POLITECNICO DI MILANO
Corso di Laurea in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

Algorithms for Finding Leader-Follower Equilibrium with Multiple Followers

AI & R Lab
Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Relatore: Ing. Nicola Gatti
Correlatore: Ing. Stefano Coniglio

Tesi di Laurea di
Stefano Conti, matricola 736590

Anno Accademico 2012-2013

Contents

List of Figures

List of Tables

Abstract

The problem of ensuring security in crowded areas as well as areas of economic or political relevance has become a matter of growing interest in the last few years. The limited amount of resources i.e. money, personnel and equipment, makes it prohibitive to provide a complete coverage of the threatened targets at all the time, hence an efficient resource allocation policy is needed. Moreover, any potential attacker would be able to observe the defender's strategy before choosing a target in order to exploit any potential weakness.

A new branch of Game Theory, namely Security Games, has been successfully applied to these multiagent problems in the last decade, as a way to calculate an optimal defense policy. In this work we provide the state of the art of Security Games describing issues and theories at the basis of the algorithms which are the core of most advanced security software. A critical assumption of those algorithms is the hypothesis that only a single attacker could decide to attack a structure at the same time, then, in the second part of the work, we drop this assumption allowing the existence of multiple concurrent attackers. Under this general assumption we analyze different scenarios in which defender and attackers are allowed or not to play mixed strategies by providing an analysis of the difficulty of finding an optimal solution in the different cases.

Sommario

Negli ultimi anni il problema di garantire la sicurezza di luoghi affollati o importanti dal punto di vista politico o economico è molto cresciuta. Questi luoghi sono costantemente esposti alla minaccia di organizzazioni terroristiche o di persone imprevedibili e pericolose o al contrabbando di armi e droghe. Garantire la sicurezza di questi luoghi spesso richiede cospicui investimenti, in termini di personale qualificato e mezzi sofisticati. Questi costi rendono economicamente meno efficienti queste infrastrutture, per esempio stadi, porti, aeroporti o stazioni ferroviarie, che devono quindi scaricare i costi direttamente sugli utenti o sull'intera comunità, riducendo inoltre la massima capacità di flusso della struttura. Il più delle volte è quindi impossibile garantire in ogni istante una protezione completa dell'intera area, poiché costi e ritardi risulterebbero insostenibili.

Diventa dunque necessario elaborare una politica di allocazione delle risorse che garantisca adeguati standard di sicurezza pur riducendo al minimo i costi. Tuttavia, l'allocazione di un numero limitato di risorse su un'area estesa o un perimetro può introdurre debolezze critiche nel sistema di sicurezza che il più delle volte un eventuale aggressore può scoprire semplicemente osservando il sistema dall'esterno, senza ricorrere a strumenti d'indagine costosi e rischiosi.

Un approccio scientifico a questo tipo di problemi ci viene fornito da un nuovo ramo della Teoria dei Giochi, sviluppatosi negli ultimi anni, i *security games*, il cui scopo è quello di calcolare la strategia di difesa ottima in scenari multiagente nei quali un difensore debba proteggere diversi bersagli dall'aggressione di uno o più attaccanti. Nella prima parte di questo lavoro ci occuperemo di fornire una dettagliata sintesi dello stato dell'arte dei security games, analizzandone la storia e i problemi e le teorie che stanno alla base degli algoritmi che costituiscono il cuore dei principali sistemi informatici progettati per difendere strutture reali, come per esempio l'aeroporto internazionale di Los Angeles.

Tutti gli algoritmi finora proposti, però, si fondano sull'ipotesi che un solo aggressore possa agire nello stesso momento, applicando l'equilibrio di Stackelberg come concetto di soluzione. Nella seconda parte del lavoro estenderemo il concetto di security game al caso multifollower, ovvero il caso in cui più di un attaccante possa agire contemporaneamente, proponendo diversi approcci risolutivi a seconda che i giocatori possano o meno giocare in strategie miste e a seconda che gli attaccanti tendano o meno a collaborare con il leader. Infine, nell'ultima parte del lavoro, eseguiremo un'analisi delle prestazioni degli algoritmi proposti, eseguiti su istanze di giochi random.

Chapter 1

Introduction

"Leadership is the art of getting someone else to do something you want done because he wants to do it."

Dwight D. Eisenhower

The problem of ensuring security in crowded areas as well as areas of economic or political relevance has become a matter of growing interest in the last few years. These scenarios are often large in scale such as coasts and ports [2], airports [35, 36], urban road networks [19], transportation networks [40] and petroleum or electricity distribution systems [7]. The critical areas of these systems often correspond to large vulnerable infrastructures whose disruption would deteriorate the whole system's performance, causing a large loss of money for each hour of disservice. This, combined with the fact that an attack to these infrastructures would potentially result in a large number of dead and wounded, makes necessary the investment of large amounts of money to provide an adequate level of security to vulnerable areas. The budget for this purpose is often limited, while to protect these areas, many different security activities must be performed, depending on the domain, for example canine patrols, passenger screening, traffic checkpoints etc. The limited amount of resources i.e. money, personnel and equipment, makes it prohibitive to provide a complete coverage of the threatened areas at all the time, hence an efficient resource allocation policy is needed. However, deterministic scheduling of resource allocations would allow a hypothetical attacker to observe the defender strategy and possibly exploit any predictable pattern in it, in order to launch a better planned attack.

A randomized scheduling is therefore needed to prevent pattern predictions but a uniform randomization strategy would not take into account neither the different values of each area nor the intelligent behavior

of the attacker. Conventional risk analysis consists on calculating a fixed probability distribution over the targets, applying, for example, the Threat Vulnerability and Consequence (TVC) model [22]. In this type of analysis the allocation of resources in defense of the targets is weighted on the value of the target, its vulnerability and the probability of being attacked. However even these methods fail to model the rational behavior of the aggressor, thus reducing the efficiency of the solution for the defender. In fact, the attacker is supposed to be able to observe the defender strategy and he would choose to attack the target that maximizes his own expected utility function, so, in order to discourage the aggressor from attack, the target with the maximum utility should have an utility lower than the attacker's costs. In most cases, considering the vulnerability of the targets and the relatively low costs for the attacker, this would mean an unaffordable investment of resources for the defender.

Game Theory provides some attractive techniques to optimally solve these multiagent problems, finding an optimal unpredictable resource allocation strategy, by taking into account both the different value of each target and the rational and adaptive behavior of the aggressor. Solution concepts such as Nash equilibrium and Stackelberg equilibrium has been found appropriate to determine an optimal defender strategy in security domains thanks to the relative ease of modeling a real world scenario with a two-player game and the quality of the solution provided.

The main problem when trying to apply Game Theory to security problems is the large size of typical real-world problems. In fact the size of the action space of the defender grows combinatorially with the number of the targets to protect and the resources available, making necessary the study of more efficient algorithms. The large size makes also difficult to create the game matrix, for example when payoffs are periodically manually entered by a user, also introducing solution inefficiencies due to potential human errors. Finally, most of the real-world scenarios present hard scheduling constraints making the problem far more difficult to solve.

In the first part of this work we describe the nature of security problems and the Stackelberg equilibrium solution concept, then we present a summary of the most efficient algorithms recently designed to deal with these problems and some real-world applications of these algorithms.

- Chapter 2: we introduce some Game Theory concepts necessary to understand the following of this work, focusing in particular on commitment.

- Chapter 3: we present a mathematical formulation of the leader-follower problem affording also Bayesian extension and compact form representation. At the end of the chapter we briefly address the complexity issue.

- Chapter 4: we describe the details of the algorithms proposed in recent years for solving complex real world security problems. Later in the chapter we also describe the most famous algorithms for Nash equilibrium calculation because it will be used as part of multi-follower equilibrium, when the leader plays in pure strategies.

- Chapter 5: we describe the most relevant examples of the application of security games to real world large-scale problems, which rely on the algorithm described in the previous chapter.

- Chapter 6: we analyze how the problem changes when the defender has to deal with more than one single attacker and we propose some algorithms to calculate an optimal defender strategy. We start with formulating some hypothesis about the multi-follower scenario such as players' behavior and payoffs' structure. We then analyze four different problems according to whether the players are allowed to play in mixed strategies or not. Finally we restrict the multi-follower scenario to polymatrix games analyzing how the problem formulation changes. In the last part of the work we provide an analysis of the performance of the proposed multi-follower algorithms over uniform random game instances.

Chapter 2

Security problems

Security problems concern a wide range of very different scenarios with heterogeneous security activities, spacing from computer network security to border patrolling. However, an issue common to all the security problems is the resource allocation task [21]. For example an agent tasked to protect a urban road network aims to find the allocation of the minimum number of road checkpoints that provides the maximum level of protection [19], while an agent responsible for patrolling a perimeter tries to minimize the probability of intrusion as well as the number of agents deployed [3].

In all these problems it is important to find the right compromise between the costs of the system and the level of security provided. In more critical scenarios, such as national security scenarios, the level of security is fixed and the task of the planners is to minimize the number of resources needed to satisfy those requirements. In less critical scenarios, instead, when security is important but its price has to be small compared to the economical relevance of the activity to protect, the number of resources is limited and the task is to optimally allocate them to provide the highest level of security possible. Optimization gets growing importance as the cost of the single resource grows, as in the missile ballistic defense scenario [6], as well as the complexity of the system increases, such as in the FAMS domain [40]. In this work we focus the attention on the latter type of problem, which is harder to solve and then more interesting to study.

The peculiarity of security problems compared to classical optimization problems is the rational behavior of the attacker. This means that the attacker reasons not only about the value of the targets or the risk associated to them, but he's also able to reason about the defender's expected behavior. Knowing this, the defender can reason about the strategy that the attacker would probably adopt, in order to maximize his own expected utility.

2.1 Non-cooperative games

The class of *non-cooperative* games, introduced by Nash [30] in 1950, includes a wide range of problems where two or more self-interested agents act independently pursuing their own interests. This means that the agents only try to maximize their own utility regardless of the payoffs obtained by the other players. Sometimes this implies some kind of collaboration between players which is however the result of two (or more than two) players' selfish behavior and it cannot be enforced in no other way than changing some of the payoffs of the game. The world is full of examples of this kind of games making Nash's Theory at the center of many disciplines, from Economy to Social Science.

2.1.1 *Pursuit-evasion* games

A more specific family of non-cooperative games, called *pursuit-evasion* games, was described by Isaacs [17] in 1965. In these games, two players, generally called patroller and evader, compete to maximize their own utility: the evader tries to minimize the probability of detection, while the patroller is tasked to capture the evader. In the first formulation this game model has been largely applied to differential games with continuous time and variables, such as missile avoidance in air and naval combat [17] . Later, in 1976, Parsons presented a discrete version of *pursuit-evasion* games, where the players move in turn over a graph [33]. This family of games covers a wide range of different problems which can be roughly classified on the basis of the mobility of the players, as follows:

- *search games* with mobile patroller and immobile evader. In this game model the patroller inspects the environment, searching the evader, which cannot move from his cache. This model is useful when the evader finds difficulties while moving in the world, while the patroller has a significantly higher mobility. One example of a real-world application of this game model in security scenarios is the scheduling of bomb-sniffing canine patrols at Los Angeles International Airport (LAX) [35] when dealing with fixed bombs.

- *ambush games* with immobile patroller and mobile evader. In this model the evader tries to attack one or more targets by moving over a graph, while the patroller allocates his immobile resources over some arcs or nodes of the graph. This model is useful when the patroller cannot move his resources frequently, due, for example, to the costs in

terms of money or time. Two examples of real-world applications in security problems are ARMOR, which schedules road checkpoints at LAX [35] and RUGGED project which is still at an embryonic stage [19].

- *hider-seeker games* and *infiltration games* with mobile patroller and mobile evader. In these models the evader tries to reach a target protected by a mobile patroller. These models are widely used especially in perimeter patrolling and intrusion detection. One example of real-world application of the *hider-seeker* game model is IRIS [40] which is tasked to schedule Federal Air Marshals (FAMS) patrols onboard of U.S. flights in order to deter hijackers and terrorists. An example of application of the *infiltration* game model is, again, the scheduling of canine patrols in ARMOR [35] when facing suicide bombers or other mobile aggressors.

2.2 Commitment

The classical formulation of these game models assumes the players to plan their strategies simultaneously. Under this assumption, the best-known solution concept is *Nash equilibrium* [30].

Definition 1 (Nash equilibrium)**.** *A* Nash equilibrium *is a strategy such that, if played by all the players, no one gets advantage by defecting individually from that strategy.*

However, in most of the real-world security problems, the aggressor is supposed to observe the strategy of the defender before choosing his attack strategy. Moreover, in most of the cases the aggressor is able to collect all necessary information from public sources, without the need of resorting to costly or illegal means [7], making useless, or at least less effective, any attempt to hide the defense policy. At first glance this appear to be a great advantage for the attacker, which can accurately plan his strategy to achieve the best result for himself. Actually, by definition, if the attacker observes the defender playing the Nash equilibrium, he will play Nash equilibrium too and the defender is not damaged by the attacker ability to observe. However there could be more than one Nash equilibrium, hence the problem of choosing the equilibrium with the highest expected reward arises. Moreover, in general-sum games, supposed the ability of the attacker to observe, even the best Nash equilibrium may not be the optimal solution concept, thus making the defender needing more resources to improve his expected utility up to

the levels required by the application domain. Despite these unfavorable settings, the ability of the attacker to observe can be read as the ability of the defender to commit to a strategy, without altering the nature of the problem, making possible the application of better solution concepts.

The advantage of commitment ability has been shown for the first time by Heinrich Freiherr von Stackelberg (1934) who applied it to the Cournot's duopoly model [42]. In this model two firms have to plan an optimal production policy by taking into account the other firm's production policy as well as the market demand. In particular, while Cournot assumed two firms which choose their production quantities simultaneously, Stackelberg showed that if an incumbent company, named *leader*, has to change its production policy as another company, the *follower*, entries in the market, she would take advantage on committing to a certain production quantity, such that the best response of the follower would maximize the leader's profit. The optimal strategy to commit to is called *Stackelberg equilibrium* or *leader-follower* equilibrium. However the equilibrium concept proposed by Stackelberg was defined only in pure strategies. This is not good for security problems in general, because if the defender allocates its resources deterministically, under the common assumption of limited resources, the aggressor can safely attack one of the unprotected targets, or the target with the best expected payoff in general.

The advantage of committing to mixed strategies was first shown by Maschler [28] in 1966, who applied it to *inspection games* which are a modified version of search games. The application of Stackelberg equilibrium concept to security games has started to receive significantly more attention since, in 2004, von Stengel and Zamir [43] demonstrated not only that a leader-follower equilibrium always exists, although under certain restrictions, but also that committing to mixed strategies in two-players games never hurts the leader. In fact, the leader is always allowed to commit to a Nash equilibrium mixed strategy, which always exists, thus making the leader-follower equilibrium never worse than the best Nash equilibrium of the game. Hence the following proposition:

Proposition 1. *In a leader-follower two-player game the utility obtained by leader in the best Nash equilibrium represents a lower bound for the value of the leader-follower equilibrium.*

For example consider a simple instance of inspection game where an inspector (the leader) can choose if to inspect (I) or not (N), while the inspectee can choose if to act legally (l) or to cheat (c). The payoff matrix of this game is shown in Table 2.1. When the inspectee acts legally, the

inspector has to pay a cost for the inspection, while he pays nothing if he decides to not inspect. As we can see by observing the first column, the inspectee is not affected by the inspector strategy when he decide to act legally. However, if the inspector decides to never inspect, the inspectee would have an incentive to cheat and the inspector would be damaged from this. So, the inspector should decide to inspect, sometimes, in order to dissuade the inspectee from cheating and to partially repay the costs of the other vain inspections as well as the costs of undetected cheating.

	a	b
A	4 3	1 -3
B	4 -3	-1 -4

Figure 2.1: An instance of inspection game.

This game has only one Nash equilibrium in mixed strategies where the inspector decides to inspect with probability $\frac{1}{10}$ and the inspectee cheats with probability $\frac{1}{4}$. In this equilibrium the expected utility for the inspector is -2 while the expected utility of the inspectee is 1. It is interesting to note that in this equilibrium, although the inspectee sometimes cheats, damaging the inspector, his utility is exactly the same he would get by acting always legally, which could seem to be a slightly irrational behavior. Anyway if he would do so, the inspector would be induced to never inspect. From this absence of coordination comes the will of the inspectee to cheat even without a direct utility improvement for him.

Now let us see how the outcome of this game would be improved for the inspector by giving him the ability to commit. Immediately we note that the inspector, now the leader, could improve his utility by committing to a strategy where he always inspects. In this case the inspectee, that now is the follower, would prefer to always act legally. In this leader-follower equilibrium the expected utility of the leader is -1, which is better than the value of the Nash equilibrium calculated before. Moreover we can see that if the leader commits to a mixed strategy he could further improve his utility. For example if he commits to a mixed strategy where he inspects with probability 0.99, he would get an expected utility of -0.99, while the follower still prefers to act always legally. In general we observe that the leader can change his strategy until the follower is not induced to cheat. This limit strategy is $I = \frac{1}{10}$, which in this case coincides with the Nash equilibrium strategy, although this is not true in general. The great difference between

this Stackelberg equilibrium and the Nash equilibrium calculated before is that now the follower will never choose to cheat, because he would obtain a lower utility. Actually in this strategy the follower is indifferent between l and c, but he still chooses to always act legally. In fact, if the follower decides to cheat with a certain probability, he would not take advantage from it while the leader would result damaged. However, if the leader plays $I \geq \frac{1}{10} + \epsilon$, with ϵ arbitrarily small, the follower is induced to always act legally, while the expected utility of the leader substantially does not change. This property is known as *compliance*, and it is the basis of the commitment theory [28], providing stability to Stackelberg equilibrium.

2.2.1 Limitations of commitment

As we seen in the previous paragraph Stackelberg equilibrium qualities match very well to the characteristics of most real-world security problems, making it the more natural solution concept to this kind of problems. Nevertheless, there are some conditions under which the advantage of committing no longer holds and committing could even damage the leader. In this section we describe three cases which are of interest in the domain of security problems.

