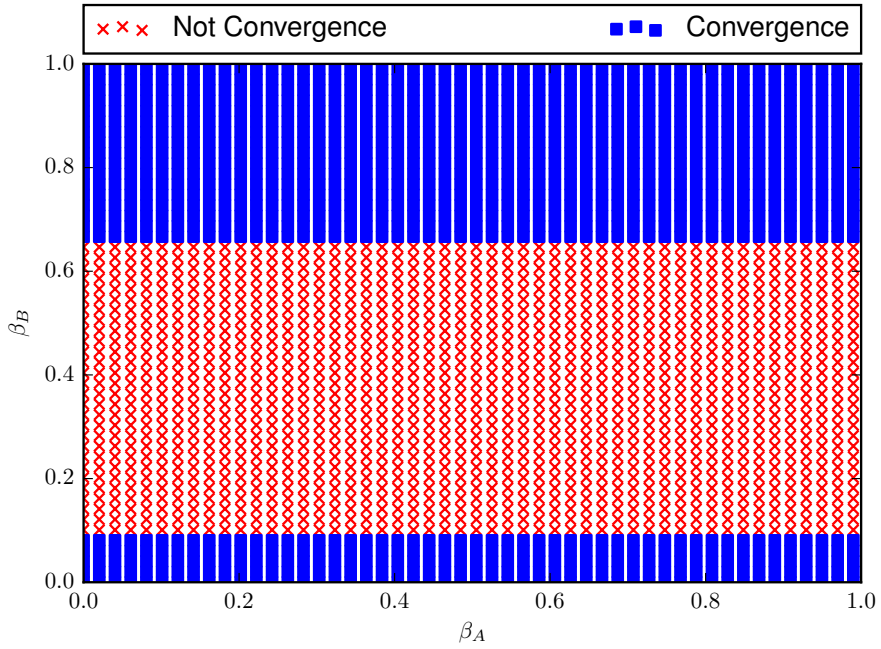


Figure 3.18: Impact of β_A and β_B on the convergence in VI in Numerical Example 3.

3.9. Conclusions and Future work

In this work we adapt dynamic programming based algorithms and mathematical programming formulations to find stationary and feedback policies forming SSE. We first show a family of SG where a unique equilibrium exists . In this case, we show that Value Iteration and Policy Iteration converge to a pair of value functions that corresponds to the value of a SSE. For the general case it is not always possible. We show an instance where VI does not converge. Our computational test show that PI outperforms VI in solution times in most of cases. We also discuss about an algorithm to detect if the operator proposed for the general case is contractive.

In future work we aim to extend the analysis to other families of SG where SSE can be achieved via dynamic programming. For these games, we would like to analyze the impact of approximate dynamic programming (see [49]) on computing this type of equilibrium. Also Rolling Horizon procedures are . A third research line is to analyze the impact of better implementations of PI and VI to improve the scalability of the algorithms exposed in this chapter. Finally, in future work we expect to formalize and understand the behavior of Cyclic policies forming strong Stackelberg equilibrium.

Chapter 4

Stackelberg games of water extraction

In this chapter we analyze an example in another domain where the methodology and analysis in Chapter 3 can be applied. We model the decision of a central agency who has to decide how to set prices to water extraction considering the actual level of the groundwater through time. We consider a setting where the leader is the central agency with a positive discount factor $\beta_A = \beta > 0$ and the followers are myopic agents ($\beta_B = 0$). The goal of the central agency is to maximize the expected discounted future utilities of these myopic followers. We also discuss stochastic and robust implementations of this problem. The models and analysis presented in this chapter form part of the working paper [34].

4.1. Introduction

A central agency has to decide the marginal cost that a set of K agents face when they have to decide the amount of water to extract from a shared groundwater. We consider a multi-stage Stackelberg setting where the central agency is the leader and the set of K agents are the followers. The dynamics of the groundwater is given by:

$$G_{t+1} = G_t + R_t - \sum_{i=1}^{K} u_t^i, \quad t \in \mathbb{N} \quad \text{with } G_0 \text{ given}, \tag{4.1}$$

where G_t is the water level at period t, R_t is the rainfall during period t and u_i^t is the water extracted by agent i at period t.