- *Games with more than two players.* Security problems with more than one attacker can be modeled as games where only the first player, the defender, has the ability to commit, while the other players play a simultaneous game. In a strictly competitive game this equals to a game where the first player induces a certain subgame by committing to a mixed strategy and the remaining players play according to the Nash equilibrium solution concept. However, as shown by Zamir and von Stengel, there are some games, such as *team games*, where commitment generally hurts the leader because it allows the followers to cooperate in order to achieve the best payoff for them. In this case the property for which the payoff of the leader in the leader-follower equilibrium is never worse than the payoff in the best Nash equilibrium of the game is not valid and in general this property no longer holds for games with more than two players [43].

- *Games with irrational adversaries.* All the solution concepts of Game Theory such as Nash equilibrium and Stackelberg equilibrium rely on the assumption that all the players are rational and selfish. The rationality of a player strongly relies on the assumption that he perfectly knows his own payoffs in all the states of the game and he chooses his strategy in order to maximize his own payoff. Moreover to be able to effectively

commit to a strategy the leader must also know the payoffs of the follower. This concept is known as *common knowledge*. However, in real-world security problems this assumptions could be not always true. The attacker may not always be a rational agent or the players may be rational but not able to exactly evaluate the payoffs of the adversary in every state of the game. Both of these conditions result in a mismatch in the common knowledge damaging the goodness of the committed strategy. To solve this problem, McKelvey and Palfrey [29] introduced quantal response model (QR) to effectively predict human behavior. This model was later adopted in the PROTECT system to depict rationality imperfections typical of human like reasoning [2], providing an increased robustness to the solution with respect to the perfect rationality model.

- *Games with uncertain observability.* As we said, one assumption on which Stackelberg equilibrium rely is the common knowledge concept, according to which both the players are mutually aware of the payoffs of the opponent. Actually, Stackelberg equilibrium does not need the follower to know all the payoffs of the leader in all the states of the game, but he only needs to know his own payoffs and the strategy to which the leader committed to. Nash equilibrium, instead, needs also the follower to know all the leader's payoffs in order to calculate the equilibrium strategy. This less restrictive condition could be a point in favor of Stackelberg equilibrium against Nash equilibrium. However, the whole concept of leader-follower equilibrium relies on the ability of the follower to observe the leader strategy. In real-world security problems it may be difficult or costly for the follower to exactly analyze the leader's security policy, think for example of the FAMS domain with undercover Federal Air Marshal deployed on a national scale. In such situations the leader may prefer to play a simultaneous game according to the Nash equilibrium solution concept. An algorithm proposed by Korzhyk, Conitzer and Parr [24] tries to solve this defender's dilemma by introducing a Nature node which sets a probability to determine whether the follower is able to observe the leader's strategy or not. Nevertheless, this algorithm actually changes the leader's committed strategy that should have again a chance to be observed by the follower.

In the next section we provide a formal definition of Security Games with a summary of the notation used following in this work. All the problems and algorithm described rely on the assumption of rational adversaries with the ability of fully analyzing the defender's strategy. Finally, in the last section,

we extend the problem to games with two followers where the followers are not able to achieve full coordination.

Chapter 3

Security Games

The objective of this section is to provide the definition of mathematical concepts and notation used hereafter in this work. After the general description of Stackelberg problems we focus our attention on compact form representation which allows the application of efficient algorithms, described in the following section, and then on Bayesian extension, which allows to deal with problems in which the adversary type is unknown a priori. Finally we address the problem of computational complexity.

3.1 Problem description and notation

Security games are a special case of pursuit-evasion games where a set of agents Θ tries to defend a set of targets T from the attacks of a set of agents Ψ. In literature, most of the problem formulations, assume $|\Theta| = 1$ where the singleton agent $d \in \Theta$ is generally called $defender$ or $leader$, referring to the solution concept adopted. This assumption, besides fitting most of the real-world security problems' needs, makes the problem easier to solve, allowing the formulation of more efficient algorithms that otherwise, in the case of multiple competitive defenders, could not be so easily applied. Furthermore, we note that a single defender agent does not always identify a single individual in the real-world, but could also refer to a team of heterogeneous individual with the same interests, coordinated by a single head, for example police force or a security agency or even a set of security agencies, even thought animated by a cooperation spirit rather than a competitive one. In this section we also assume $|\Psi| = 1$, calling the agent $a \in \Psi$ $attacker$ or $follower$, in order to describe the general game formulation that underlies all the literature algorithms later analyzed in this work. The complete notation used in this work is shown in Table 3.1.

Symbol	Definition
d	Leader, Defender
a	Follower, Attacker
T	Set of targets
D	Set of defender's strategies
A	Set of attacker's strategies
S	Support: set of pure strategies played with positive probability
Λ	Set of attacker types
$\mathcal{P}_\lambda$	Probability of facing an attacker of type lambda
δ	Array representing a defender's mixed strategy
ρ	Array representing a attacker's mixed strategy
δ_i	Probability assigned to the pure strategy i when the defender plays the mixed strategy δ
ρ_j	Probability assigned to the pure strategy j when the attacker plays the mixed strategy ρ
$U_d(i,j)$	Utility that defender gets when he plays i and attacker plays j
$U_a(i,j)$	Utility that attacker gets when he plays σ_Θ and defender plays σ_Ψ
$U_d^\lambda(i,j^\lambda)$	Utility that defender gets when he plays i and attacker of type λ plays j
$U_a^\lambda(i,j^\lambda)$	Utility that attacker of type λ gets when he plays i and defender plays j
$\mathcal{V}_d$	Expected utility of the defender when at least one player plays mixed strategies
$\mathcal{V}_a$	Expected utility of the attacker when at least one player plays mixed strategies
$\mathcal{S}$	Stackelberg equilibrium (not to be confused with support)
$\mathcal{N}$	Nash equilibrium

Table 3.1: Table of symbols.

By abstracting the nature of real-world problems, the game can be generally formulated as a normal form game by identifying D as the set of the defender's pure strategies and A as the set of the attacker's pure strategies. The outcomes of the game are represented by a bimatrix, where the single outcome only depends on the strategy adopted by the players. Here we define $U_d(i,j)$ as the payoff obtained by the defender when playing the pure strategy $i \in D$ while the follower plays $j \in A$ and we define $U_a(i,j)$ as the payoff of the attacker in the same status.

$$U_d, U_a : \quad D \times A \longrightarrow \mathbb{R} \tag{3.1}$$

A defender's pure strategy generally refers to an allocation of resources over the set of targets, or sometimes schedules, while each pure strategy of the attacker corresponds to a single target to attack, which means $T = A$. Both attacker and defender are allowed to play mixed strategies, this means to assign a probability distribution over the set of pure strategies. Here we call δ a mixed strategy for the defender and ρ a mixed strategy for the attacker. When playing randomized strategies the outcome of the game is no longer deterministic, then it has to be expressed as an expected value, which can be calculated as the sum of the bimatrix payoffs weighted with the probability distributions:

$$\mathcal{V}_d(\delta, \rho) = \sum_{i \in D} \sum_{j \in A} \delta_i \rho_j U_d(i,j) \tag{3.2}$$

$$\mathcal{V}_a(\delta, \rho) = \sum_{i \in D} \sum_{j \in A} \delta_i \rho_j U_a(i,j) \tag{3.3}$$

Given a strategy of the attacker, the optimal strategy of the defender is called *best response* and can be defined as a strategy δ^* such that:

$$\mathcal{V}_d(\delta^*, \rho) \geq \mathcal{V}_d(\delta, \rho) \quad \forall \delta \tag{3.4}$$

In the same way the best response of the attacker can be defined as ρ^* such that:

$$\mathcal{V}_a(\delta, \rho^*) \geq \mathcal{V}_a(\delta, \rho) \quad \forall \rho \tag{3.5}$$

Both defender and attacker aim to maximize their own utility acting selfishly. In Game Theory, the most famous solution concept for simultaneous games is Nash equilibrium, which is a strategy $\mathcal{N} = \{\delta^*, \rho^*\}$ such that both the players play their best response. However, in Security Games the leader can take advantage of committing to a strategy by playing

the Stackelberg equilibrium, which is a refinement of Nash equilibrium applied to Stackelberg games, that is a strategy such that the expected utility of the leader is maximized while the follower plays his best response. In literature there is a distinguish between Strong Stackelberg Equilibrium (SSE) and Weak Stackelberg Equilibrium (WSE) [5]: the first concept assumes that the follower is compliant and, in case of indifference, he will choose the strategy that maximizes the leader's utility, while the second concept assumes that the follower will choose the strategy that minimizes the leader's utility. As explained in Section 2.2, in security problems it is reasonable to assume that the follower will break ties in leader's favor therefore SSE is generally accepted as the standard solution concept for security games.

Definition 2 (Strong Stackelberg Equilibrium)**.** *a* Strong Stackelberg Equilibrium (SSE) *is a strategy* $\mathcal{S} = (\delta^*, \rho^*)$ *such that:*

- *The leader commits to his best response (3.4)*
- *The follower plays his best response (3.5)*
- *The follower breaks ties in leader's favor (3.6)*

$$\mathcal{V}_d(\delta^*, \rho^*) \geq \mathcal{V}_d(\delta^*, \rho) \quad \forall \rho \tag{3.6}$$

Given this definition, a SSE for security games can be calculated by solving Problem 1 that is nonlinear.

Problem 1: Leader-Follower equilibrium NLP

$$\max_{\delta} \quad \mathcal{V}_d(\delta, \rho^*) \tag{3.7}$$

$$\text{s.t.} \quad \mathcal{V}_a(\delta, \rho^*) \geq \mathcal{V}_a(\delta, \rho) \qquad \forall \rho \tag{3.8}$$

$$\sum_{i \in D} \delta_i = 1 \tag{3.9}$$

$$\sum_{j \in A} \rho_j = 1 \tag{3.10}$$

$$\delta_i \geq 0 \qquad \forall i \in D \tag{3.11}$$

$$\rho_j \geq 0 \qquad \forall j \in A \tag{3.12}$$

3.2 Compact form representation

Most of security games grow combinatorially in the number of the defender's resources and the number of targets to be protected. In most of the cases the payoffs of the matrix in normal form only depend on the value of the attacked target and on whether or not the target is protected by some defender's resources. For example, imagine a situation where the defender can allocate two resources to protect a set of four targets, say t_1, t_2, t_3, t_4 and the attacker must choose a single target to attack. When hit, each target has a certain value for the attacker, for example these values could be respectively $\{2, 3, 7, 5\}$, and a certain value for the defender, which is supposed to be always zero in this example. Each target also assumes a different payoff for both the attacker and the defender when the attacker attempts to hit it when it's covered by the defender. In this example they are supposed to be $\{10, 10, 7, 5\}$ for the defender and $\{0, 0, 1, 2\}$ for the attacker. Given these hypothesis, the normal form representation of the payoff matrix would be structured like this:

	$t1$	$t2$	$t3$	$t4$
$t1t2$	0 10	0 10	5 0	7 0
$t1t3$	0 10	3 0	1 7	7 0
$t2t3$	2 0	0 10	1 7	7 0
$t1t4$	0 10	3 0	5 0	2 5
$t2t4$	2 0	0 10	5 0	2 5
$t3t4$	2 0	3 0	1 7	2 5

Figure 3.1: Security game in normal form.

In the first column we have all the possible resources allocations which correspond to the pure strategies of the defender, while the first row is the list of the targets, which corresponds to the set of the attacker's pure strategies. In all the columns there are two pairs of payoffs repeated for all the defender's allocations. The compact representation proposed by [21] allows to exploit the structure of this type of game by defining for each target a table with four payoff: one for the defender when the attacked target is covered, $U_d^c(t)$, and one when it's uncovered, $U_d^u(t)$, one for the attacker when the attacked target

is covered, $U_a^c(t)$, and one when it's uncovered, $U_a^u(t)$. Then the compact representation of the game in the example would be:

t_1	Covered	Uncovered
Attacker	0	2
Defender	10	0

t_2	Covered	Uncovered
Attacker	0	3
Defender	10	0

t_3	Covered	Uncovered
Attacker	1	5
Defender	7	0

t_4	Covered	Uncovered
Attacker	2	7
Defender	5	0

Figure 3.2: Compact form representation of the security game described in figure 3.1.

By using this representation the number of payoffs is linear with the number of the targets, regardless of the number of defender's resources. The strategy of the leader is no longer represented as a probability distribution over all the possible resources allocations, but it's now defined as a *coverage vector* C, where each target has a certain probability c_t of being protected and the sum of all the probabilities is equal to the number of defender's available resources. Note that the compact representation is equivalent to the normal form representation, in fact, the probability of a target of being covered is equal to the sum of the probabilities over all the normal form allocations which cover that target. With reference to the previous example the coverage vector $C = \{c_{t_1}, c_{t_2}, c_{t_3}, c_{t_4}\}$ can be expressed in function of the normal form strategy δ in this way:

$$c_{t_1} = \delta(t_1, t_2) + \delta(t_1, t_3) + \delta(t_1, t_4) \tag{3.13}$$

$$c_{t_2} = \delta(t_1, t_2) + \delta(t_2, t_3) + \delta(t_2, t_4) \tag{3.14}$$

$$c_{t_3} = \delta(t_1, t_3) + \delta(t_2, t_3) + \delta(t_3, t_4) \tag{3.15}$$

$$c_{t_4} = \delta(t_1, t_4) + \delta(t_2, t_4) + \delta(t_3, t_4) \tag{3.16}$$

Similarly, the strategy of the attacker is defined by an *attack vector*:

$$A = \{a_t : \forall t \in T\} \tag{3.17}$$

In order to effectively exploit the compact representation we have to reformulate the expected payoff of the leader as a function of the coverage vector and attack vector:

$$U_d(C, A) = \sum_{t \in T} a_t \cdot (c_t \cdot U_d^c(t) + (1 - c_t) \cdot U_d^u(t)) \tag{3.18}$$

More specifically, the defender's expected payoff when a certain target is attacked is given by:

$$U_d(t, C) = c_t U_d^c(t) + (1 - c_t) U_d^u(t) \tag{3.19}$$

Similarly, the attacker expected payoff will be:

$$U_a(t, C) = c_t U_a^c(t) + (1 - c_t) U_a^u(t) \tag{3.20}$$

3.3 Bayesian extension

When addressing security problems, in most real-world cases, the defender has to protect a system against many different types of threat. For example, in the task of securing passenger flights, the defender may come up against simple hijackers who wants the aircraft to land in some other airport or, in the worst case, against motivated terrorists who want the aircraft to crash over some important location. While in the first case the defender must consider almost solely the economic loss caused by hijacking, in the second case he must consider the risk of losing hundreds of human lives.

In Game Theory, the uncertain about the type of a player is commonly modeled with the concept of *Bayesian games*, since the Harsanyi's formalization [16] in 1967. Unfortunately, computing the optimal mixed strategy to commit to in Bayesian games has been shown to be NP-hard in 2006 by Conitzer and Sandholm [9] even for a two-player Stackelberg game where the leader is restricted to a single type. In their paper they also provided an algorithm to find the optimal solution by solving a number of linear programs (see Section *Multiple-LPs*). However, Paruchuri, Pearce and Kraus [26] in 2007 insisted on the importance for the security domain of the computation of the leader-follower equilibrium in Bayesian games and provided a heuristic algorithm named ASAP (Agent Security with Approximate Policies). Due to the growing of importance of the application of these games to real world security domains, in the last years, many other algorithms have been sequently developed in order to solve Bayesian Stackelberg Games, see Chapter 4.

In a Bayesian Stackelberg Game the follower is allowed to be of different types coming out from a set of follower types Λ. A follower of type λ has a probability $\mathcal{P}_\lambda$ to be chosen from the set. Being $\mathcal{P}_\lambda$ the probability of facing follower λ we can say that

$$\sum_{\lambda \in \Lambda} \mathcal{P}_\lambda = 1 \tag{3.21}$$

The utility of the leader depends on the type of the follower he faces in the game, then it can be represented with a different matrix U_d^λ for each follower of type λ. With this assumption the expected utility of the leader when the follower plays action ρ^λ can be written as:

$$\mathcal{V}_d(\delta, \rho^\lambda) = \sum_{\lambda \in \Lambda} \mathcal{P}_\lambda \mathcal{V}_d^\lambda(\delta, \rho^\lambda) \tag{3.22}$$

where

$$\mathcal{V}_d^\lambda(\delta, \rho^\lambda) = \sum_{i \in D} \sum_{j \in A} \delta_i \rho_j^\lambda U_d^\lambda(i, j^\lambda) \tag{3.23}$$

Also the follower gains a different utility depending on his type, then, for each follower type λ, the follower's utility can be reperesented using a different matrix U_a^λ. Then, for each follower, his utility can be expressed as

$$\mathcal{V}_a^\lambda(\delta, \rho^\lambda) = \delta_i \rho_j^\lambda U_a^\lambda(i, j^\lambda) \tag{3.24}$$

The follower is a rational agent, independenlty of his type, then he would play his best response. Thus, for each follower type, the following constraint must hold:

$$\mathcal{V}_a^\lambda(\delta, \rho^{*\lambda}) \geq \mathcal{V}_a^\lambda(\delta, \rho^\lambda) \tag{3.25}$$

By applying these changes to Problem 1 we finally obtain the following NLP:

Problem 2: Bayesian Leader-Follower equilibrium NLP

$$\max_{\delta} \quad \sum_{\lambda \in \Lambda} \mathcal{P}_\lambda \mathcal{V}_d^\lambda(\delta, j^\lambda) \tag{3.26}$$

$$\text{s.t.} \quad \mathcal{V}_a^\lambda(\delta, \rho^{*\lambda}) \geq \mathcal{V}_a^\lambda(\delta, \rho^\lambda) \qquad \forall \rho \tag{3.27}$$

$$\sum_{i \in D} \delta_i = 1 \tag{3.28}$$

$$\sum_{j \in A} \rho_j = 1 \tag{3.29}$$

$$\sum_{\lambda \in \Lambda} \mathcal{P}_\lambda = 1 \tag{3.30}$$

$$\delta_i \geq 0 \qquad \forall i \in D \tag{3.31}$$

$$\rho_j \geq 0 \qquad \forall j \in A \tag{3.32}$$

$$\mathcal{P}_\lambda \geq 0 \qquad \forall \lambda \in \Lambda \tag{3.33}$$

3.4 Complexity

When addressing a real world security problem it is important to consider how quickly the solution time increases with the number of targets to defend, resources to allocate, number of attackers or in general any other parameter related to the problem domain. In general, when designing a security assistant software, like any other information system, it is important to take into account the grade of scalability of the system. For example, when defending a perimeter, the defender may get new patrollers in order to enhance the security level or the area to protect may increase in size during time. Then, some questions arise: would the algorithm be able to find a solution in a reasonable time even with more possible strategies? How many targets can be defended using this system before the time needed to find the optimal strategy becomes too high?