The utility function of each player is given by

$$\pi_t^i = F_i(u_t^i) - C_i\left(G_t + m_t R_t - n_t \sum_{i=j}^{K} u_t^j\right) \cdot u_i^t, \ t \in \mathbb{N}_0, \ i = 1, 2, \ldots, K, \tag{4.2}$$

where F is a production function that models the technology of agent i, C_i is the marginal

cost function, that depends on the level of water. Binary Controls m_t and n_t are used to determine if the level is measured before the rain at this period, or whether consider the total extraction of the agents or not.

Agents knowing controls $\{n_t,\, m_t\}$, at each time t each agent maximizes its instantaneous rewards considering the others players decision. In other words, agents observe m_t and n_t and then play a Nash equilibrium with payoffs π_t^i. This is equivalent to the case where the discount factor of each agent is $\beta_B = 0$.

On the other hand, the central agency maximizes the total discounted reward of the agents, controlling values of n_t, m_t. Then central agency faces the following multistage problem:

$$\max \bar{J}\left(\{n_t,\, m_t\}_t\right) = \sum_{t=0}^{+\infty} \beta_A^t \pi_t^i\left(n_t, m_t\right). \tag{4.3}$$

4.2. A first study

In this preliminary work we consider symmetric agents and production functions and marginal costs as:

$$F(u) = u - \frac{b}{2}u^2, \tag{4.4}$$

$$C(x) = z - cx > 0, \tag{4.5}$$

we call this case the linear-quadratic case. Also we assume a constant rainfall R. Given n_t, m_t the Nash equilibrium of the followers can be computed as

$$u_t = \underbrace{\frac{c}{b+(K+1)n_t c}}_{\alpha_t} G_t + \frac{1-z+cm_tR_t}{b+(K+1)n_t c} = \alpha\left[G_t + m_tR_t + \frac{1-z}{c}\right], \quad i = 1,\dots,K,\ t \in \mathbb{N}_0, \tag{4.6}$$

where we note that $\alpha_t = \alpha_t(n_t)$. The equilibrium utilities for each agent is given by:

$$\pi = u^2\frac{b+2n_tc}{2} = \pi_t, \quad i = 1,\dots,K,\ t \in \mathbb{N}_0. \tag{4.7}$$

Then, the leader faces the following optimal control problem:

$$\begin{aligned} &\max_{n_t,\, m_t} \sum_{t=0}^{+\infty} \beta_A^t K \pi_t, \\ &\text{s.t. } \pi_t = \alpha^2\left(G_t + m_tR_t + \tfrac{1-z}{c}\right)^2 \frac{b+2n_tc}{2} \end{aligned} \tag{4.8}$$

with a Bellman equation given by:

$$\begin{aligned} V(G) = \max_{n,m}\ & K\left(\frac{cG + cmR + (1-z)}{b+(K+1)nc}\right)^2 \frac{b+2nc}{2} \\ & + \beta_A V\left(G + R - K\frac{c}{b+(K+1)nc}G + \frac{1-z+cmR}{b+(K+1)nc}\right). \end{aligned} \tag{4.9}$$

Parameter	Value
b	1.0
c	0.6
z	0.9
R	1.0
K	4

Table 4.1: Parameters in our preliminar study.

Note that in this case, the consumption of steady state, denoted by $\bar{u}$, can be computed by $G = G + R - K\bar{u}$. Then $\bar{u} = \frac{R}{K}$, in the steady state, the agents consume only the rainfall at each period.

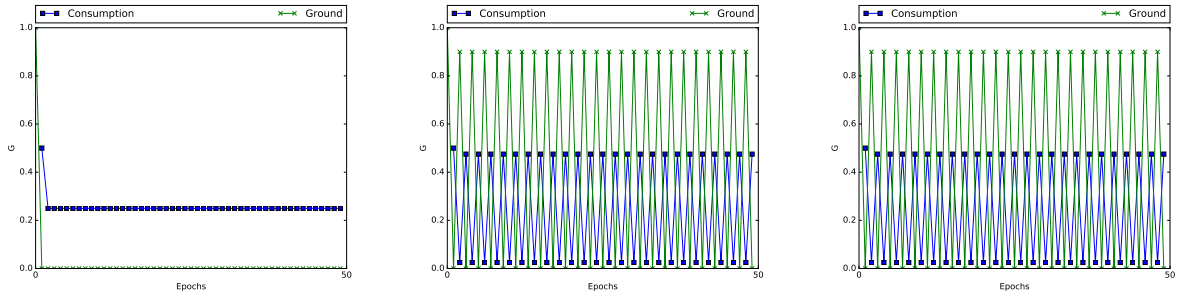
We can also analyze the case where agents are non-symmetric, where the Nash equilibrium is given by the following linear system:

$$(c_i n_t + b_i)u_t^i + c_i n_t \sum_{j=1}^{K} u_t^j = 1 - z_i + c_i G_t + c_i m_t R_t \quad i = 1, \ldots, K. \tag{4.10}$$

4.3. Preliminary results

In this section we show our preliminary results. We test the linear quadratic case with symmetric myopics agents, with the parameters listed in Table 4.1.

First, as we expected, β has a positive impact on the water trajectories. That is as β increases, water trajectories converge to a higher levels (see Figure 4.1).



a. $\beta_A = 0.2$ b. $\beta_A = 0.7$ c. $\beta_A = 0.9$

Figure 4.1: Groundwater level and consumption for $\beta_A \in \{0.2, 0.7, 0.9\}$.

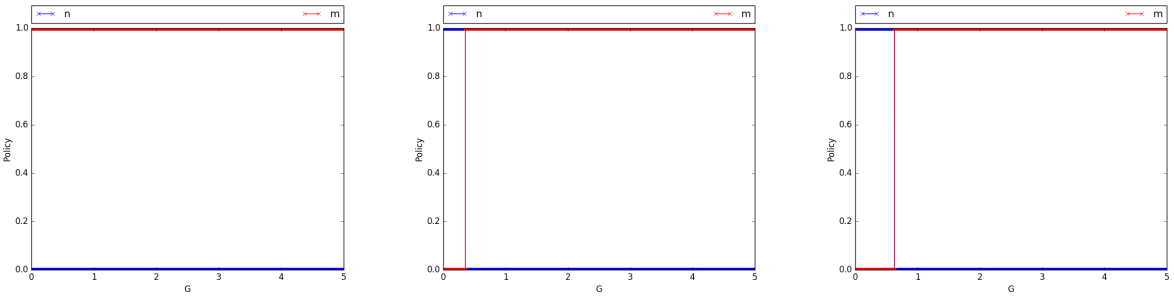
Note that in Figure 4.1.a. agents consume all the water available in each period. Farmers in each period has a consumption of steady state $\bar{u}$ and the steady state level of the groundwater is zero. When β_A increases to 0.7, Figure 4.1.b., the groundwater level fluctuates between 0 and 0.9, making also the agent's consumption fluctuates between two values, one higher than $\bar{u}$ and other lower than $\bar{u}$. This behavior is related with the optimal policy, changing at each step to levels of G where the optimal policies change.

Second, we found two types of optimal policies:

- Constant policies, that is for each level of water the optimal policies are $n_t = 0$ and $m_t = 1$ (see Figure 4.2a.). Given that β_A has low value has the consequence that the marginal cost of extracting water given by the optimal policy is low. This occurs when it is not considered the extraction of the agents but it is considered the rainfall in computing the marginal cost of extraction.
- Threshold policies (see Figure 4.2b. and 4.2.c), that is policies of the form:

$$n_t, m_t = \begin{cases} \underline{n}, \underline{m} & \text{if } G_t \leq \bar{G} \\ \overline{n}, \overline{m} & \text{otherwise.} \end{cases} \tag{4.11}$$

  That means, if the groundwater has a level lower than $\bar{G}$, then the agency increases the marginal cost. Otherwise, the marginal cost of extraction water is cheaper. Note that the optimal threshold $\bar{G}$ is an increasing function in the value of β_A.



a. $\beta_A = 0{,}2$ b. $\beta_A = 0{,}7$ c. $\beta_A = 0{,}9$

Figure 4.2: Policies for $\beta_A \in \{0{,}2,\ 0{,}7,\ 0{,}9\}$.