Nash equilibrium is generally considered a valid solution concept in competitive games, but, as we previously noted, it does not always provide the best possible outcome and often it does not provide a unique solution. Moreover, the problem of finding a Nash equilibrium even in two-player general-sum games has been proven to be $PPAD$-complete [12]. $PPAD$ is a class of total-search problems belonging to NP in which a solution is guaranteed to exist. By Nash existence theorem, in fact, it is known that a Nash equilibrium in mixed strategies always exists [30], while this is not true in general for all NP problems. However, as for any other problem belonging to NP, a polynomial time algorithm to find it is unlikely to exist, unless $P = NP$.

Earlier in this work, we discussed about benefits of Stackelberg equilibrium over Nash equilibrium. Besides those advantages, in the next section we show how Stackelberg equilibrium in two-player games can be calculated in polynomial time by solving a certain number of LPs. Yet, Bayesian Stackelberg problems are *NP-HARD* as well as the problem of solving games with more than two players [9] [27]. Below we show how Bayesian Stackelberg two-player games can be solved more efficiently by reformulating the original problem or by applying some assumptions over the the game structure of real world security problems. Finally we face the multi-follower problem, providing an analysis of the time complexity of calculating the leader-follower equilibrium both in the general case and by applying some restrictions to the problem.

Chapter 4

Algorithms

4.1 Multiple-LPs

Until 2006, game-theoretic researchers, almost ignored games with leadership commitment. Conitzer and Sandholm proposed Multiple-LPs, sometimes called MULTI-LP, as one of the first approaches to optimally solve Bayesian Stackelberg games [9]. In their work, they proposed a linear-time algorithm to calculate SSE in pure strategies, in a generic multiplayer game. However, as we told in the previous section, randomization of the defense strategy is a specific requirement in most of the real world security problems.

The SSE formulation as shown in Problem 1 allows the leader to commit to mixed strategies but it cannot be solved efficiently due to the nonlinear objective. The MULTI-LP approach provides an instrument to defender to calculate the optimal strategy to commit to in polynomial time using linear programming [9]. The same algorithm can be also applied to Bayesian games by applying it to the Harsanyi transformation of the problem. However in this case, the algorithm is not efficient as the size of the problem grows exponentially in the number of follower types.

4.1.1 The MULTI-LP algorithm

In a leader-follower game, the follower observes the strategy committed by the defender and chooses a strategy to maximize his own utility by reasoning on the payoffs of the game. In general, both leader and follower are allowed to play in mixed strategies. However, once fixed the leader strategy, the follower's best response can be supposed to be in pure strategies, thanks to follower's compliance (see Section 2.2): if the follower, after leader's commitment, is indifferent between two or more strategies, he will always choose the one that maximizes the leader's payoff.

Even in the case that more than one follower strategy satisfies this requirement, the leader's utility would not be affected from an eventual follower's randomization between them. Under this assumption, the problem of finding the best strategy to commit to can be solved by calculating, for each follower's pure strategy j, a mixed strategy δ for the leader such that:

- j is best response to δ
- δ is the strategy, under the previous constraints, which maximizes the leader's outcome

Then, for each follower's pure strategy j, the problem to solve Problem 3:

Problem 3: Leader-Follower equilibrium LP

$$\max_{\delta} \quad \sum_{i \in D} \delta_i U_d(i,j) \tag{4.1}$$

$$\text{s.t.} \quad \sum_{i \in D} \delta_i U_a(i,j) \geq \delta_i U_a(i,j') \qquad \forall j' \in A \tag{4.2}$$

$$\sum_{i \in A} \delta_i = 1 \tag{4.3}$$

$$\delta_i \geq 0 \qquad \forall i \in D \tag{4.4}$$

4.2 RUGGED

Many security problems take place in some kind of network such as urban road networks, transportation networks or computer networks, making the problem of ensuring security over a network a concern of growing importance in the last few years. In these scenarios the aggressor is supposed to observe the security system before choosing an attack strategy, making ineffective the classical deterministic resource allocation techniques which predictable solutions could be exploited by the attacker.

Game Theory provides attractive techniques to solve these problems by considering the attacker as an intelligent agent which reasons about defender's strategy. These games are classified as *interdiction games*, an extension of ambush games, where the task of the defender is to allocate static resources over the edges of the network, trying to maximize his expected reward, while the attacker has to choose the target with the maximum expected payoff for himself and the path to cross to reach it.

Unfortunately, the huge size of the action space of a real world scenario, for example a urban city road network, makes often impossible the application of game-theoretic algorithms due to their computational complexity. In a urban network security domain, for example, the defender could be the police and has to choose which streets to control in order to protect some important locations, while the attacker has to choose a location to attack and which roads to move through to get there. Hence, the action space of the defender grows exponentially with the number of his resources and the action space of the attacker grows exponentially with the size of the network.

RUGGED (Randomization in Urban Graphs by Generating strategies for Enemy and Defender) is a scalable algorithm proposed by Manish et al. 2011 [19], to provide an optimal solution to network security games. It relies on a double oracle column and row generation approach that allows to find an optimal solution without the need to explore the entire action space. RUGGED models the game as a zero-sum game, assuming that the aggressor's reward is always the opposite of the defender's reward. Although this assumption may be too restrictive in situations where the attacker's and the defender's rewards cannot be considered so strongly correlated, it makes possible to adopt a maximin solution concept, with polynomial complexity, making possible the application of the algorithm to complex real world scenarios. The solution provided by maximin is optimal, in fact, due to von Neumann's minimax theorem (von Neumann 1927), in two-player zero-sum games, maximin and minimax strategies coincide both with Nash equilibrium strategies and with Stackelberg equilibrium strategies [41].

4.2.1 The algorithm

RUGGED algorithm consists of three main components: CoreLP, Defender Oracle (DO), Attacker Oracle (AO). These three functions are executed in sequence until the optimal solution has been found, as shown in Algorithm 1. Let's call D the set of the defender's pure strategies to take into account and call A that of the attacker. Each pure strategy i of the defender corresponds to an allocation of the available resources over the network nodes, while each attacker pure strategy corresponds to a path from an entry point to a target. At the beginning of the procedure D and A are initialized to some random subset of their respective players' action sets, forming a subgame. The CoreLP function calculates the maximin mixed-strategy equilibrium of such subgame. This solution, not optimal in general, is then passed to the defender oracle which searches a better pure strategy for the defender.

The attacker oracle does the same, providing if exists, a best response to the defender's strategy found in the subgame equilibrium. If both defender oracle and attacker oracle are unable to find a best response to the equilibrium found by CoreLP, the algorithm stops and returns the optimal solution. Otherwise, if at least one of the two oracles adds a column or a row to the subgame, a new iteration of the algorithm is performed.

Algorithm 1 Double Oracle

```
1: initialize(A)
2: initialize(D)
3: repeat
4:     (δ, ρ) := CoreLP(D, A)
5:     i* := DO(ρ)
6:     D := D ∪ {i*}
7:     j* := AO(δ)
8:     A := A ∪ {j*}
9: until convergence
10: return (δ, ρ)
```

Proceeding in this way, Algorithm 1 starts by solving a small subgame, whose size is incremented at each iteration, increasing also the quality of the solution found from time to time, avoiding the waste of time of considering all the suboptimal strategies. In the following paragraphs we will see in the details how these three functions work.

4.2.2 CoreLP

CoreLP is the function tasked to find a maximin mixed strategy equilibrium in a small subgame. The subgame is extended at each iteration, adding rows corresponding to pure strategies of D and columns corresponding to pure strategies of A. Each column is associated with an attack path, with a single target associated with it. The attacker's reward, that is always the opposite of the defender's reward, corresponds to the value of the target associated with the path, whereas the defender allocation does not intersect the attacker path and zero when the defender allocation intersects the attacker path, due to his capture. To provide a solution of this subgame, CoreLP solves for Nash equilibrium formulated as a maximin problem, shown below as Problem 4.

Problem 4: Nash zero-sum LP

$$
\begin{aligned}
\max_{\mathcal{V}_d^*, \delta} \quad & \mathcal{V}_d^* & & \\
\text{s.t.} \quad & \mathcal{V}_d^* \leq \mathcal{V}_d(\delta, j) & \forall j \in A & \quad (4.5) \\
& \sum_{i \in D} \delta_i = 1 & & \quad (4.6) \\
& \delta_i \in [0, 1] & \forall i \in D & \quad (4.7)
\end{aligned}
$$

This is the typical formulation of a maximin problem, where $\mathcal{V}_d^*$ is the expected utility of the defender in the equilibrium. Under the zero-sum hypothesis it is possible to solve the problem by considering only the defender expected utility. The objective of the algorithm is to find the defender mixed strategy δ whose attacker mixed strategy best response maximizes the defender expected utility. The set of constraints keeps the defender utility under the utility obtained in each pure strategy of the attacker. This is the mathematical description of the attacker's reasoning who tries to maximize his own utility. The expected utility of the attacker can be expressed as the sum of the probabilities of being captured, multiplied for the capture penalty and the probability of a successful attack to the target associated with the path i, multiplied for the target's value.

By setting the capture penalty to zero and defining z_{ij} as the flag indicating whether the attacker path is obstructed by the defender's allocation, given the zero-sum property, the defender expected utility in each attacker pure strategy can be defined as:

$$\mathcal{V}_d(\delta, j) = -\tau_j \cdot \sum_{i \in D} i(1 - z_{ij}) \tag{4.8}$$

where τ_j is defined as the value of the target associated with the path j.

4.2.3 Defender Oracle and Attacker Oracle

Once a solution of the subgame has been found, its optimality has to be evaluated by considering the entire action space. The defender oracle is tasked to evaluate the optimality for the defender's strategy, while the attacker oracle evaluates the optimality for the attacker's strategy. The defender oracle generates all the possible resource allocations, selecting at last, the one with the highest expected payoff for the defender. If this allocation is better than the mixed strategy calculated in the subgame

equilibrium provided by CoreLP, then it's the best response to the attacker's strategy and it's included in D. The attacker oracle does the same for the attacker's side, generating all the possible paths and, for each one, evaluates the attacker's expected utility against the defender's strategy in the equilibrium. If the path with the highest payoff is even better than the payoff obtained in the equilibrium, then the path is included in A.

Algorithm 2 Defender Oracle

```
1: V_d^* := V_d(δ, ρ)
2: i* := null
3: for each i ∈ D \ S(δ) do
4:     if V_d(i, ρ) > V_d^* then
5:         V_d^* := V_d(i, ρ)
6:         i* := i
7:     end if
8: end for
9: return i*
```

definire supporto di rho $\mathcal{S}(\delta) = \{\sigma_\Theta | \delta(\sigma_\Theta) > 0\}$

4.3 DOBSS

In most real world security problems the defender must be prepared to face different types of attacker, from simple robbers up to terrorists. The Bayesian Stackelberg model combines effectively the uncertainty over the follower's type with the leader-follower equilibrium concept, typical of security games. Unfortunately, the problem of finding an optimal mixed strategy for the leader in Bayesian Stackelberg game is NP-Hard even for a two-player game where the leader has only a single type [10]. *Multiple-LPs*, converts the Bayesian game into a perfect information normal-form game using Harsanyi transformation, then solves a large number of linear programs that is exponential with the number of follower types.

In this section we describe DOBSS, the algorithm at the core of the ARMOR system. DOBSS, acronym of Decomposed Optimal Bayesian Stackelberg Solver, is an efficient exact algorithm designed to find an optimal mixed strategy to commit to in Bayesian Stackelberg games. This method presents three main advantages:

- it does not require Harsanyi transformation, thus allowing a more compact representation of the game.

- it solves a unique MIP instead of an exponential number of LPs.
- it exploits the advantage of being leader by solving for a Stackelberg equilibrium rather than a Nash equilibrium.

The Harsanyi transformation converts the Bayesian game into a perfect information game where the number of the action of the defender is the same of the Bayesian version of the game but the set of the actions of the attacker is the cross product of the sets of actions of all the follower types, causing the problem to explode in size. DOBSS finds the optimal strategy to commit to in Stackelberg Bayesian games by efficiently solving a MIP, without the need of the Harsanyi transformation. The key idea behind DOBSS is that evaluating the leader strategy against a Harsanyi-transformed follower is equivalent to evaluating it using the matrix of each individual follower type. Thanks to this problem decomposition, DOBSS procedure exponentially reduces the problem over the Multiple-LPs approach in the number of adversary types [34].

Here we present how to construct the MIP, as shown by Paruchuri, Pierce and Kraus [34], starting from the more intuitive MIQP (mixed-integer quadratic problem) formulation with a single follower type. Then we extend the problem in a multi-follower configuration and finally we decompose it into a MIPas described in [34].

4.3.1 MIQP

The first problem to solve is the follower best response problem. In fact, before the leader can choose his strategy, he must be aware of the follower preferences, which will be the constraints to the leader problem. In this optimization problem the follower tries to maximize his own expected utility by choosing the strategy ρ that is best response to the fixed leader mixed strategy δ. Defining the follower's expected utility as in Equation 3.3 and fixed a defender mixed strategy δ, the optimization problem of the follower can be formulated as:

$$\begin{aligned} \max_{a} \quad & \mathcal{V}_a(\delta, \rho) \\ \text{s.t.} \quad & \sum_{j \in A} \rho_j = 1 \\ & \rho_j \geq 0 \qquad\qquad \forall j \in A \end{aligned} \tag{4.9}$$

In this formulation the strategy of the follower is allowed to be mixed, however, the support of any follower mixed strategy that is best response to the leader strategy δ is composed by only pure strategies that are best

response to δ too. Therefore the optimal follower strategy is a pure strategy where $\rho_j = 1$ in correspondence with the follower's maximal reward. The follower's expected utility when he chooses j can be expressed as:

$$\mathcal{V}_a(\delta, j) = \sum_{i \in D} \delta_i U_a(i, j) \tag{4.10}$$

By applying LP duality and complementary slackness theorems to 4.9 we obtain the dual problem 4.11.

$$\begin{aligned} \min_{\mathcal{V}_a^*} \quad & \mathcal{V}_a^* \\ \text{s.t.} \quad & \mathcal{V}_a^* \geq \mathcal{V}_a(\delta, j) && \forall j \in A \\ & \rho_j(\mathcal{V}_a^* - \mathcal{V}_a(\delta, j)) = 0 && \forall j \in A \end{aligned} \tag{4.11}$$

The complementary slackness condition implies that $\rho_j > 0$ only where the follower expected reward is maximal, equal to $\mathcal{V}_a^*$. While the follower tries this way to maximize his own utility, the leader searches for a strategy to commit to such that the follower's best response maximizes his expected utility. Calling ρ^* the follower's best response to δ, the leader's expected utility when the follower plays his best response is:

$$\mathcal{V}_d(\delta, \rho^*) = \sum_{j \in A} \rho_j^* U_d(i, j) \tag{4.12}$$

Then, the maximization problem for the leader can be formulated as problem 4.13.

$$\begin{aligned} \max_{\delta} \quad & \sum_{i \in D} \delta_i \mathcal{V}_d(\delta, \rho^*) \\ \text{s.t.} \quad & \sum_{i \in D} \delta_i = 1 \\ & \delta_i \in [0, 1] && \forall i \in D \end{aligned} \tag{4.13}$$

The leader solves this problem to find the strategy δ that maximizes his own utility, assuming that the follower will always play the best response. In order to add the follower's best response constraints to this problem, the two linear programs can be merged to compose the entire MIQP:

Problem 5: Leader-follower MIQP

$$\max_{\delta,\rho,\mathcal{V}_a^*} \quad \mathcal{V}_d(\delta,\rho) \tag{4.14}$$

$$\text{s.t.} \quad \sum_{i\in D}\delta_i = 1 \tag{4.15}$$

$$\sum_{j\in A}\rho_j = 1 \tag{4.16}$$

$$0 \le (\mathcal{V}_a^* - \mathcal{V}_a(\delta,j)) \le M(1-\rho_j) \qquad \forall j \in A \tag{4.17}$$

$$\delta_i \in [0,1] \qquad \forall i \in D \tag{4.18}$$

$$\rho_j \in \{0,1\} \qquad \forall j \in A \tag{4.19}$$

$$\mathcal{V}_a^* \in \mathbb{R} \tag{4.20}$$

Here the objective function is quadratic as the leader's expected utility is defined as:

$$\mathcal{V}_d(\delta,\rho) = \sum_{i\in D}\sum_{j\in A}\delta_i\rho_j U_d(i,j) \tag{4.21}$$

In this problem the follower is allowed to play only pure strategies, because, as mentioned, any best response in mixed strategies is composed by pure strategies that are best response too. This assumption simplifies the complementary slackness condition that here is written, linearized, as constraint (4.17) in combination with the constraint of the dual follower problem. In this statement M is a large number and ρ_j acts as a trigger for the bound: if j is best response, then $\rho_j = 1$ and $\mathcal{V}_a^* = \mathcal{V}_a(\delta,j)$, otherwise the constraint does not exist.

4.3.2 Bayesian extension and DOBSS

The previous problem considered only a single follower type. Now we will show how the problem can be extended to the multi-follower case without using the Harsanyi transformation. In the Bayesian game the follower is allowed to be of many different types $\lambda \in \Lambda$, each one associated to a different payoff matrix. The leader and follower payoff matrices associated with each follower of type λ are denoted by U_d^λ and U_a^λ respectively while $\mathcal{P}_\lambda$ indicates the probability to face a follower of type λ and ρ^λ is the mixed strategy of the follower of type λ. The expected utility for the follower of type λ when he plays j is calculated as:

$$\mathcal{V}_a^\lambda(\delta,j) = \sum_{i\in D}\delta_i U_a^\lambda(i,j) \tag{4.22}$$

For each follower type, fixed a leader strategy δ independent of the follower type, we have to solve the following problem:

$$\begin{aligned} &\sum_{j \in A} \rho_j^\lambda = 1 \\ &0 \leq (\mathcal{V}_a^{\lambda *} - \mathcal{V}_a^\lambda(\delta, j^\lambda)) \leq (1 - \rho_j^\lambda) M \quad \forall j \in A \\ &\rho_j^\lambda \in \{0, 1\} \qquad\qquad\qquad\qquad\qquad\quad j \in A \end{aligned} \tag{4.23}$$

The leader's expected utility can be calculated independently for each follower type, too:

$$\mathcal{V}_d^\lambda(\delta, \rho^\lambda) = \sum_{i \in D} \sum_{j \in A} \delta_i \rho_j^\lambda U_d^\lambda(i, j) \tag{4.24}$$

The Bayesian extension to the leader problem simply consists in weighting the leader's expected utility against each follower type with the prior probability of facing a follower of that specific type (Problem 6).

Problem 6: Leader-follower Bayesian MIQP

$$\max_{\delta, \rho, \mathcal{V}_a^*} \quad \sum_{\lambda \in \Lambda} \mathcal{P}^\lambda \mathcal{V}_d^\lambda(\delta, \rho^\lambda) \tag{4.25}$$

$$\text{s.t.} \quad \sum_{i \in D} \delta_i = 1 \tag{4.26}$$

$$\sum_{j \in A} \rho_j^\lambda = 1 \qquad \forall \lambda \in \Lambda \tag{4.27}$$

$$0 \leq (\mathcal{V}_a^{\lambda *} - \mathcal{V}_a^\lambda(\delta, j^\lambda)) \leq (1 - \rho_j^\lambda) M \qquad \forall j \in A, \forall \lambda \in \Lambda \tag{4.28}$$

$$\delta_i \in [0, 1] \qquad \forall i \in D \tag{4.29}$$

$$\rho_j^\lambda \in \{0, 1\} \qquad \forall j \in A, \forall \lambda \in \Lambda \tag{4.30}$$

$$\mathcal{V}_a^{\lambda *} \in \mathbb{R} \qquad \forall \lambda \in \Lambda \tag{4.31}$$

As this problem can be solved independently for each follower type, the Bayesian game is said to be decomposed. The solution of this problem is optimal and it's equivalent to the solution of Problem 7 applied to the Harsanyi-transformed payoff matrix, as demonstrated in [34]. To obtain the DOBSS formulation, the quadratic objective function of this MIQP must be linearized by introducing a new variable

$$z_{ij}^\lambda = \delta_i \rho_j^\lambda \qquad \forall i \in D, j \in A \tag{4.32}$$

Using this new variable, the leader's expected utility can be expressed as

$$\mathcal{V}_d^\lambda(z^\lambda) = \sum_{i \in D} \sum_{j \in A} z_{ij}^\lambda U_d^\lambda(i, j) \tag{4.33}$$

Then, the decomposed MIP at the core of DOBSS can be finally written as:

Problem 7: DOBSS

$$\max_{\rho,z,\mathcal{V}_a^*} \quad \sum_{\lambda\in\Lambda}\sum_{i\in D}\sum_{j\in A}\mathcal{P}^\lambda z_{ij}^\lambda U_d^\lambda(i,j) \tag{4.34}$$

$$\text{s.t.} \quad \sum_{i\in D}\sum_{j\in A} z_{ij}^\lambda = 1 \qquad \forall\lambda\in\Lambda \tag{4.35}$$

$$\sum_{j\in A} z_{ij}^\lambda \leq 1 \qquad \forall\lambda\in\Lambda \tag{4.36}$$

$$\rho_j^\lambda \leq \sum_{i\in D} z_{ij}^\lambda \leq 1 \qquad \forall j\in A, \forall\lambda\in\Lambda \tag{4.37}$$

$$\sum_{j\in A}\rho_j^\lambda = 1 \qquad \forall\lambda\in\Lambda \tag{4.38}$$

$$0 \leq (\mathcal{V}_a^{\lambda*} - \mathcal{V}_a^\lambda(\delta, j^\lambda)) \leq (1-\rho_j^\lambda)M \qquad \forall j\in A, \forall\lambda\in\Lambda \tag{4.39}$$

$$\sum_{j\in A} z_{ij}^\lambda = \sum_{j\in A} z_{ij}^1 \qquad \forall j\in A, \forall\lambda\in\Lambda \tag{4.40}$$

$$z_{ij}^\lambda \in [0,1] \qquad \forall i\in D, \forall j\in A, \forall\lambda\in\Lambda \tag{4.41}$$

$$\rho_j^\lambda \in \{0,1\} \qquad \forall j\in A, \forall\lambda\in\Lambda \tag{4.42}$$

$$\mathcal{V}_a^{\lambda*} \in \mathbb{R} \qquad \forall\lambda\in\Lambda \tag{4.43}$$

The solution of this linear program, can be shown to be equivalent to that of 4.25 [34].

4.4 HBGS and HBSA

The problem of calculating the leader-follower equilibrium in general sum Bayesian Stackelberg games is known to be NP-hard [27] and the fastest known exact algorithms to solve them, such as DOBSS and Multiple-LPs, fail to scale up beyond the few tens of actions and types [18]. While DOBSS solves the Bayesian game by representing it with a MIP, Multiple-LPs converts the game into a perfect information game, using the Harsanyi transformation, then solves the exponential number of small linear programs separately, to find the leader-follower equilibrium of the entire game. Solving many small linear programs is much simpler than solving a single large MIP.

The key idea behind HBGS (Hierarchical Bayesian solver for General Stackelberg games) is that Multiple-LPs could outperform DOBSS if it just was able to solve only a small number of all the linear programs generated by the Harsanyi transformation, discarding all those that are not essential in order to find the optimal solution. To make this possible, HBGS organizes

the games hierarchically, by using a tree where the root corresponds to the entire game and each node represents a partition of the game where only some follower types are present. The algorithm then applies a branch and bound technique that significantly reduces the number of linear programs to be solved, by removing all the unfeasible strategies and calculating an upper bound to the leader expected utility, for all the feasible follower strategies.

The authors also proposed a modified version of HBGS, named HBSA, which scales up better in the number of pure strategies using a column generation technique. Finally, both the algorithms support a quality bound technique which provides a considerable speed up by renouncing to optimality thus without significant loss in the solution quality.

4.4.1 The tree structures

The Harsanyi transformation converts the Bayesian game into a perfect information normal form game. In this representation the action space of the follower corresponds to the Cartesian product of the action spaces of all the follower types of the Bayesian game. This set can also be represented using a tree where each leaf corresponds to a column of the Harsanyi transformed game. Another tree used to organize the game in order to make possible the application of the branch and bound technique. The root of this tree contains the entire game complete of all the follower types, while the sons of each node contain a set of types that is a partition of the set of types of their father node. Any partitioning strategy produces a valid tree if the following two conditions are respected:

- the set of the follower types of each sibling must be disjoined from those of the other siblings.
- the union of the sets of the types of all the siblings must coincide with the set of the types of their father.

Once the type tree has been constructed, the games are solved by applying the Algorithm 3 to all its nodes, starting from the leaves to the root, exploiting the simplifications extracted from the small games to reduce the size of the bigger ones. The first simplification consists in removing the unfeasible actions in the lower levels of the tree, propagating the effect to the higher levels. The second simplification is performed by calculating an upper bound for each feasible action. This allows to sort the follower actions in descending order of their upper bound utility, making the traversal examination to stop when the upper bound becomes lower than the leader maximum expected utility evaluated so far.

4.4.2 Removing unfeasible actions

If a follower action is found to be always worse than some other action, for any leader mixed strategy, it can be removed from the game, marking it unfeasible, because the follower would never include that action in his strategy. Moreover, as shown in [18], the following proposition holds.

Proposition 2. *An action marked unfeasible in a node of the type tree can be marked unfeasible also in all the nodes at higher hierarchical levels.*

The result of removing unfeasible actions in a low level node propagates naturally up in the tree, amplifying the effect on each level, proportionally to the number of the pure strategies of the follower. In general, in a fully binary branched tree with depth ω, the removal of a single strategy for a single follower type causes to remove $|A|^{log_2|\Lambda|}$ pure strategies in the root node, because $\omega = log_2|\Lambda|$.

4.4.3 Bounds

The second part of the technique adopted by HBGS to reduce the action space consists of calculating the leader utility upper bound in the feasible pure strategies of the follower. The follower actions with an upper bound lower than the leader's maximum expected utility don't need to be evaluated. In fact, the leader would prefer to commit to a different mixed strategy such that the follower would avoid such bounded actions. In a game restricted to a single follower type, where $\Lambda = \{\lambda\}$, the maximum reward that the leader could obtain when the follower of type λ plays j^λ, subject to the follower's best response constraint is

$$\mathcal{V}_d^\lambda(\delta, j^\lambda) \leq \sum_{i \in D} \delta_i^* U_d^\lambda(i, j^\lambda) \quad \forall \delta$$

where δ^* is the defender mixed strategy that maximizes the problem. In other words an upper bound can be defined on the defender's utility for each action j^λ:

$$B_j^\lambda = \sum_{i \in D} \delta_i^* U_d^\lambda(i, j^\lambda)$$

In a subgame with $|\Lambda| = \{\lambda_1, \lambda_2, ...\lambda_n\}$, the defender expected utility when the follower plays $j^\Lambda = \{j^{\lambda_1}, j^{\lambda_2}, ..., j^{\lambda_n}\}$ is

$$\mathcal{V}_d(\delta, j^\Lambda) = \sum_{\lambda \in \Lambda} \mathcal{P}^\lambda \mathcal{V}_d^\lambda(\delta, j^\lambda)$$

As a bound has been defined for each $\mathcal{V}_d(\delta, j^\Lambda)$ the bound for the leader reward when the follower plays a pure strategy j^Λ can be expressed as

$$B_j^\Lambda = \sum_{\lambda \in \Lambda} \mathcal{P}^\lambda B_j^\lambda$$

Algorithm 3 HBGS(Λ, D, A^Λ,B^Λ,U_d,U_a)

```
1: (A^Λ , B^Λ) = initialization(Λ)
2: FT := constructFollowerActionTree(A^Λ)
3: A* := leaves-of(FT)
4: B*_j := getBounds(j,B^Λ) ∀j ∈ A*
5: sort(A*,B*_j) // sort j in descending order of B*_j
6: j := [A^1_1,A^2_1,...,A^|Λ|_1] // left-most leaf
7: r* := -inf // current known best solution
8: repeat
9:    (feasible,δ,r) := solve(D,j)
10:   if feasible then
11:      if r > r* then
12:         r* := r
13:         δ* := δ
14:      end if
15:      B*_j := r
16:   else
17:      A* := A* - {j}
18:   end if
19:   j := getNextStrategy(j,r*,A^Λ,B^Λ)
20: until j == NULL
21: return (δ*,r*,A*,B*)
```

4.4.4 HBGS algorithm

HBGS algorithm (Algorithm 3) is executed in the root node. The `initialization` function (Algorithm 4) constructs the type tree recursively, by splitting the set of type of the node according to the partition technique indicated in the `partition` function. The recursion of HBGS is executed in each partition. Finally the results of each partition A^Λ and B^Λ are joined together and returned to HBGS.

The follower pure strategies are then sorted in descending order of B_j^*, for each follower type. For all the feasible actions j, the algorithm calculates

Algorithm 4 initialization(Λ)

1: $A^{\Lambda} = \emptyset$
2: $B^{\Lambda} = \emptyset$
3: **if** $|\Lambda| > 1$ **then**
4: $\quad \{\Lambda_1, ...\Lambda_{|\Lambda|}\} = partition(\Lambda)$
5: $\quad$ **for** $l = 0 \rightarrow |\Lambda|$ **do**
6: $\quad\quad (\delta_l, r_l, A^{\Lambda_l}, B^{\Lambda_l}) = \text{HBGS}(\Lambda_l)$
7: $\quad\quad A^{\Lambda} = A^{\Lambda} \cup A^{\Lambda_l}$
8: $\quad\quad B^{\Lambda} = B^{\Lambda} \cup B^{\Lambda_l}$
9: $\quad$ **end for**
10: **else**
11: $\quad A^{\Lambda} = actionsOfType(\Lambda)$ // actions of the single follower type
12: **end if**
13: **return** $(A^{\Lambda}, B^{\Lambda})$

the defender mixed strategy δ that maximizes his own payoff, under the constraint that j remains the best response for the follower, by solving Problem 3. If it does not exist any δ such that j is the follower best response, then the action is marked as unfeasible and is subsequently removed from the set of feasible actions A^*. Otherwise the action's bound is updated and, if greater than the leader maximum payoff, it becomes the new maximum expected reward and δ is set as the leader optimal strategy.

The function `getNextStrategy` (Algorithm 5) returns the next strategy to be evaluated. As mentioned, the actions of each follower type are ordered in descending order of their bound for each subtree. The function explores the leaves of the follower action tree from left to right so that when the bound of the next action is lower than r^* the algorithm stops to examine the siblings of that action and jumps to the left-most leaf of the upper subtree. If does not exist any action with a bound greater than the value of the best known solution, the function returns *null* and the HBGS algorithm stops, returning the solution found.

4.4.5 HBSA

Many real world security applications have an exponential sized hard constrained defender action space. One example of such application is the FAMS domain, which is later discussed in the section N, where the defender has the task of protecting flights by planning feasible patrols for onboard air marshals. IRIS, the system adopted by FAMS for this purpose, is based on ERASER-C, which is one of the more recent algorithm conceived to solve

Algorithm 5 `getNextStrategy`(j,r^*,A^Λ,B^Λ)

for $l = |\Lambda|$ to 1 Step -1 **do**
  $j := indexOf(A^{\lambda_l}, j^{\lambda_l})$
  // Fix the pure strategies of parents: j^{λ_k}, $k < l$
  // Update the pure strategy of type λ_l: $A^{\lambda_l}(j+1)$
  // Children choose their best pure strategy: $A^{\lambda_k}(1)$, $k > l$
  $j := [j^{\lambda_1}, ...j^{\lambda_{l-1}}, A^{\lambda_l}(j+1), A^{\lambda_{l+1}}(1), ...A^{|\Lambda|}(1)]$
  if $r^* < getBounds(j, B^\Lambda)$ **then**
    return j
  end if
end for
return null

large and complex Stackelberg games. Unfortunately, although ERASER-C works well in international routes, where schedules have typically a single departure and a single return flight, it does not generally provide the optimal solution in longer and more complex paths, which are typical of domestic flights. Moreover, recent results showed that solving Stackelberg games with general scheduling constraints is NP-hard in general and only in rare cases it can be solved in polynomial time.

To overcome these limitations new algorithms have been designed recently, such as ASPEN that uses a column generation technique applied to SPARS problems [20]. In the SPARS problem the action space of the defender is composed by all the feasible joint schedules, which correspond to the sets of schedules that satisfy all the problem's constraints. The set of feasible joint schedules is most of the times so large that it's not even possible to store it on a conventional hard disk. As an example taken from FAMS domain, scheduling only 10 air marshals over 100 flights generates more than $1.7 \cdot 10^{13}$ joint schedules. ASPEN is a branch and price method that does not require to load in memory the entire defender's action space, it only selects the strategies useful to find the optimal solution by solving a master problem and a slave problem alternatively. The master problem calculates the Strong Stackelberg Equilibrium in the restricted game built so far, by searching the defender strategy x that maximizes the defender reward while keeping the attacker play the pure best response. Then the slave problem is applied to the solved to generate the best column to add to the game. The best column corresponds to the joint schedule which most improve the defender's expected utility and it is determined using the concept of reduced cost. In fact, as shown by Bertsimas and Tsitsiklis in 1994 [4], the

column with the minimum reduced cost improves the objective function the most. Calculating reduced cost for all the possible joint schedule would be an inefficient method to determine the best column. For this reason ASPEN formulates a MCNF (minimum cost network flow problem) to efficiently find the best column. HBSA includes a Bayesian version of ASPEN able to solve hard constrained games with multiple follower types that replaces the *solve* function of HBGS, maintaining the rest of the branch and bound algorithm unchanged. This column generation extension makes HBSA to scale well in the number of the targets as well as the number of follower types.

4.4.6 Quality bounds

As mentioned, HBGS, as well as HBSA, calculates the upper bound of the defender's reward for each follower pure strategy. The maximum of these bounds represents an absolute theoretical bound on the value of the optimal solution. Moreover, at each step, the algorithm knows the best solution found so far, which value represents a guaranteed lower bound on the defender's reward. Then, when the difference between these two bounds become enough low, the algorithm stops and returns the current solution, without any significant quality loss. The tests shown that the speed-up obtained by using this approximation technique can reduce computational time of an order of magnitude with only 1% of loss in solution quality [18].

4.5 ERASER

ERASER (Efficient Randomized Allocation of Security Resources) is an algorithm designed by Kiekintveld et al. [21] to optimally solve security games in compact form by solving Problem 8. The solution of this MIP is the optimal coverage vector C that maximizes the defender's expected utility U_d^* and corresponds to a SSE [21]. The first two constraints allow the defender to protect multiple targets with a certain probability c_t by assigning all the available m resources, which are considered all of the same type. The second two constraints force the attacker to play in pure strategies, by choosing a single target to attack with probability 1. For the defender side, the optimality of the assignment is obtained by satisfying constraint 4.49. Last constraint (4.50) forces the attacker to play only the best response to the defender strategy.