As future work we aim to prove that threshold policies are optimal and what are the conditions in the parameters of the dynamic to detect both, constant and threshold policies.

4.4. Robust approach

In a first attempt to introduce uncertainty in our model, we consider a robust optimization approach (see [7]) in the amount of rain that the leader predicts to fall. In other words, we do not consider R as a deterministic but we consider it as a parameter in an uncertainty set $\mathcal{R}_t$, and the leader protects himself from uncertainty by minimizing the impact of the worst outcome. We consider the following uncertainty set:

$$(\tilde{R}_t)_{t=0}^{\tau} \in \left\{ (\bar{R}_t + \Lambda_t)_{t=0}^{\tau} : -y_t\delta_t \leq \Lambda_t \leq y_t\delta_t,\ \sum_{t=0}^{\tau} y_t \leq \Lambda,\ y_t \in \{0,1\} \right\},$$

$\bar{R}_t$ represents the average rain at period t, and δ_t represents the maximum deviation that is observed in the rainfall at period t. To avoid being too conservative, we add an uncertainty budget Λ and we restrict the time steps that y_t takes values 1 at most Λ times.

Then, in our robust setting the leader problem can be stated as:

$$\max_{n_t,m_t} \min_{y_t} \sum_{t=0}^{+\infty} \beta_A^t K \pi(\tilde{R}_t, n_t, m_t). \tag{4.12}$$

In order to write a more friendly formulation, we rewrite the problem stated in (4.12):

$$\max_{n_t,m_t} \sum_{t=0}^{+\infty} \beta^t \pi_t + \min_{y_t} \sum_{t=0}^{+\infty} \beta_A^t \xi_t m_t \delta_t (m_t \delta_t - 2\mu) y_t \tag{4.13}$$

where $\xi_t = K\alpha^2 \left[\frac{b+2n_t c}{2}\right]$ and $\mu = G_t + m_t \bar{R}_t + \frac{1-z}{c}$. Now, we consider the problem associated to the second term of (4.13):

$$\begin{aligned} \min_{y_t} \sum_{t=0}^{+\infty} \beta^t \xi_t m_t \delta_t (m_t \delta_t - 2\mu) y_t & \\ \sum_t y_t \leq \Lambda & \\ y_t \leq 1 & \\ y_t \geq 0 & \end{aligned} \tag{4.14}$$

and its dual:

$$\begin{aligned} \max \Lambda\theta + \sum_t \lambda_t & \\ \theta + \lambda_t \leq \beta_A^t \xi_t m_t \delta_t (m_t \delta_t - 2\mu) & \quad (4.15) \\ \lambda_t,\, \theta_t \leq 0. & \quad (4.16) \end{aligned}$$

Using the fact that both problems have optimal finite solution, we can state the robust counterpart as:

$$\max_{m_t,n_t,\theta,\lambda_t} \quad \sum_{t=0}^{\tau} \beta_A^t \left(\frac{c}{b+(K+1)n_t c}\right)^2 \left(G_t + m_t \bar{R}_t + \tfrac{1-z}{c}\right) \left(\tfrac{b+2n_t c}{2}\right) + \Lambda\theta + \sum_{t=0}^{T} \lambda_t \tag{4.17}$$

$$\text{s.a.} \quad \theta + \lambda_t \leq \beta_A K \left(\frac{c}{b+(K+1)n_t c}\right)^2 \left(\tfrac{b+2n_t c}{2}\right) m_t \delta_t \left[m_t \delta_t - 2G_t - 2m_t R_t - 2\tfrac{1-z}{c}\right] \quad \forall t \ (4.18)$$

$$\theta, \lambda_t \geq 0 \quad \forall t \ (4.19)$$

$$0 \leq m_t,\, n_t \leq 1 \quad \forall t. (4.20)$$

We plan to design efficient ways to solve this problem. To do so, we note that constraints are separable on t. Then we can use Dynamic Programming to solve this formulation.