Problem 8: ERASER MIP

$$\max_{\delta} \quad U_d^* \tag{4.44}$$

$$s.t. \quad c_t \in [0,1] \qquad \forall t \in T \tag{4.45}$$

$$\sum_{t \in T} c_t \leq m \tag{4.46}$$

$$a_t \in \{0,1\} \qquad \forall t \in T \tag{4.47}$$

$$\sum_{t \in T} a_t = 1 \tag{4.48}$$

$$U_d^* - U_d(t,C) \leq (1 - a_t) \cdot M \qquad \forall t \in T \tag{4.49}$$

$$0 \leq U_a^* - U_a(t,C) \leq (1 - a_t) \cdot M \qquad \forall t \in T \tag{4.50}$$

4.6 ORIGAMI

In almost all security games the defender gets a greater payoff when the attacker tries to hit a protected target and the attacker prefers to attack unprotected targets:

$$\begin{aligned} U_d^u(t) &< U_d^c(t) \\ U_a^u(t) &> U_a^c(t) \end{aligned} \tag{4.51}$$

ORIGAMI algorithm (Optimizing Resources In Games using Maximal Indifference) [21] exploits this assumption to efficiently solve security games in compact form, outperforming all the other known algorithms working under the same hypothesis. The idea behind ORIGAMI comes from three observations:

- Increasing c_t for any target not in the attack set doesn't affect the expected payoffs of the players. In fact it only decreases $U_a(t,C)$ leaving all the other expected payoffs unchanged.
- Adding an additional target to the attack set cannot hurt the defender due to the SSE assumption.
- Call x the attacker's expected payoff in the SSE, then $c_t \geq \frac{x - U_a^u(t)}{U_a^x(t) - U_a^u(t)}$ for every target with $U_a^u(t) > x$. This inequality keeps the attacker indifferent between all the targets in the attack set.

The algorithm starts with an attack set composed by only the target with the greatest $U_a^u(t)$ and goes on by adding targets to the attacking and

redistributing defender's resources in order to maintain the indifference in the attack set. The algorithm stops when the additional coverage needed to maintain the indifference is greater than the amount of resources still available to the defender or when one target is covered with probability 1. In the first case the defender can distribute remaining probability to the targets in the attack set to maintain indifference but no more targets can be added to the attack set. In the second case the attack set is expanded as much as possible so that for all the targets in the attack set have the same utility as the $U_a^c(t)$ of the target covered with probability 1. The ORIGAMI pseudocode is shown in Algorithm 6.

Algorithm 6 ORIGAMI

$targets := T$ sorted by $U_a^u(t)$
$payoff[t] := U_a^u(t)$
$coverage[t] := 0$
$left := m$
$next := 2$
$covBound := -\infty$
while $next \leq n$ **do**
  $addedCov[t] := \frac{payoff[next]-U_a^u(t)}{U_a^c(t)-U_a^u(t)} - coverage[t]$
  if $coverage[t] + addedCov[t] \geq 1$ **then**
    $covBound := max(covBound, U_a^c(t))$
  end if
  if $covBound > -\infty$ OR $\sum_{t \in T} addedCov[t] \leq left$ **then**
    BREAK
  end if
  $coverage[t] += addedCov[t]$
  $left -= \sum_{t \in T} addedCov[t]$
  $next{+}{+}$
end while
$ratio[t] := \frac{1}{U_a^u(t)-U_a^c(t)}$
$coverage[t] += \frac{ratio[t] \cdot left}{\sum_{t \in T} ratio[t]}$
if $coverage[t] \geq 1$ **then**
  $covBound := max(covBound, U_a^c(t))$
end if
if $covBound > -\infty$ **then**
  $coverage[t] := \frac{covBound-U_a^u(t)}{U_a^c(t)-U_a^u(t)}$
end if
return $coverage$

4.7 Porter-Nudelman-Shoham's algorithm

Up to now we described the most important algorithms relying on Stackelberg equilibrium solution concept. As we pointed out in Section 2.2.1, Stackelberg equilibrium is not always applicable. In such cases, by its definition, Nash equilibrium is still the reliable solution concept, neverthless the difficulty of calculating it.

A simple algorithm, valid in many real world problems, has been proposed by Porter, Nudelman and Shoham in 2004[37]. At the basis of the PNS algorithm is the idea that, although calculating Nash equilibrium in a two-player game is PPAD-complete[1], the existence of a Nash equilibrium in a given support can be verified by solving a feasibility LP. In fact, given a support for each player, $(\mathcal{S}_a, \mathcal{S}_b)$, the conditions to check are simple: given the adversary's strategy, all strategies in the support of a player must give an utility equal to the best response utility (4.52) and all strategies which provide a lower utility must stay out of the support (4.53). The feasibility problem is shown as Problem 9.

Problem 9: Nash equilibrium Feasibility LP

$$v^a = \sum_{k \in A_b} \rho_k^b U_a(j,k) \qquad \forall j \in \mathcal{S}_a,\ \forall a, b \in \Psi, b \neq a \qquad (4.52)$$

$$v^a \geq \sum_{k \in A_b} \rho_k^b U_a(j,k) \qquad \forall j \notin \mathcal{S}_a,\ \forall a, b \in \Psi, b \neq a \qquad (4.53)$$

$$\sum_{j \in A_a} \rho_j^a = 1 \qquad \forall a \in \Psi \qquad (4.54)$$

$$\rho_j^a \geq 0 \qquad \forall j \in \mathcal{S}_a,\ \forall a \in \Psi \qquad (4.55)$$

$$\rho_j^a = 0 \qquad \forall j \notin \mathcal{S}_a,\ \forall a \in \Psi \qquad (4.56)$$

The algorithm (Algorithm 7) is based on a support enumeration technique, where all possible combinations of variables are generated, sorted by ascending support size and solved by applying Problem 9. Although the support size is approximately 4^n, in games where $|A_a| = |A_b| = n$ [31], meaning that PNS has exponential worst-case complexity, an equilibrium with a small sized support often exists, then, by solving games starting from small supports generally allows to greatly reduce the number of LPs to be solved.

[1] this information was not known at the time that the algorithm has been proposed but it was well known that computing Nash equilibrium was an hard problem.

Algorithm 7 Porter-Nudelman-Shoham's algorithm (PNS)

supportSizes = list of support sizes $s = (s_a, s_b)$ sorted by, first, $|s_a - s_b|$ and then $(s_1 + s_2)$
for each $s \in$ supportSizes **do**
  for each $S_a \in A_a : |S_a| = s_a$ **do**
    $A'_b = \{k \in A_b : k$ not cond. dominated, given $S_a\}$
    if $\nexists j \in S_a$ cond. dominated, given A'_b **then**
      for each $S_b \in A'_b : |S_b| = s_b$ **do**
        if $\nexists i \in S_a$ cond. dominated, given S_b **then**
          if Problem 9 is satisfable for $S = (S_a, S_b)$ **then**
            return the Nash equilibrium
          end if
        end if
      end for
    end if
  end for
end for

Another technique to reduce the number of problems to be solved is to remove *conditionally strictly dominated actions* from the search space.

Definition 3 (Conditionally strictly dominated action)**.** *An action i of player a is* conditionally strictly dominated *if, given a set of adversary's actions $R_b \subseteq A_b$, $\exists i' \in A_a \; \forall j \in R_b : U_a(i, j) < U_a(i', j)$*

By starting from small support sizes, the benefit of pruning conditionally dominated strategies is amplified because with small supports the conditions under which a strategy is conditionally dominated can be easily satisfied[37]. The algorithm is complete and in the worst case it returns a Nash equilibrium after exploring the whole space of supports whose size grows exponentially with the number of actions.

4.8 Nash equilibrium MIP

Many algorithms have been proposed to calculate Nash equilibria in normal form games, such as Lemke-Howson, Porter-Nudelman-Shoham, MIP Nash and none of them appear to be better than others in all situations[38]. However, when maximizing an objective, the MIP Nash formulation proposed by Sandholm and Conitzer outperforms other algorithms becoming the best algorithm for our purpose [38].

Problem 10: Nash equilibrium MIP

$$\sum_{j \in A_a} \rho_j^a = 1 \qquad \forall a \in \Psi \tag{4.57}$$

$$u_j^a = \sum_{k \in A_b} \rho_k^b U_a(j,k) \qquad \forall j \in A_a,\ \forall a, b \in \Psi, b \neq a \tag{4.58}$$

$$v_a \geq u_j^a \qquad \forall j \in A_a, \forall a \in \Psi \tag{4.59}$$

$$r_j^a = v_a - u_j^a \qquad \forall j \in A_a, \forall a \in \Psi \tag{4.60}$$

$$\rho_j^a \leq 1 - s_j^a \qquad \forall j \in A_a, \forall a \in \Psi \tag{4.61}$$

$$r_j^a \leq M s_j^a \qquad \forall j \in A_a, \forall a \in \Psi \tag{4.62}$$

$$s_j^a \in \{0,1\} \qquad \forall j \in A_a,\ \forall a \in \Psi \tag{4.63}$$

$$\rho_j^a \geq 0 \qquad \forall j \in A_a,\ \forall a \in \Psi \tag{4.64}$$

The MIP Nash, Problem 10, is presented as a feasibility problem, where the only feasible solutions are the Nash equilibria. The followers' pure strategies are included into the mixed strategy using the boolean vector s_j^a, which activates or deactivates the strategy j for each follower a. A pure strategy j is included into the mixed Nash equilibrium, namely $b_j^a = 0$, if no regret r_j^a is associated with that action. A regret r_j^a is defined as the difference between the utility v_a obtained playing the best response and the utility obtained playing strategy j. We note that a regret greater than zero would induce the player to deviate from that strategy. With this formulation, the absence of an objective allows us to introduce a custom objective, such as the maximization of a player's utility, or the leader's utility as we will show in Section 6.3.2.

4.9 Nash LCP and Lemke-Howson algorithm

The problem of calculating a Nash equilibrium in a two-player normal form game can also be formulated as a Linear Complementary Problem (LCP)[39]. This formulation (Problem 11) relies on the definition of mutual best response. By defining v_a as the expected utility of the player a when playing his best response to other player's strategy, it follows that each pure strategy j of a is allowed to be in the support of his best response only if:

$$\sum_{k \in A_b} \rho_k^b U_a(j,k) = v_a \tag{4.65}$$

Conversely, a strategy j cannot be included in the support of the best response of player a if

$$\sum_{k \in A_b} \rho_k^b U_a(j,k) < v_a \tag{4.66}$$

This properties can be obtained by introducing a vector of slack variables r_j^a for each player a. Slack variables are defined greater than zero and equal to the gap between the best response's expected utility and the expected utility of playing j, which is following reported as constraint 4.67. When a slack variable is greater than zero, the corresponding pure strategy is not optimal thus must be excluded from the equilibrium support. This can be obtained with complementarity constraint 4.71, which states that two variables ρ_j^a and r_j^a cannot be different from zero at the same time. Given this property, each couple ρ_j^a and r_j^a such that 4.71 are called complementary variables. This also implies that when a strategy ρ_j^a is played with non-zero probability, the complementary slack variable r_j^a must be equal to zero. In other words a slack variable can measure the *regret* of playing that strategy instead of deviating from it which is the same concept of *regret* also used in MIP formulation discussed in Section 4.8.

Problem 11: Nash equilibrium LCP

$$\sum_{k \in A_b} \rho_k^b U_a(j,k) + r_j^a = v_a \qquad \forall j \in A_a, \forall a, b \in \Psi, b \neq a \tag{4.67}$$

$$\sum_{j \in A_a} \rho_j^a = 1 \qquad \forall a \in \Psi \tag{4.68}$$

$$\rho_j^a \geq 0 \qquad \forall j \in A_a, \forall a \in \Psi \tag{4.69}$$

$$r_j^a \geq 0 \qquad \forall j \in A_a, \forall a \in \Psi \tag{4.70}$$

$$\rho_j^a r_j^a = 0 \qquad \forall j \in A_a, \forall a \in \Psi \tag{4.71}$$

This formulation can be solved using Lemke-Howson algorithm (LH). In order to explain the algorithm details we need to introduce the concept of strategy labeling [39]. We define the following set of labels:

$$L = \{L_i \mid i \in \bigcup_{a \in \Psi} A_a\} \tag{4.72}$$

Where Ψ is defined as the set of players $\Psi = \{a\,, b\}$. A label is defined as follows:

Definition 4 (Labeled strategy)**.** *A mixed strategy ρ_a of player a has label L_i, we write $L_i \in L(\rho_a)$, iff one of the following assertions hold:*

- $i = j$ *such that* $j \in A_a$ *and* $j \notin \mathcal{S}_a$
- $i = k$ *such that* $k \in A_b$ *and* k *is a best response by player* b *to* ρ_j^a

If a strategy contains all possible labels we call it *completely labeled*[39] or *complementary*[13].

Definition 5 (Completely labeled strategy)**.** *A strategy* $\rho = (\rho_a, \rho_b)$ *is said* completely labeled *if*

$$L(\rho_a) \cup L(\rho_b) \equiv L \tag{4.73}$$

A completely labeled strategy is a strategy such that, for each pure action of the two players, defines if the action is in the support, thus being best response of other player's strategy, or if the action is not in the support. Then a completely labeled strategy entirely describes a Nash equilibrium defined as solution of Problem 11.

Proposition 3. *A strategy* $\rho = (\rho_a, \rho_b)$ *is a Nash equilibrium iff is a completely labeled strategy.*

The Lemke-Howson algorithm (Algorithm 8) starts from a basis formed by only slack variables $B = \{r_j^a \mid \forall j \in A_a, \forall a \in \Psi\}$, which is a fictitious completely labeled strategy. This strategy is said fictitious because it is not a valid strategy as it violates constraint 4.68. After leaving fictitious basis the algorithm visits a sequence of vertices of the best response polytope (see [39] for an exaustive graphical exposition) until reaching another completely labeled strategy which is a Nash equilibrium. This phase is called *pivoting*. Starting from constraint 4.67 we can write the initial condition of the system as follows, where r_j^a are the variables in the basis:

$$r_j^a = v_a - \sum_{k \in A_b} \rho_k^b U_a(j,k) \qquad \forall j \in A_a, \forall a, b \in \Psi, b \neq a \tag{4.74}$$

At first step, the variable to enter the basis is chosen randomly from $A_a \cup A_b$, while the leaving variable the basis during pivoting steps is always the complementary of the entering variable. Then, in the next steps, the variable to leave the basis is determined by calculating the *minimum ratio test*. In general, the system of equations 4.76 can be rewritten as:

$$\beta_j^a = c_j^a - \sum_{k \in A_b} \eta_k^b q_k^b \qquad \forall j \in A_a, \forall a, b \in \Psi, b \neq a \tag{4.75}$$

where variables β_j^a are the variables in the basis while η_k^b are the non-basis variables. The variables candidated to enter the basis are all the variables η_k^b

Algorithm 8 Lemke-Howson

```
1: basis = dummyBasis()
2: enteringVariable = pickRandomVariable(A_a ∪ A_b)
3: repeat
4:    leavingVariable = minimumRatioTest()
5:    basis = swap(enteringVariable,leavingVariable)
6:    if isCompletelyLabeled(basis) then
7:       convergence = true
8:    else
9:       enteringVariable = complementaryOf(leavingVariable)
10:   end if
11: until convergence
12: return basis
```

which have non-zero coefficient[2] q_k^b in the equation where the leaving variable is explicit. For each candidate variable η_i^b, equation 4.75 can be rewritten as

$$\beta_j^a = c_j^a - q_i^b \eta_i^b - \sum_{k \in A_b \setminus \{i\}} \eta_k^b q_k^b \qquad \forall j \in A_a, \forall a, b \in \Psi, b \neq a \tag{4.76}$$

Over all candidates, the variable to enter the basis is chosen as that where ratio $\frac{c_j^a}{q_i^b}$ is minimum. In nondegenerate games, the result of minimum ratio test is always a single variable, however, in degenerate games the ratio could be minimum for more than one variable. In those cases the solution is to apply a lexicographic perturbation[31] which, functionally talking, simply consists in applying an order to the pure actions of the players. In such way it's always possible to choose the entering variable deterministically, ensuring the algorithm to converge in a finite number of steps. The algorithm terminates when a completely labeled strategy is reached, which can be written as follows:

$$\forall i \in (A_a \cup A_b) \exists a \in \Psi \, , \beta_i^a : \; \beta_i^a = \rho_i^a \vee \beta_i^a = r_i^a \tag{4.77}$$

In general, LH algorithm converges to equilibrium faster than other known algorithm such as PNS or MIP. Moreover, by restarting LH from a Nash equilibrium it is sometimes possible to reach other Nash equilibria. However it is important to note that, in general, not all the Nash equilibria of the game are reachable using LH. This limitation is more important if the

[2]matrices U_a and U_b must have only non-negative values, otherwise a preprocessing over the matrix is needed before applying Lemke-Howson.

aim is to find a Nash equilibrium that maximizes some quantity, in which case MIP formulation is more convenient.

4.9.1 Randomizing over Lemke-Howson paths

In the previous section we introduced the concept of complementary strategy emphasizing its equivalence with Nash equilibrium. A strategy is said *almost complementary*, instead, if the basis contains one and only one couple of complementary variables[13]. Then, LH algorithm, during pivoting phase, starts from the fictitious solution and follows a path of almost complementary strategies until another complementary strategy is reached. The last reached basis is a Nash equilibrium. The idea behind Random Restart Lemke-Howson (rrLH)[13] is the following: considering that restarting LH from another basis may bring LH to find another equilibrium, if the current path is taking too long to converge, which in the worst case means an exponentially long time, it could be useful to restart the algorithm to try a different path. The algorithm is presented below as Algorithm 9.

Algorithm 9 Lemke-Howson with random restarts (rrLH)

```
1: cutoff = cutoff0
2: pathToExplore = pickRandomVariable((A_a ∪ A_b)\ VisitedPaths)
3: repeat
4:    apply pivoting from pathToExplore until cutoff or equilibrium
5:    if not equilibrium then
6:       if VisitedPaths ≡ (A_a ∪ A_b) then
7:          increase(cutoff)
8:       end if
9:       goto step 2
10:   end if
11: until convergence
12: return basis
```

At start, the `cutoff` is initialized to `cutoff0` and a random variable is chosen to enter the basis. Then, the LH algorithm is executed until the number of pivoting operations exceeds `cutoff` or until an equilibrium is found. If `cutoff` is passed and there are still paths to explore, LH is restarted from a random variable not yet explored, if any. When all paths have been explored and no equilibrium has been found, the list of explored paths is cleared and the `cutoff` is increased. Then, the algorithm restarts from one of the previously interrupted paths until the new `cutoff` is passed. The algorithm stops when the first Nash equilibrium is found.