4.5. Conclusions and future (current) work

We model a decision problem where a central agency aims to control the groundwater level through the marginal cost. We present a Stackelberg stochastic discounted game, now with several followers playing Nash equilibrium, where the controls are whether the marginal cost depends on the rainfall and the agents extraction at each period t. We test our models in a linear quadratic setting with symmetric myopic agents. We show experimentally that optimal policies are constant or threshold based policies. Moreover, we can affect the water trajectories to higher stationary levels, even with myopic agents, by considering a central agency maximizing the social welfare. In order to introduce uncertainty in the rainfall, we propose a robust approach formulation of the central agency problem.

We aim in future work to find algorithms to solve the robust counterparts. Moreover, we expect to consider stochastic versions of this problem, considering the rainfall as random variable. We are studying two cases:

1. R_t is iid, taking values in $\overline{R}$ and $\underline{R}$, with probability distribution $\pi_R = \begin{bmatrix} p \\ (1-p) \end{bmatrix}$.
2. R_t is Markovian with states, $\{\overline{R}, \underline{R}\}$, representing cycles in climate with a transition matrix $Q = \begin{bmatrix} p & (1-p) \\ (1-q) & q \end{bmatrix}$.

Finally, we expect to compare both problems with uncertainty.

Conclusion

In this thesis we provided algorithms and models to solve Stackelberg Security Games outperforming the state of the art approaches, addressing problem size due to large scale set of strategies, targets in a geographical area or because the interaction between the leader and the follower is evolving through the time. Our main methodological contributions are twofold. We first developed decomposition methods based on a nice connection between the problem and the matching polyhedral structure. And second we provided new operators to analyze dynamic programming algorithms to compute Stackelberg equilibrium in Stochastic Games.

In Chapter 1 we studied a special type of SSG played on a network. In this game, a defender has to combine resources to do patrol labors in a set of targets. We proposed a novel formulation, (COMB), that represent the set of mixed strategies in a compact form by setting marginal probabilities over the possible pairings of team up resources and the targets to be covered. We proposed a method to retrieve an implementable strategy given the optimal marginals and we proved the validity of our formulation. In this method we proposed two ways to decompose fractional size constrained matchings as convex combination of pure matchings. (COMB) formulation has an exponentially large family of constraints. To scale-up the instances that solvers are able to optimize, we propose a cut generation algorithm.

Furthermore, we provided an alternative sampling method to recover an implementable defender strategy given the optimal coverage distributions. Computational tests have shown that the two-stage sampling method we describe, provides implementable strategies that do not deviate much from the optimal coverage distributions. Further computational tests have shown (COMB) to have smaller solution times and better scaling capabilities than the extensive formulation (D2) on randomly generated security instances. Both methods are considerably outperformed by the cut generation algorithm.

In addition, we described a real-life border patrol problem and have presented a parameter generation methodology that takes into account past crime data and geographical and social factors to construct payoffs for the Stackelberg game. Robustness tests have shown that the solutions our software provides are fairly robust to the networks we generate as well as to minor changes in the flow of crime along the border.

In future work we aim to extend our model and methodology to the case where each resource can be paired up with more than one resource, taking into account different capabilities, time schedules, and other considerations. We are also working on improving the parameter estimation for the real case exposed in this chapter.

In Chapter 2 we developed algorithms and heuristics to scale up algorithms in games that are applied to a large geographical area. In future work we aim to analyze the performance of our approximation algorithm

In Chapter 3 we adapted dynamic programming based algorithms and mathematical programming formulations to find stationary policies forming strong Stackelberg equilibrium. We first show a family of SG where this type of equilibrium exists. In this case, we show that VI and policy iteration converges to a pair of value functions that corresponds to the value of a SSE. For the general case it is not always possible. We show an instance where VI does not converge. Our computational test show that PI outperforms VI in solution times in most of cases. We also discuss about an algorithm to detect if the operator proposed for the general case is contractive.

In future work we consider how to extend the analysis to other families of stochastic games where strong Stackelberg equilibrium can be achieved via dynamic programming. For these games, we would like to analyze the impact of approximate dynamic programming (see [49]) on computing this type of equilibrium. A third research line is to analyze the impact of better implementations of PI and VI to improve the scalability of the algorithms exposed in this chapter. Finally, in future work we expect to formalize and understand the behavior of Cyclic policies forming strong Stackelberg equilibrium.