4.10 Leadership with correlated followers

Correlated equilibrium solution concept was described by Aumann in 1974 as a way to coordinate players in non-cooperative games [1]. In correlated games an agent, either external or internal to the game, recommends secretly to each player which pure strategy to play in the next game. For simplicity we suppose this agent being the first player, or the defender in security games and we call him coordinator. The aim of coordinator is to coordinate other players in order to maximize his own utility, taking into account the rational behavior of each other player. Then, other players will follow the given recommendation which is a strategy such that, in absence of knowledge about others' recommendations, gives to the player an expected utility not lower than that he would obtain by playing the Nash equilibrium.

In 2010 von Stengel and Zamir showed the result that the Stackelberg equilibrium strategy gives to the leader an utility not lower than the utility he would get in any correlated equilibrium [44]. Then, Conitzer and Korzhyk showed that in two-player games, when the leader acts as a coordinator, the problem of finding SSE coincides with the problem finding a correlated equilibrium in which the incentive constraints of the leader has been removed and a maximization of the leader's utility has been introduced [8].

A very general multiplayer formulation of the problem proposed by Conitzer et Korzhyk in [8] in the form of a linear program, is shown as Problem 12, rewritten following our notation. First, we define the expected utility of the player d, namely the coordinator, as:

$$\mathcal{V}_d(p) = \sum_{i \in D} \sum_{\bar{j} \in A} p(i, \bar{j}) U_d(i, \bar{j}) \tag{4.78}$$

In a n-player game, the strategy p is a n-dimensional matrix where each element $p(i, \bar{j})$ represents the joint probability that coordinator and all other players $a \in \Psi$ play respectively i and $\bar{j}$, which is defined as the $(n-1)$-dimensional vector of coordinated players' strategies. Being p the matrix of joint probabilities of all possible players' strategies, the sum of its elements must be 1. For each other player a, instead, we define the expected utility of his each single action $j \in A_a$ in a different way. Once received the coordinator's recommendation to play j, each player, knowing the probability matrix p, lacking of knowledge about other players' actual recommendations, can only speculate about them. His expected utility of choosing a generic action $j' \in A_a$, knowing his recommendation j, can be

written using conditional probability as

$$\mathcal{V}_a(p, j', j) = \sum_{i \in D} \sum_{k \in \Psi'} P(i, k|j) U_a(i, j', k) \tag{4.79}$$

Where Ψ' is the set of strategies of coordinated players different from a. From probability theory we know that

$$P(i, k|j) = \frac{p(i, j, k)}{\sum_{i' \in D} \sum_{k' \in \Psi'} p(i', j, k')} \tag{4.80}$$

Then we can rewrite expected utility defined in (4.79) as follows:

$$\mathcal{V}_d(p, j', j) = \sum_{i \in D} \sum_{k \in \Psi'} \frac{p(i, j, k) U_a(i, j', k)}{\sum_{i' \in D} \sum_{k' \in \Psi'} p(i', j, k')} \tag{4.81}$$

In order to make the players following his recommendations, the coordinator must choose p such that the players have no advantage of playing a different strategy. This can be expressed with the following set of constraints:

$$\mathcal{V}_a(p, j', j) \leq \mathcal{V}_a(p, j, j) \quad \forall a \in \Psi, \quad \forall j, j' \in A_a \tag{4.82}$$

We note that denominator of (4.81) is always positive, then it can be removed from the constraint (4.82), making it linear, without altering the solution. Then, the solution of the leadership problem with correlated strategies can be calculated by solving Problem 12.

Problem 12: Correlated equilibrium LP

$$\max_{p} \quad \mathcal{V}_d(p) \tag{4.83}$$

$$\text{s.t.} \quad \mathcal{V}_a(p, j', j) \leq \mathcal{V}_a(p, j, j) \qquad \forall a \in \Psi, \quad \forall j, j' \in A_a \tag{4.84}$$

$$\sum_{i \in D} \sum_{\bar{j} \in \bar{A}} p(i, \bar{j}) = 1 \tag{4.85}$$

$$p(i, \bar{j}) \geq 0 \qquad \forall i \in D, \bar{j} \in \bar{A} \tag{4.86}$$

As we will show in Section 6.3.2, the solution of this problem in games with more than two followers playing correlated strategies, provides to the leader an expected utility not lower than the expected utility obtained in the Stackelberg equilibrium.

Chapter 5

Real world applications

Security in crowded areas as well as areas of economic or political relevance has become a matter of growing importance in the last few years. The limited amount of resources makes impossible to provide a complete coverage of the threatened areas hence an efficient resource allocation policy is needed. Deterministic scheduling of resource allocations would allow a hypothetical attacker to observe the defender strategy and possibly exploit any predictable pattern in it, in order to launch a better planned attack. A randomized scheduling is therefore needed to prevent pattern predictions but a uniform randomization strategy wouldn't take into account neither the different values of each area nor the intelligent behaviour of the attacker. Moreover human brain has proven to be a poor randomizer and the size of the problem makes the task of manually calculating an optimal solution impossible in most cases. For all these reasons many governments and security companies in all the world are interested in developing intelligent software able to provide optimal resource allocation strategies.

This section summarizes some of the most relevant systems recently developed that apply Game Theory, in particular Stackelberg Games, to face security problems in some real world scenarios.

5.1 ARMOR

Los Angeles International Airport (LAX) is the fifth busiest airport in the United States and serves more than 60 millions passengers per year. In this huge number of people may be hidden many types of threats: arms or drug traffickers, active shooters, terrorists, etc. The task of secure LAX space is given to the LAWA police that performs it by means of different systems of protection such as road checkpoints, canine patrols, passengers screening

and so on. The limited amount of resources makes impossible to ensure a full security coverage to all sensible areas inside the airport therefore it is necessary to allocate the available resources in certain areas leaving other areas uncovered for certain periods of time.

ARMOR (Assistant for Randomized Monitoring over Routes) is a game theoretical software assistant agent successfully deployed since August 2007 at LAX in order to schedule road checkpoints and canine patrol routes to provide an optimal unpredictable resource allocation strategy to the LAWA police [35]. ARMOR is able to take into account the different types of the attackers and the different effects of an attack in the various areas by representing the security problem as a two players non-zero-sum Bayesian Stackelberg Game. The defender action space of the game is the set of all possible resource allocations over all the airport areas, while the attacker action space is composed by the set of the areas of the airport. The rewards in the payoff matrix of the defender indicates how much damaging could be an attack in a certain area launched by a certain type of attacker while the payoffs of the attacker matrix contains the rewards obtained by the attacker launching an attack on the various targets.

ARMOR is composed by two different applications: ARMOR-checkpoint that allows to schedule the road checkpoints and ARMOR-canine that aids to plan routes for the canine units patrols. Each application consists of a user interface, a method to build the game matrices, a game solver and an interface that shows the solution found by means of a spreadsheet. The user interface allows the personnel to alter the action space, editing the number of resources available in a particular timeslot, the timeslot duration and the number of the days to schedule over. Moreover the user can add environmental extra information previous stored in a file to give more flexibility to the system that would otherwise ignore the dynamic changes in the real world. Then the system builds the Bayesian Game matrices, using information provided by LAWA to fill the matrices payoffs and solves it by applying DOBSS algorithm, the fastest solver for this kind of games known so far. After the game has been solved it is possible to add some constraints to the solution, like force a checkpoint (or a canine patrol) or forbid a checkpoint (or a canine patrol) in certain time slots, in order to adapt the solution to the LAWA exceptional needs. Because adding constraints could deteriorate significantly the optimality of the solution an alert is shown to the personnel who can still ignore it.

Before its deploying, ARMOR has been tested in various simulations that showed that the value of the solution provided by ARMOR is always greater than the value of a uniform randomized scheduling allowing the police

to achieve the same performance using significantly less resources. After the deployment in August 2007 it has been well received by LAWA police that, before of ARMOR, used deterministic strategies such as alternation of checkpoints in different days instead of an unpredictable randomized strategy as that provided by ARMOR system.

5.2 GUARDS

The United States Transportation Security Administration (TSA) is responsible for protecting the entire American transportation system which, among other things, includes over 400 airports and approximately 48000 employees. The extension of ARMOR on a national scale is an unfeasible task because this system has been designed for a single airport and for specific security activities such as canine patrols and car checkpoints. Applying this system to such a large number of airports and security activities would mean having to make each time substantial changes to the project in order to take into account the specific characteristics of each airport.

GUARDS (Game-theoretic Unpredictable and Randomly Deployed Security) is a scheduling assistant software designed to aid TSA to optimally allocate resources across hundreds of heterogeneous security activities in order to provide airport protection against different type of threats on a national scale [36]. To meet these goals GUARDS introduces new a game-theoretic approach based on Bayesian Stackelberg.Games, named Security Circumvention Games (SGCs). In this new type of game the defender is allowed to protect targets with multiple types of security activities at the same time, while the attacker has the ability to choose the threat in order to circumvent specific security activities.

The system is distributed over all the airports to better fit the specific needs of each individual airport. However, as TSA wants to maintain a common standard of security among all airports, an entirely decentralized approach would not be an effective solution. For this reason GUARDS includes a central core that allows to perform two centralized primary tasks: the first task is a knowledge acquisition process useful to determine standard matrix payoffs and the second task is to provide to domain experts a detailed overview of the security level in all the airports. In this configuration each airport has to ensure the safety of his own space by allocating his own resources on some security activities. The airport space is subdivided into a certain number of areas, such as waiting areas, ticket areas, cargo areas, etc. In each area it is possible to conduct many different types of security activities. In this way each security activity is composed by the activity

type and the area of coverage. In a classical security game the action space for the defender consists of all distributions of resources over the security activities making the problem growing combinatorially with the number of the security activities.

A new technique is introduced with GUARDS allowing to build a more compact matrix representing the game that, otherwise, would soon become unsolvable. The compact representation introduced with GUARDS allows to significantly reduce the defender action space grouping the security activities that provide coverage to the same area and have the same circumvention cost. This also preserves the optimality of the solution, in fact, allocating resources on similar activities, or even distributing a uniform probability over them, has the same effect on the payoffs. As mentioned, the payoffs are determined by a knowledge acquisition process. This process consists of two phases: in the first phase domain experts are asked to provide some information common to all the airports, such as area definitions, security activities definitions, etc. In the second phase the individual airports have to customize this information to create unique game instances, for example by adding or editing areas or security activities. Each airport is also responsible to provide additional information to determine the unique payoffs associated to each area.

In order to easily define standard comparable payoffs in all the airports, GUARDS system includes a module, which, through a series of quantifiable questions is able to determine automatically the reward and the penalties associated to each area. In this way it is also possible to correct biased payoffs by only editing the mathematical formulas centrally, otherwise, all the local matrices should have to be correct manually. This process allows to generate a unique game for each airport. Once the game matrices are ready the personnel has to specify the number of resources available each day and the number of days to schedule over. Then the game is complete and the solution is calculated by the back-end module using DOBSS. The solution of the game corresponds to an optimal resource allocation scheduling that is presented to the user by means of a spreadsheet.

The simulation results show that, even for a small number of security activities, the compact representation requires significantly less computational time than that required using the standard representation and the gap grows exponentially with the number of security activities. The test performed evaluating the quality of the solution confirms that the game-theoretic approach allows to obtain expected rewards far better that those obtained with a uniform randomized resource allocation. GUARDS system

is currently under evaluation by the TSA and, after the tests, could become part of the TSA nationwide airport security program.

5.3 IRIS

In the United States tens of thousands of commercial flights serve hundreds of thousands of passengers everyday, making this air transportation network a primary target for terrorist aims. The Federal Air Marshal Service (FAMS), under the supervision of TSA, is the agency tasked of providing undercover security personnel to ensure protection aboard commercial flights by detecting or simply deterring potential aggressors. As the resources available to FAMS are not enough to protect each flight all the times, FAMS has to choose which flights should be protected and which should not, reasoning about the value that each flight is believed to have from a terrorist viewpoint. Each flight assumes a different value for terrorists depending, for example, on the number of people onboard or on the importance of the areas overflown along its path. Moreover a flight that is usually marked as a low risk flight could become a high risk flight after some special event occurs.

Previous systems such as ARMOR were designed to schedule canine patrols and road checkpoints, where the only constraints were the amount of available resources. In the FAMS domain instead, other constraints arise because the resources are intrinsically moved through the transportation network, meaning not only that during the flight the resources must become unavailable but also that, after the flight, the resources are located in airports different from departures, making the problem much more difficult. For these reasons, the application of Game Theory to this domain presents three main challenges. First, the huge number of flights per day makes the problem unsolvable for DOBSS. The second issue is the high amount of user input required to provide the payoffs for each individual flight. Finally, expressing these hard scheduling constraints by listing all the valid schedules would make the strategy space to explode in size.

IRIS (Intelligent Randomization In Scheduling) is a scheduling assistant software developed for TSA to overcome these challenges, aiding to efficiently schedule Air Marshal patrols to provide a feasible optimal coverage strategy to the American air transportation network [40]. In order to exponentially reduce the action space, IRIS adopts a new game representation, called compact representation. The compact representation removes redundancies in the normal form matrix and can be efficiently solved using ERASER-C algorithm, which has also the capability to represent constraints in a compact way. However, even with the compact representation, the number

of inputs required to describe all the payoffs would be at least in the tens of thousands, four for each flight. In IRIS this problem is solved by introducing an attribute-based preference elicitation system based on the Threat, Vulnerability and Consequence model (TVC). This system applies a mathematical formula to combine a vector of risk data automatically obtained from the flight attributes and calculates the payoffs relating to that flight. With this system, the amount of user inputs required remains constant as the number of flights increases.

The architecture of IRIS is composed by four main components: an input module, a back-end module, a display/output module and a project management module. The input module receives four classes of inputs, required by the back-end module to construct the Stackelberg Game: the resource data, which describes the number of FAMS and the subsets of flights they can cover; the target data which contains all the necessary information to describe a flight, like departure and destination or the flight number; the risk data required by the preference elicitation system to calculate the payoffs of the game; finally some boundary data useful to the user to better understand the system's outputs. These inputs are then passed to the back-end module that processes them with six primary components. The preprocessing engine combines the target data and the resource data to produce a set of valid schedules. These schedules are used in combination with the risk data by the second module to generate the MIP representing the Stackelberg game. The MIP is then solved using ERASER-C which solution is used by the fifth component of the back-end module to produce a randomized schedule. Finally, using this probability distribution, a sample schedule is generated. The solution produced is finally displayed to the user with a user-friendly representation in the display/output module.

The project management tool allows to create a project file that stores the input files, the additional risk data and the various settings for future uses. IRIS has been tested using real flight data taken from different regions, comparing the quality of its solution to the ones provided by a uniform randomized scheduler and a weighted randomized scheduler. The quality of the solution provided by IRIS is always better than other solutions and the difference becomes clearer as the size of the action space increases. The system has been delivered to FAMS and it's currently under evaluation to assess the possibility of integrate it into their scheduling system.

Chapter 6

Multi-follower games

All the problems and algorithms described so far rely on the assumption that, at all the time, there is at most one single attacker who plans to hit the infrastructure protected by the defender. Although this hypothesis holds good in many situations, in some circumstances it could be reasonable to suppose that two or more aggressors could plan to attack independently but simultaneously. For example, in an airport more than one terrorist organization, with different heads and different aims, could plan to attack different areas of the airport concurrently. Another situation where it would be useful, if not necessary, to consider more than one simultaneous attacker are public demonstrations, such as protests against G8 summits, where many groups attending the manifestation, act without coordination in order to achieve their own distinct objectives, some peaceful, some not, while the task of the police is to preserve vulnerable areas of the city as well as ensuring the safety of non-violent protesters. Another, even more interesting real world application, would be in the war against narcotraffic, for example the Mexican Drug War, where the government tries to suffocate criminal organizations, while multiple drug cartels fight each other, pursuing exclusively their own interests.

The aim of the second part of this work is to analyze how the structure and the complexity of the game change when a second attacker joins the game. Finally we propose some algorithms to calculate the optimal strategy for the defender, providing an analysis of their performance.

6.1 Problem analysis

In classical Bayesian security games the payoffs of the defender and the attacker depend on the type of the latter but only a single attacker is

allowed to play at the same time. However, in some games where the defender has to deal with attackers of different types, they could be ruled by different autonomous agents with independent heads. In this case it could be reasonable to suppose that more than one of the attackers of the different types could attack simultaneously. The resulting game is a three players security game. As the nature of the game does not change, the followers are still supposed to observe the leader strategy before planning the attack. So, the leader keeps the ability to commit to a strategy as in the single follower case, while the followers play at the same time with the aim of maximizing their own utility. As the followers are supposed to play simultaneously, selfishly and without coordination the followers subgame can be modeled as a normal form game in which teh followers play according to the Nash equilibrium solution concept.

6.1.1 Payoffs structure

In two players security games some assumptions on the structure of the payoffs could be considered valid in general, for example the zero sum formulation or the compact form simplification, because in a real world scenario, a high payoff for the attacker would likely correspond to a low payoff for the defender and vice versa. When trying to apply a simplification on the payoff structure in a multi-follower game, instead, a question arises naturally: how would the payoffs change when the followers decide to attack the same objective? Basically this strongly depends on the application context. To give a rough idea we only mention the following cases, reasoning on followers payoffs:

- The followers' payoff becomes higher in the case that the effects of the attacks are additive in some way.
- The followers' payoff becomes lower when only the first attack is possible or in the case that the followers hinder each other.
- The followers' payoff does not change.

The same considerations could also affect the leader's payoffs. In fact, in the case of a single attack of two follower types he would get one of the two payoffs, depending on which type of attacker he would deal with. When two attacks happen at the same time on the same target, instead, the leader's payoff would be affected in one of the following ways:

- The leader's payoff becomes lower than the lowest of the two payoffs when the effects of the attacks are additive in some way.