Finally, in Chapter 4 we model a decision problem where a central agency aims to control the groundwater level in steady state. We present a Stackelberg stochastic discounted game, where the controls are wheter the marginal cost depends on the rainfall and the agents extraction at each period t. We test our models in a linear quadratic setting with symmetric myopic agents. We show experimentally that optimal policies are constant or threshold based policies. Moreover, we can affect the water trajectories to higher stationary levels, even with myopic agents, by considering a central agency maximizing the social welfare. In order to introduce uncertainty in the rainfall, we propose a robust approach formulation of the central agency problem.

We aim in future work to find algorithms to solve the robust counterparts. We expect to consider stochastic versions of this problem, considering the rainfall as random variable.

Bibliography

[1] Aduanas de Chile. https://www.aduana.cl/importaciones-de-productos/aduana/2007-02-28/161116.html, 2016.

[2] Tansu Alpcan and Tamer Başar. *Stochastic security games*, page 74–97. Cambridge University Press, 2010.

[3] Eitan Altman, Konstantin Avratchenkov, Nicolas Bonneau, Mérouane Debbah, Rachid El-Azouzi, and Daniel Sadoc Menasché. Constrained stochastic games in wireless networks. In *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*, pages 315–320. IEEE, 2007.

[4] Rabah Amir. Stochastic games in economics and related fields: an overview. In *Stochastic Games and Applications*, pages 455–470. Springer, 2003.

[5] Tamer Basar, Geert Jan Olsder, and GJ Clsder. *Dynamic noncooperative game theory*, volume 200. SIAM, 1995.

[6] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 57–64. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[7] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[8] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[9] Darse Billings, Aaron Davidson, Jonathan Schaeffer, and Duane Szafron. The challenge of poker. *Artificial Intelligence*, 134(1-2):201–240, 2002.

[10] Jerome Bracken and James T. McGill. Mathematical programs with optimization problems in the constraints. *Operations Research*, 21(1):37–44, January 1973.

[11] Victor Bucarey, Carlos Casorran, Karla Rosas, Hugo Navarrete, and Fernando Ordóñez. Coordinated defender strategies for border patrols. 2017.

[12] Victor Bucarey, Eugenio Della Vecchia, Alain Jean-Marie, Mabel Tidball, and Fernando Ordóñez. Solving Stackelberg equilbrium in stochastic games. 2017.

[13] Victor Bucarey, Fernando Ordóñez, and Enrique Bassaletti. Shape and balance in police districting. In *Applications of Location Analysis*, pages 329–347. Springer, 2015.

[14] Arnaud Canu and Abdel-Illah Mouaddib. Collective decision-theoretic planning for planet exploration. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pages 289–296. IEEE, 2011.

[15] Yanling Chang, Alan L Erera, and Chelsea C White III. A leader-follower partially observed, multiobjective markov game. *Annals of Operations Research*, 235(1):103, 2015.

[16] Comisión Económica para América Latina y el Caribe. Costo económico de los delitos, niveles de vigilancia y políticas de seguridad ciudadana en las comunas del gran santiago. http://www.cepal.org/es/publicaciones/7258-costo-economico-de-los-delitos-niveles-de-vigilancia-y-politicas-de-seguridad, 2000.

[17] Vincent Conitzer. Should Stackelberg mixed strategies be considered a separate solution concept? 2014.

[18] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90. ACM, 2006.

[19] Council of the European Union. http://www.consilium.europa.eu/en/press/press-releases/2016/09/14-european-border-coast-guard/, 2016.

[20] George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Oper. Res.*, 8(1):101–111, February 1960.

[21] Francesco Maria Delle Fave, Albert Xin Jiang, Zhengyu Yin, Chao Zhang, Milind Tambe, Sarit Kraus, and John P Sullivan. Game-theoretic patrolling with dynamic execution uncertainty and a case study on a real transit system. *Journal of Artificial Intelligence Research*, 50:321–367, 2014.

[22] Department of Homeland Security of the United States. https://www.dhs.gov/border-security, 2016.

[23] Steven P Dirkse and Michael C Ferris. The PATH solver: a non-monotone stabilization scheme for mixed complementarity problems. *Optimization Methods and Software*, 5(2):123–156, 1995.

[24] Jack Edmonds. Maximum matching and a polyhedron with 0, l-vertices. *J. Res. Nat. Bur. Standards B*, 69(1965):125–130, 1965.

[25] Julius Farkas. Theorie der einfachen ungleichungen. *Journal fur die reine und angewandte Mathematik*, 124:1–27, 1902.

[26] Michael C Ferris and Todd S Munson. Complementarity problems in GAMS and the PATH solver. *Journal of Economic Dynamics and Control*, 24(2):165–188, 2000.

[27] Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer Science & Business Media, 2012.

[28] Janos Flesch, Frank Thuijsman, and Koos Vrieze. Cyclic Markov equilibria in stochastic games. *International Journal of Game Theory*, 26(3):303–314, 1997.

[29] Dan Gusfield. Very simple methods for all pairs network flow analysis. *SIAM Journal on Computing*, 19(1):143–155, 1990.

[30] Onésimo Hernández-Lerma and Jean-Bernard Lasserre. *Discrete-time Markov control processes: basic optimality criteria*, volume 30. Springer Science & Business Media, 2012.

[31] Dorit S Hochbaum, Cheng Lyu, and Fernando Ordóñez. Security routing games with multivehicle chinese postman problem. *Networks*, 64(3):181–191, 2014.

[32] Manish Jain, Erim Kardes, Christopher Kiekintveld, Fernando Ordoñez, and Milind Tambe. Security games with arbitrary schedules: A branch and price approach. In Maria Fox and David Poole, editors, *AAAI*. AAAI Press, 2010.

[33] Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordóñez. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces*, 40(4):267–290, July 2010.

[34] Alain Jean-Marie, Mabel Tidball, Fernando Ordóñez, Ma. Evangelina Alvarez, Victor Bucarey, and Eugenio Della Vecchia. Stackelberg games of water extraction. 2017.

[35] Jörg Kalcsics, Stefan Nickel, and Michael Schröder. Towards a unified territorial design approach: Applications, algorithms and GIS integration. *Top*, 13(1):1–56, 2005.

[36] Michael Kearns, Yishay Mansour, and Satinder Singh. Fast planning in stochastic games. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 309–316. Morgan Kaufmann Publishers Inc., 2000.

[37] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pages 689–696, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

[38] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03 1951.

[39] George Leitman. On generalized Stackelberg strategies. *J. Optim. Theory Appl.*, 26(4):637–643, 1978.

[40] Joshua Letchford, Liam MacDermed, Vincent Conitzer, Ronald Parr, and Charles L Isbell. Computing optimal strategies to commit to in stochastic games. In *AAAI*, 2012.

[41] Zhi-Quan Luo, Jong-Shi Pang, and Daniel Ralph. *Mathematical programs with equilibrium constraints*. Cambridge University Press, 1996.

[42] Ministerio del Trabajo y Previsión Social. http://www.leychile.cl/Navegar?idLey=20763, 2016.

[43] Kien C Nguyen, Tansu Alpcan, and Tamer Basar. Stochastic games for security in networks with interdependent nodes. In *Game Theory for Networks, 2009. GameNets' 09. International Conference on*, pages 697–703. IEEE, 2009.

[44] Fernando Ordónez, Milind Tambe, Juan F Jara, Manish Jain, Christopher Kiekintveld, and Jason Tsai. Deployed security games for patrol planning. In *Handbook of Operations Research for Homeland Security*, pages 45–72. Springer, 2013.

[45] Manfred W Padberg and M Ram Rao. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7(1):67–80, 1982.

[46] Praveen Paruchuri, Jonathan P. Pearce, Janusz Marecki, Milind Tambe, Fernando Ordóñez, and Sarit Kraus. Playing games for security: An efficient exact algorithm for solving bayesian Stackelberg games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '08, pages 895–902, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[47] Hans Peters. *Game theory: A Multi-leveled approach*. Springer, 2015.

[48] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, pages 125–132. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[49] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[50] Martin L Puterman. *Markov decision processes*. Wiley-Interscience, 1994.

[51] QGIS. *QGIS Geographic Information System*. Open Source Geospatial Foundation, 2009.

[52] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.