- The leader's payoff becomes higher in the case that the followers hinder each other or in the case that he would capture both the attackers.
- The leader's payoff is the lowest[1] of the payoffs he would get in the case of a single attack, when only a single attack is possible.

However, all these considerations can be evaluated only by security experts who design the payoffs matrices by analyzing the players' utilities in a real world scenario, in all the different situations so this will not be further discussed here. Hereinafter in this work, in order to provide a very general analysis to the problem, we suppose the payoffs to be randomly distributed according to a uniform distribution.

6.1.2 Compliance

In two players security games, Strong Stackelberg Equilibrium solution concept relies on the assumption that the follower will always break ties in leader's favor, because, as demonstrated by Zamir and Von Stengel [43] compliance always provides to him better outcomes or at least equal to those obtained in the Nash equilibrium. In the same article they also showed how this assumption no longer holds, in general, in games with more than one follower. However, compliance, in a slightly different form, can still be considered a valid assumption in multi-follower security games.

Take for example the game shown in Figure 6.1 in which a leader Θ faces two followers $\{a, b\} \in \Psi$. In this game the strategy set of the leader is $A_d = \{l_1, l_2\}$, while the strategy sets of the followers are $A_a = \{\alpha_1, \alpha_2\}$ and $A_b = \{\beta_1, \beta_2\}$. Each mixed strategy of the leader induces the followers to play a different normal form game whose payoffs are calculated as a linear combination of the payoffs of the followers' games induced by the leader's pure strategies, reported in Table 6.1.2.

l_1	β_1	β_2
α_1	0, 9, 6	8, 0, 0
α_2	8, 0, 0	0, 6, 9

l_2	β_1	β_2
α_1	2, 0, 9	1, 8, 8
α_2	1, 7, 7	2, 9, 0

Figure 6.1: An example of multi-follower game.

When the leader commits to l_1, the followers play the classic *battle of sexes* game, while when he commits to l_2 the induced game is a *prisoner dilemma*. As in the single follower case, the aim of the leader is to induce the followers to play the equilibrium that maximizes his own utility. However

[1] worst case scenario

in this case committing to a mixed strategy is not sufficient to force the followers to play a certain equilibrium. When the leader commits to l_1, for example, the followers' game, as well known from theory, presents three Nash equilibria: two stable pure strategies Nash equilibria $\mathcal{N}^1 = \{\rho_a^1, \rho_b^1\}$ where $\rho_a^1 = \{\alpha_1 = 1, \alpha_2 = 0\}$ and $\rho_b^1 = \{\beta_1 = 1, \beta_2 = 0\}$ and $\mathcal{N}^2 = \{\rho_a^2, \rho_b^2\}$ where $\rho_a^2 = \{\alpha_1 = 0, \alpha_2 = 1\}$ and $\rho_b^2 = \{\beta_1 = 0, \beta_2 = 1\}$ and one unstable mixed strategies Nash equilibrium $\mathcal{N}^3 = \{\rho_a^3, \rho_b^3\}$ where $\rho_a^3 = \{\alpha_1 = \frac{3}{5}, \alpha_2 = \frac{2}{5}\}$ and $\rho_b^3 = \{\beta_1 = \frac{2}{5}, \beta_2 = \frac{3}{5}\}$. The expected utilities of the leader in both the pure equilibria is 0, while his expected utility in the mixed equilibrium is 4.16. So, the leader would commit to the strategy l_1 only if the followers play their mixed equilibrium, otherwise he would prefer to commit to l_2 where the unique pure strategies Nash equilibrium would give to him an utility of 1. However, once he commits, the followers are free to play their best strategy.

In this example, the followers a and b get symmetric payoffs, which give them an expected utility of 6 in one of the pure equilibria and 9 in the other and an expected utility of 3.6 in the mixed equilibrium, so if they could coordinate, they would choose to play one of the pure equilibria because the expected utility of the worst one is still better than that obtained in the mixed equilibrium. In this case, as stated by Zamir and Von Stengel the followers would not take advantage of breaking ties in leader's favor and the commitment would not be an advantage for the leader. However in a scenario where the followers cannot coordinate their strategies, such as the multi-follower security game scenario, the leader would be able to induce the followers to play the equilibrium which is best for himself. In fact, when the leader commits to a strategy, he may also suggest a strategy that each follower should play. If any follower knows that all the other followers will play that equilibrium, by definition of Nash equilibrium, no one would take advantage from being the only one who plays a different strategy.

In this context compliance is still a strong assumption on the basis of which we can build the multi-follower problem formulation proposed in the following section. Although in this work we adopt SSE as the more natural solution concept for multi-follower games, in some case we will analyze how the problem changes when solving for WSE or when the followers simply play a Nash equilibrium.

6.2 Mathematical formulation

In this section we propose the most general mathematical formulation of the multi-follower problem. As in all the Stackelberg games, the leader, or defender, here indicated with d, must commit to a strategy that maximizes

his own utility. As discussed in the previous section, we find reasonable the hypothesis that the followers, $a \in \Psi$, will break ties in leader's favor.

Definition 6 (Multifollower Game)**.** *A multifollower game is a structure* $\mathcal{G} = (N, A, U)$ *such that*

- $N = (d, \Psi)$ *is the set of players where* d *represents the leader and* Ψ *is the set of followers.*
- $A = A_d \times A_\Psi$ *is the set of players' actions where* A_d *is the set of leader's actions and* $A_\Psi = A_1 \times A_2 \cdots \times A_{|\Psi|}$ *is the set of followers' actions.*
- $U = (U_d, U_\Psi)$ *where* $U_d : A \to \mathbb{R}$ *and* $U_\Psi : A \to \mathbb{R}^{|\Psi|}$ *represent the utilities of all players in each game's outcome.*

Given these hypothesis and definitions we can formulate the objective function of the maximization problem as follows:

$$\max_{\delta} \sum_{i \in D} \delta_i \mathcal{V}_d(\Psi, \mathcal{U}_d(i)) \tag{6.1}$$

In this formulation we use $\mathcal{U}_d(i)$ to indicate a slice of the n-dimensional matrix indexed by i with $n = |\Psi|$. $\mathcal{V}_d$ is the expected utility obtained by the leader in the Nash equilibrium of the followers' subgame generated by the leader's commitment. To fit the very general case we can express $\mathcal{V}_d$ with the following recursive function:

$$\mathcal{V}_d(\Psi', \mathcal{U'}_d) = \begin{cases} \sum_{j \in A_a} \rho_j^a \mathcal{V}_d(\Psi' \setminus \{a\}, \mathcal{U'}_d(i)) : & a \in \Psi', |\Psi'| > 1 \\ \sum_{j \in A_a} \rho_j^a \mathcal{U'}_d(i) : & |\Psi'| = 1 \end{cases} \tag{6.2}$$

When the set Ψ' contains more than one follower, the function is recalled removing an element from the set. When Ψ' is finally a singleton set, the function actually calculates the expected utility of the leader using the payoffs of the array $\mathcal{U'}_d$. After the leader commits to a strategy, the game becomes a normal form game where the followers play their mutual best responses. Then, said v_a the expected utility of the best response of follower a, by definition, we assert that:

$$v_a \geq \sum_{i \in D} \delta_i \mathcal{V}_a(\Psi \setminus \{a\}, \mathcal{U}_a(i, j)) \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.3}$$

where $\mathcal{U}_d(i, j)$ is a $(n-1)-$dimensional matrix, indicating the slice of the payoff matrix of follower a indexed by the pure strategies i and j.

Analogously to $\mathcal{V}_d$ we define the expected utility of the follower a as:

$$\mathcal{V}_a(\Psi', \mathcal{U}'_a) = \begin{cases} \sum_{j \in A_a} \rho_j^a \mathcal{V}_a(\Psi' \setminus \{a\}, \mathcal{U}'_a(j)) : & a \in \Psi', |\Psi'| > 1 \\ \sum_{j \in A_a} \rho_j^a \mathcal{U}'_a(j) : & |\Psi'| = 1 \end{cases} \tag{6.4}$$

Condition 6.3 assures that v_a is greater than or equal to the greatest expected utility. Finally we can assert that j is in the strategy support of follower a when j is a best response to the other players' strategies:

$$j \in \mathcal{S}_a \implies v_a = \sum_{i \in D} \delta_i \mathcal{V}_a(\Psi \setminus \{a\}, \mathcal{U}_a(i,j)), \qquad \forall j \in A_a, \forall a \in A_a \tag{6.5}$$

This comes from the definition of Nash equilibrium, meaning that only strategies which give an expected utility equal to v_a are included in the support $\mathcal{S}_a$. A pure strategy j is in the support of a mixed strategy ρ_a when $\rho_j^a > 0$. The logical proposition 6.5 can be equivalently rewritten as the following equation:

$$\rho_j^a v_a - \sum_{i \in D} \delta_i \rho_j^a \mathcal{V}_a(\Psi \setminus \{a\}, \mathcal{U}_a(i,j)) = 0 \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.6}$$

This equation is satisfied when the strategy j is not in the support or when v_a corresponds to the best expected utility.

6.3 Security games with two attackers

Now we propose some algorithms to calculate the optimal solution for security games with more than one follower. We analyze the simplest case where a defender faces two attackers. Depending on the application scenario it can be useful distinguish the following four cases:

- All the players can play only pure strategies ($LPFP$)
- The leader can play only pure strategies while the followers are allowed to play mixed strategies ($LPFM$)
- The leader is allowed to play mixed strategies while the followers can only play pure strategies ($LMFP$)
- All the players are allowed to play mixed strategies ($LMFM$)

We propose different algorithms to solve these four problems, providing an analysis of their performances in terms of solution time and quality of the solution found.

6.3.1 LPFP: Leader Pure Followers Pure

The simplest case is a game where all the players are only allowed to play pure strategies. We define LPFP as follows:

Definition 7 (LPFP equilibrium)**.** *The* LPFP equilibrium*, if exists, is a strategy* $\mathcal{S} = (\bar{i}, \bar{j}, \bar{k})$ *such that:*

- $\mathcal{N}(G_{\bar{i}}) = (\bar{j}, \bar{k})$ *is a Nash equilibrium for the followers' subgame induced by leader's pure strategy* $\bar{i}$
- $\forall i \in D, \exists \mathcal{N}(G_i) = (j, k) \implies U_d(\bar{i}, \bar{j}, \bar{k}) \geq U_d(i, j, k)$

The strategy $\mathcal{S}$ can be found by simply enumerating, for each leader strategy, all the pure Nash equilibria of the related follower subgame and choosing the one which grants the best utility for the leader, if at least one equilibrium exists, otherwise the game is declared infeasible. The algorithm proposed examines one by one all the pure strategies of the players checking whether they are Nash equilibria for the followers or not. If the strategy is a Nash equilibrium and the payoff obtained by the leader is better than the best payoff found so far, the solution is temporarily flagged as optimal. The search continues until all the strategies have been examined. The computational complexity of this algorithm is in $\Omega(|A_d| \cdot |A_a| \cdot |A_b|)$ and in $\mathcal{O}(|A_d|^2 \cdot |A_a| \cdot |A_b|)$. If no equilibria are found after every possible follower strategy profile has been checked, the game is declared infeasible.

Existence of a LPFP equilibrium

It's interesting to note that, although the existence of a LPFP equilibrium is not guaranteed in general, the probability of LPFP-infeasibility is as lower as the number of leader's actions grows.

Proposition 4 (Probability of feasibility of LPFP)**.** *Given a multifollower game* $\mathcal{G}$ *with uniform random payoffs, the probability* $P(LPFP)$ *of having a LPFP equilibrium is such that*

$$\lim_{|A_d| \to +\infty} P(LPFP) = 1 \tag{6.7}$$

Proof. The critical element for the existence of a LPFP equilibrium in a multifollower game is the existence of at least a pure strategy Nash equilibrium in at least one of the follower subgames. The probability of

the non-existence of a pure strategy Nash equilibrium in a $m \times n$ two-players game has been discussed in [15] and is showed to be equal to

$$1 - P_{\mathcal{N}} = \sum_{k=0}^{K} (-1)^k \binom{m}{k} \binom{n}{k} \frac{k!}{(mn)^k} \tag{6.8}$$

where $K = \min(m, n)$. If no pure strategy Nash equilibria exist for any leader strategy, then the game is LPFP-infeasible. The probabilities of having a pure strategy Nash equilibrium are independent for each subgame, then, the probability of a game being LPFP-infeasible, is calculated as 6.8 elevated to the cardinality of the set of leader's actions:

$$P(LPFP^c) = (1 - P_{\mathcal{N}})^{|A_d|} \tag{6.9}$$

which clearly tends to zero for increasing $|A_d|$, meaning that $P(LPFP)$ tends to 1. □

In Table 6.1 we reported some probabilities of dealing with a feasible game instance supposing the followers having the same number of actions, which is a reasonable hypothesis for most of security games. The payoff are supposed to be uniformly random distributed. The probabilities $P(LPFP)$ have been calculated starting from formula 6.8.

$\lvert A_a \rvert =$	2	3	4	5	6	7	8
$\lvert A_d \rvert = 2$	0.984	0.954	0.933	0.920	0.911	0.904	0.899
3	0.998	0.990	0.983	0.977	0.973	0.970	0.968
4	1.000	0.998	0.996	0.994	0.992	0.991	0.990
5	1.000	1.000	0.999	0.998	0.998	0.997	0.997
6	1.000	1.000	1.000	0.999	0.999	0.999	0.999
7	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table 6.1: Calculated theoretical probability of having a LPFP-feasible multifollower game depending on the number of actions per player, supposing followers having the same number of actions.

By observing Table 6.1 we note that as the number of defender's actions grows, the probability of having a feasible game quickly tends to 1. It's interesting to note that, instead, when the number of followers' actions is increased, the probability of having a LPFP-feasible instance decreases. However it has a non-zero limit because, as shown in [15], the probability of a Nash equilibrium in pure strategies has a limit when $|A_a|$ tends to infinity, which is:

$$\lim_{|A_a| \to +\infty} P_{\mathcal{N}} = 1 - \frac{1}{e} \tag{6.10}$$

Then, by applying limit 6.10 to 6.7 and fixing the number of leader's pure strategies, we obtain the following limit:

$$\lim_{|A_a|\to+\infty} P(LPFP) = \frac{1}{e^{|A_d|}} \tag{6.11}$$

This property is expecially good in security games, because in many scenarios, for example when the defender randomizes over schedules, the number of leader's pure strategies grows exponentially with the number of targets, whose cardinality is equal to the number of followers' pure strategies. Moreover, as later shown in Figure 6.6 on the right, the value of the LPFP solution in random games with integer utilities ranging from 0 to 100 grows with the number of players' actions. By increasing the number of payoffs and the number of subgames, in fact, the probability of having a good leader payoff in correspondence of a pure strategy Nash equilibrium of the follower subgame tends to 1. However, it's important to note that these properties are valid only under the hypotesis of uniform random games. In games where the followers' interests are in direct conflict, for example, the limits 6.10 and 6.11 tend to zero as the number of players' actions increases [15].

6.3.2 LPFM: Leader Pure Followers Mixed

The second problem we analyze is a scenario where the leader has to commit to a pure strategy while the followers play in mixed strategies. To solve this problem we cannot simply enumerate all the Nash equilibria of the followers' subgames, for each leader's pure strategy, as the problem of calculating a Nash equilibrium in mixed strategies, in a single two players game has proven to be *PPAD-complete*. Moreover in this case we need to calculate the Nash equilibrium which grants to the leader the best possible outcome, which makes the problem *NP-HARD*. Then, a better approach would be to identify and exclude from the leader's set of strategies those that cannot be optimal before calculating any mixed Nash equilibrium in the related followers' subgame.

For this purpose we propose a MULTI-MINLP branch and bound algorithm, whose pseudocode is presented as Algorithm 10. This algorithm allows to find the optimal solution with a considerable speedup by solving only a small subset of all the followers' subgames. We also present a single MINLP formulation and we compare their performance in section 6.5.

MULTI-MINLP and MULTI-LCP algorithm

By decomposing the problem into many subproblems generated by leader's commitment we can explore three different cases depending on followers' behavior:

- followers act to maximize leader's utility (SSE),
- followers act to minimize the leader's utility (WSE),
- followers simply play a Nash equilibrium.

We note that in all of these cases, the leader's commitment is aimed at maximizing his own utility, then the branch and bound algorithm remains identical in all cases with the only difference of the lower bound calculation. At first, the algorithm calculates an upper bound on the leader's expected utility, for each leader's pure strategy i. A first upper bound can be determined by finding the maximum leader payoff in the follower subgame. Stackelberg equilibrium with correlated followers [8] represents a tighter upper bound of each strategy, when maximizing leader's utility, and it can be found by solving a LP in polynomial time. The proof of feasibility of correlated equilibrium as a leader-follower upper bound is provided in the following paragraph. In the case that followers act to minimize the leader's utility, instead, for each leader pure strategy an upper bound is obtained by looking for the pure followers Nash equilibrium with the lowest leader's payoff. A tighter upper bound can be obtained using the minimun between the upper bound just described and a Nash equilibrium calculated using rrLHin the followers subgame. The leader's pure strategies are then sorted with decreasing upper bound. Then, a first sweep is performed calculating the best pure Nash equilibrium, for each i.

The search stops when the upper bound of the next strategy to evaluate is lower than the best solution found so far. This modified version of the LPFP algorithm benefits of the upper bounds already calculated and provides a cheap instrument to prune some of the strategies before starting to search for a mixed strategies equilibrium. The next step consists in calculating the best mixed strategies Nash equilibrium for each i, ordered by descending upper bound, until the next strategy's upper bound becomes lower than the best known lower bound. To find the best mixed Nash equilibrium in a subgame we use the MINLP reported as Algorithm 6.3.2, obtained from the MIP Nash [38] algorithm by adding maximization or minimization of the

Algorithm 10 Stackelberg equilibrium LPFM

1: **for each** $i \in D$ **do**
2: $UB(i)$ = correlated2p($G(i)$) // or minimumPureNash in the minimization case
3: **end for**
4: D = sort(D,UB,*descending*)
5: $LB = -1$
6: **for each** $i \in D$ **and** $UB(i) > LB$ **do**
7: lb = lowerBoundPure($G(i)$)
8: **if** $lb > LB$ **then**
9: $lb = LB$
10: **end if**
11: **end for**
12: **for each** $i \in D$ **and** $UB(i) > LB$ **do**
13: lb = lowerBoundMixed($G(i)$)
14: **if** $lb > LB$ **then**
15: $lb = LB$
16: **end if**
17: **end for**

leader's expected utility. In the case that followers play any Nash equilibrium regardless of the leader's utility, instead, `lowerBoundMixed` can be calculated more quickly using Lemke-Howson algorithm (Lemke-Howson 1964) [25].