[53] Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.

[54] Hanif D Sherali and Frederick L Nordai. Np-hard, capacitated, balanced p-median problems on a chain graph with a continuum of link demands. *Mathematics of Operations Research*, 13(1):32–49, 1988.

[55] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. Protect: A deployed game theoretic system to protect the ports of the united states. In SC Richland, editor, *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, volume 1 of *AAMAS '12*, pages 13–20, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[56] Olivier Sigaud and Olivier Buffet. *Markov decision processes in artificial intelligence*. John Wiley & Sons, 2013.

[57] Milind Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

[58] Mabel M Tidball and Eitan Altman. Approximations in dynamic zero-sum games i. *SIAM journal on control and optimization*, 34(1):311–328, 1996.

[59] Jason Tsai, Christopher Kiekintveld, Fernando Ordonez, Milind Tambe, and Shyamsunder Rathi. Iris-a tool for strategic security allocation in transportation networks. 2009.

[60] H. von Stackelberg. *Principios de teoria economica*. Oxford University Press, New York (USA), 1952.

[61] Yevgeniy Vorobeychik, Bo An, and Milind Tambe. Adversarial patrolling games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1307–1308. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[62] Yevgeniy Vorobeychik, Bo An, Milind Tambe, and Satinder Singh. Computing solutions in infinite-horizon discounted adversarial patrolling games. In *Proc. 24th International Conference on Automated Planning and Scheduling (ICAPS 2014)(June 2014)*, 2014.

[63] Yevgeniy Vorobeychik and Satinder Singh. Computing Stackelberg equilibria in discounted stochastic games (corrected version). 2012.

[64] OJ Vrieze, SH Tijs, TES Raghavan, and JA Filar. A finite algorithm for the switching control stochastic game. *Operations-Research-Spektrum*, 5(1):15–24, 1983.

[65] Okko Jan Vrieze. Stochastic games with finite state and action spaces. *CWI tracts*, 33:1–221, 1987.

[66] Richard A Waltz and Jorge Nocedal. Knitro user's manual. *Northwestern University, Evanston, Illinois, Technical Report OTC-2003/5*, 2003.

[67] Rong Yang, Benjamin Ford, Milind Tambe, and Andrew Lemieux. Adaptive resource

allocation for wildlife protection against illegal poachers. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2014.

[68] Rong Yang, Albert Xin Jiang, Milind Tambe, and Fernando Ordoñez. Scaling-up security games with boundedly rational adversaries: A cutting-plane approach. In Francesca Rossi, editor, *IJCAI*. IJCAI/AAAI, 2013.

[69] Rong Yang, Christopher Kiekintveld, Fernando Ordóñez, Milind Tambe, and Richard John. Improving resource allocation strategy against human adversaries in security games. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 458, 2011.

[70] Rong Yang, Fernando Ordóñez, and Milind Tambe. Computing optimal strategy against quantal response in security games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 847–854. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[71] Zhengyu Yin, Albert Xin Jiang, Milind Tambe, Christopher Kiekintveld, Kevin Leyton-Brown, Tuomas Sandholm, and John P. Sullivan. Trusts: Scheduling randomized patrols for fare inspection in transit systems using game theory. *AI Magazine*, 33(4):59–72, 2012.

[72] Zhengyu Yin and Milind Tambe. A unified method for handling discrete and continuous uncertainty in bayesian Stackelberg games. In Winikoff Conitzer and van der Hoek, editors, *AAMAS*, 2012.

[73] Chao Zhang, Victor Bucarey, Ayan Mukhopadhyay, Arunesh Sinha, Yundi Qian, Yevgeniy Vorobeychik, and Milind Tambe. Using abstractions to solve opportunistic crime security games at scale. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 196–204. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[74] Chao Zhang, Arunesh Sinha, and Milind Tambe. Keeping pace with criminals: Designing patrol allocation against adaptive opportunistic criminals. In *Proceedings of the 2015 international conference on Autonomous agents and multiagent systems*, pages 1351–1359. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[75] Martin Zinkevich, Amy Greenwald, and Michael L Littman. Cyclic equilibria in markov games. In *Advances in Neural Information Processing Systems*, pages 1641–1648, 2006.