Leadership with correlated followers as upper bound

As we said, in games with more than two players, committing to a correlated strategy gives to the leader an expected utility greater than or equal to that obtained when committing to a mixed strategy.

Proposition 5 (Commitment to correlated strategies as upper bound)**.** *In a multi-follower game, the expected utility of the leader when the followers play correlated strategies is never worse than the expected utility in the Stackelberg equilibrium.*

Proof. In order to give a proof that when the followers play correlated strategies the leader's expected utility is never worse than that in the leader-follower equilibrium, in games with more than two players, we will show that exist some games in which, if followers play in correlated strategies, the leader get better outcomes than those obtained in the Stackelberg equilibrium but there are no games where leader-follower equilibrium provides better

outcomes than those obtained when followers play correlated strategies.
Part 1: The objective function with correlated followers possibly allows better outcomes than those obtained in the leader-follower equilibrium, under their respective constraints. Referring to 6.1 and 4.78 we obtain 6.12.

$$\exists \mathcal{G} : \sum_{i \in D} \sum_{\bar{j} \in \bar{A}} p(i, \bar{j}) U_d(i, \bar{j}) \geq \sum_{i \in D} \delta_i \mathcal{V}_d(\Psi, U_d(i)) \tag{6.12}$$

A simple proof of this assertion can be achieved by proving the existence of a single instance where this condition is valid. Take for example the case where the followers' Nash equilibrium, $\mathcal{N}_\Psi$, composing the Stackelberg equilibrium is formed by a mixed strategy for the players a and b while the other followers' strategies are irrelevant. In particular we suppose that the support of the $\mathcal{N}_\Psi$ would include the following pure strategies $\{\alpha_1, \alpha_1\} \in A_a$, $\{\beta_1, \beta_2\} \in A_b$.

$$\{\alpha_1, \alpha_2, \beta_1, \beta_2\} \in \mathcal{N}_\Psi \qquad a, b \in \Psi \tag{6.13}$$

Now call $\hat{\mathcal{V}}_d$ the bidimensional matrix of the leader's payoffs induced by the leader strategy in the Stackelberg equilibrium and by the strategies of the followers different from a and b. Suppose that the payoffs of this matrix are such that:

$$\hat{\mathcal{V}}_d(\alpha_1, \beta_1) < \hat{\mathcal{V}}_d(\alpha_1, \beta_2) \tag{6.14}$$
$$\hat{\mathcal{V}}_d(\alpha_1, \beta_1) < \hat{\mathcal{V}}_d(\alpha_2, \beta_1) \tag{6.15}$$
$$\hat{\mathcal{V}}_d(\alpha_2, \beta_2) < \hat{\mathcal{V}}_d(\alpha_1, \beta_2) \tag{6.16}$$
$$\hat{\mathcal{V}}_d(\alpha_2, \beta_2) < \hat{\mathcal{V}}_d(\alpha_2, \beta_1) \tag{6.17}$$

Although this equilibrium is the best solution for the leader in a Stackelberg game, it could be improved if the leader would have the ability to coordinate the followers to play only $\{\alpha_2, \beta_1\}$ and $\{\alpha_1, \beta_2\}$. A mixed Nash equilibrium does not allow to assign a probability of zero to the combinations $\{\alpha_1, \beta_1\}$ and $\{\alpha_2, \beta_2\}$ while keeping positive probability over $\{\alpha_2, \beta_1\}$ and $\{\alpha_1, \beta_2\}$, in fact, as independent events, the probability of playing each combination coincides with the product of the probabilities over the followers' pure strategies.

$$P(\alpha_1, \beta_1) = \rho^a_{\alpha_1} \cdot \rho^b_{\beta_1} \tag{6.18}$$
$$P(\alpha_1, \beta_2) = \rho^a_{\alpha_1} \cdot \rho^b_{\beta_2} \tag{6.19}$$
$$P(\alpha_2, \beta_1) = \rho^a_{\alpha_2} \cdot \rho^b_{\beta_1} \tag{6.20}$$
$$P(\alpha_2, \beta_2) = \rho^a_{\alpha_2} \cdot \rho^b_{\beta_2} \tag{6.21}$$

The correlated case, instead, allows to assign a probability directly over a certain combination of pure strategies, thanks to the ability of the leader to coordinate the followers' behaviour:

$$P(\alpha_1, \beta_1) = p(\alpha_1, \beta_1) \tag{6.22}$$
$$P(\alpha_1, \beta_2) = p(\alpha_1, \beta_2) \tag{6.23}$$
$$P(\alpha_2, \beta_1) = p(\alpha_2, \beta_1) \tag{6.24}$$
$$P(\alpha_2, \beta_2) = p(\alpha_2, \beta_2) \tag{6.25}$$

Part 2: To provide a complete proof we have to show that does not exist any game where Stackelberg equilibrium provides better outcomes than correlated case, or, equivalently, all the feasible solutions of the Stackelberg problem are also solutions of the correlated problem. Then assume, for the sake of contradiction, the existence of a game where Stackelberg equilibrium is actually better than correlated case. In the first part we have already shown that unconstrained objective function of correlated equilibrium allows solutions better than the unconstrained objective function of the Stackelberg optimization problem. Now, we suppose that, in this game, all the correlated solutions better than the Stackelberg equilibrium are infeasible due to the constraints of the correlated problem:

$$\forall p : \tag{6.26}$$
$$\sum_{i \in D} \sum_{\bar{j} \in \bar{A}} p(i, \bar{j}) U_d(i, \bar{j}) > \sum_{i \in D} \delta_i \mathcal{V}_d(\Psi, U_d(i)) \tag{6.27}$$
$$\exists a \in \Psi, \quad \exists j, j' \in A_a : \quad \mathcal{V}_a(p, j', j) > \mathcal{V}_a(p, j, j) \tag{6.28}$$

However, we can easily find that if this condition is satisfied, Stackelberg equilibrium coincides with committing to correlated strategies. In fact, the coordinator is allowed to choose p such that

$$p(i, \bar{j}) = \delta_i \prod_{j \in \bar{A}} \rho_j^a \quad j \in A_a, \mathcal{S} = (\delta_i, \bar{j}) \tag{6.29}$$

Such strategy is feasible in the correlated problem iff it is feasible in the Stackelberg problem. In fact, if some player a plays a strategy j with a zero probability, the correlated constraint 4.82 holds trivially. Otherwise, when all players play a strategy with positive probability, the best response constraint of the Nash equilibrium, which is satisfied in the Stackelberg equilibrium, also satisfies the correlated constraint. This means that the coordinator is always allowed to coordinate the players to play Stackelberg equilibrium, making it a lower bound to the correlated problem. □

In order to use commitment to correlated strategies to determine an upper bound for each leader strategy we need to apply the correlated problem to the followers' subgame corresponding to leader strategy. Then, fixed the leader strategy the correlated strategy p and players' utilities will be bidimensional matrices. The resulting problem is shown below as Problem 13.

Problem 13: Correlated upper bound LP

$$\max_{p} \quad \sum_{j \in A_a} \sum_{k \in A_b} p(j,k) U_d(j,k) \tag{6.30}$$

$$\text{s.t.} \quad \sum_{k \in A_b} p(j,k) U_a(j',k) \leq \sum_{k \in A_b} p(j,k) U_a(j,k) \qquad \forall j, j' \in A_a \tag{6.31}$$

$$\sum_{j \in A_a} p(j,k) U_b(j,k') \leq \sum_{j \in A_a} p(j,k) U_b(j,k) \qquad \forall k, k' \in A_b \tag{6.32}$$

$$\sum_{j \in A_a} \sum_{k \in A_b} p(j,k) = 1 \tag{6.33}$$

$$p(j,k) \geq 0 \qquad \forall j \in A_a, \forall k \in A_b \tag{6.34}$$

MINLP Nash

In Section 4.8 we described a MIP formulation to calculate Nash equilibrium as a feasibility problem, allowing the optimization of an arbitrary objective. To provide a lower bound for the multi-follower equilibrium we need to maximize (SSE) or minimize (WSE) the leader's utility, then we add a nonlinear objective which transforms the original MIP into the MINLP proposed as Problem 14.

Problem 14: Nash MINLP

$$\max_{\rho_1,\rho_2} \quad \sum_{j\in A_a}\sum_{k\in A_b} \rho_j^a \rho_k^b U_d(j,k) \tag{6.35}$$

$$\text{s.t.} \quad \sum_{j\in A_a} \rho_j^a = 1 \qquad \forall a \in \Psi \tag{6.36}$$

$$u_j^a = \sum_{k\in C} \rho_k^b U_a(j,k) \qquad \forall j \in A_a, \forall a \in \Psi, b \neq a \tag{6.37}$$

$$v_a \geq u_j^a \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.38}$$

$$r_j^a = v_a - u_j^a \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.39}$$

$$\rho_j^a \leq 1 - s_j^a \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.40}$$

$$r_j^a \leq M s_j^a \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.41}$$

$$s_j^a \in \{0,1\} \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.42}$$

$$\rho_j^a \geq 0 \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.43}$$

$$\tag{6.44}$$

Variables z_{ijk} and z_{ij}^a have been introduced in order to relax objective nonlinearities through nonlinear complementarity constraints 6.50-6.53, however a nonlinearity is still present in constraint 6.50.

Single MINLP formulation

Another approach to exploit game structure is to formulate the whole LPFM problem with a single MINLP and solving it using a generic MINLP solver. In this way a single solver is tasked to perform both branch and bound on leader's strategies and computation of followers' Nash equilibrium. The MINLP in Problem 15 is obtained starting from MINLP Nash (Problem 14): the leader strategy δ has been introduced both in the objective calculation and in the calculation of follower's utilities u_j^a.

Problem 15: LPFM MINLP

$$\max_{\delta,\rho_1,\rho_2} \quad \sum_{i\in D}\sum_{j\in A_a}\sum_{k\in A_b} z_{ijk}U_d(i,j,k) \tag{6.45}$$

$$\text{s.t.} \quad \sum_{i\in D}\delta_i = 1 \tag{6.46}$$

$$\sum_{j\in A_a}\rho_j^a = 1 \qquad \forall a \in \Psi \tag{6.47}$$

$$\sum_{i\in D}\sum_{j\in A_a} z_{ij}^a = 1 \qquad \forall a \in \Psi \tag{6.48}$$

$$\sum_{i\in D}\sum_{j\in A_a}\sum_{k\in A_b} z_{ijk} = 1 \qquad \forall a,b \in \Psi, b \neq a \tag{6.49}$$

$$z_{ijk} \geq z_{ij}\rho_k^b - (1-\delta_i) \qquad \forall i \in D, \forall j \in A_a, \forall k \in A_b \tag{6.50}$$

$$z_{ijk} \leq \delta_i \qquad \forall i \in D, \forall j \in A_a, \forall k \in A_b \tag{6.51}$$

$$z_{ij}^a \geq \rho_j^a - (1-\delta_i) \qquad \forall i \in D, \forall j \in A_a, \forall a \in \Psi \tag{6.52}$$

$$z_{ij}^a \leq \delta_i \qquad \forall i \in D, \forall j \in A_a, \forall a \in \Psi \tag{6.53}$$

$$u_j^a = \sum_{i\in D}\sum_{k\in C} z_{ik}^a U_a(i,j,k) \qquad \forall j \in A_a, \forall a \in \Psi, b \neq a \tag{6.54}$$

$$v_a \geq u_j^a \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.55}$$

$$r_j^a = v_a - u_j^a \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.56}$$

$$\rho_j^a \leq 1 - s_j^a \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.57}$$

$$r_j^a \leq M s_j^a \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.58}$$

$$s_j^a \in \{0,1\} \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.59}$$

$$\rho_j^a \geq 0 \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.60}$$

$$z_{ij}^a \geq 0 \qquad \forall i \in D, \forall j \in A_a, \forall a \in \Psi \tag{6.61}$$

$$z_{ijk} \geq 0 \qquad \forall i \in D, \forall j \in A_a, \forall k \in A_b \tag{6.62}$$

$$\delta_i \in \{0,1\} \qquad \forall i \in D \tag{6.63}$$

6.3.3 LMFP: Leader Mixed Followers Pure

In this section we analyze the problem of calculating the optimal mixed strategy for the leader when the followers are forced to play in pure strategies. When the leader commits to a mixed strategy, a normal form follower subgame is generated, whose payoffs correspond to the weighted sum of the payoffs of the normal form games related to the leader's pure strategies in

the support of the mixed strategy. As known from theory a pure strategies Nash equilibrium in a normal form game not always exists. Then, also in LMFP games, being pure strategies Nash equilibrium part of the solution, a leader-follower equilibrium is not granted to exist. Take for example the game in Figure 6.2.

l_1	β_1	β_2
α_1	3, 3, 6	8, 0, 7
α_2	8, 1, 6	2, 6, 3

l_2	β_1	β_2
α_1	4, 2, 4	3, 6, 9
α_2	7, 0, 7	2, 7, 2

Figure 6.2: An example of infeasible game.

When the leader commits to l_1 the followers cannot play a pure Nash equilibrium. In fact, if the follower a plays α_1, we have

$$U_b(\alpha_1, \beta_1) < U_b(\alpha_1, \beta_2) \tag{6.64}$$

which makes the follower b to prefer to play the pure strategy β_2. But when he plays β_2, the follower a will prefer to play α_2 as

$$U_a(\alpha_1, \beta_2) < U_a(\alpha_2, \beta_2) \tag{6.65}$$

When the follower a plays α_2, the other follower will prefer to play β_1 because

$$U_b(\alpha_2, \beta_2) < U_b(\alpha_2, \beta_1) \tag{6.66}$$

Finally, if the follower b plays β_1, the best response of the follower a would be α_1 as

$$U_a(\alpha_2, \beta_1) < U_a(\alpha_1, \beta_1) \tag{6.67}$$

Then, in the subgame generated when the leader commits to l_1, do not exist two pure strategies which are mutual best responses, which is the necessary and sufficient condition for the existence of a pure Nash equilibrium.

The same reasoning is also valid when the leader commits to l_2. Moreover we can easily see that in any game generated as a linear combination of these two games, no pure Nash equilibria exist. In fact also the preferences of the followers in the second subgame are the same of those described with inequalities 6.64, 6.65, 6.66 and 6.67. Then, as shown in Figure 6.13, each mixed leader strategy will generate a subgame in which those preferences still hold making the LMFP game infeasible. The game presented in Figure 6.4, instead, admits some feasible solutions.

l_1	β_1	β_2
α_1	3, 3, 6	8, 0, 7
α_2	8, 0, 6	2, 6, 3

l_2	β_1	β_2
α_1	4, 0, 4	3, 7, 3
α_2	7, 1, 7	2, 6, 8

Figure 6.4: An example of a game admitting a LMFP equilibrium.

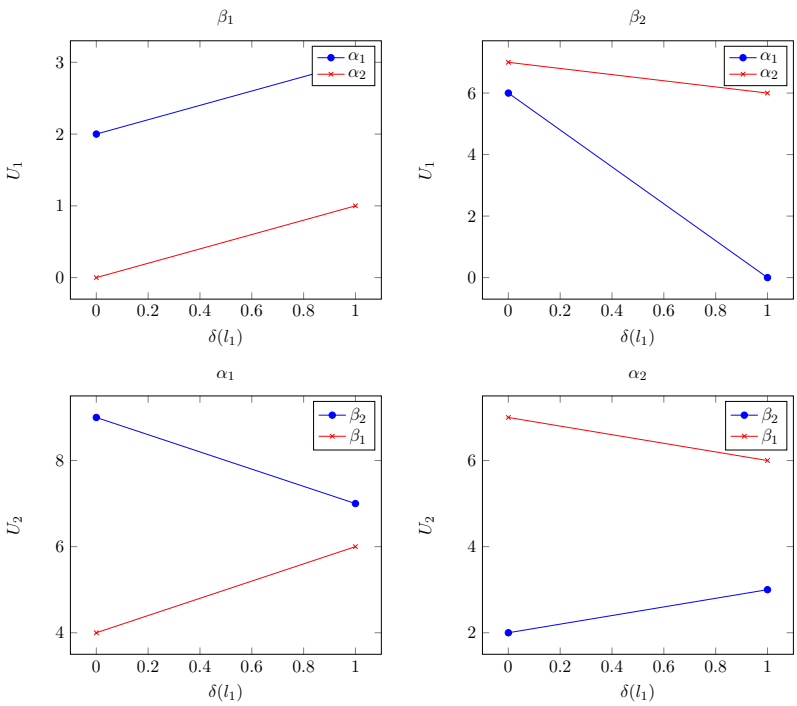


Figure 6.3: Plots showing how the utilities of the followers in game in Figure 6.2 change by varying the leader strategy, once fixed the followers' pure strategies. The upper plots shows the utility of the first follower while the lower plots shows the utility of the second follower. The left plots shows assume the other follower playing his first pure strategy while the right plots assume that the other followers play his second pure strategy. The red line represents the utility the player gets when he plays his first action, while the blue line is the utility he gets by playing his second action. We note that the followers' preferences do not change for any mixed strategy the leader can commit to.

In fact, although both the games associated with leader's pure strategies are infeasible, there exist some mixed strategies generating subgames where a pure strategies Nash equilibrium does exist. Figure 6.5 shows that when the leader commits to a strategy such that $0.25 \leq \delta_1 \leq 0.5$, the followers' strategies α_1 and β_1 are mutual best responses, while for any other strategy the game is infeasible. In order to provide a solution concept to this type of problems we define the optimal solution as the mixed leader strategy which generates the followers' subgame that allows the followers' pure strategies Nash equilibrium with the highest leader's payoff. While calculating the

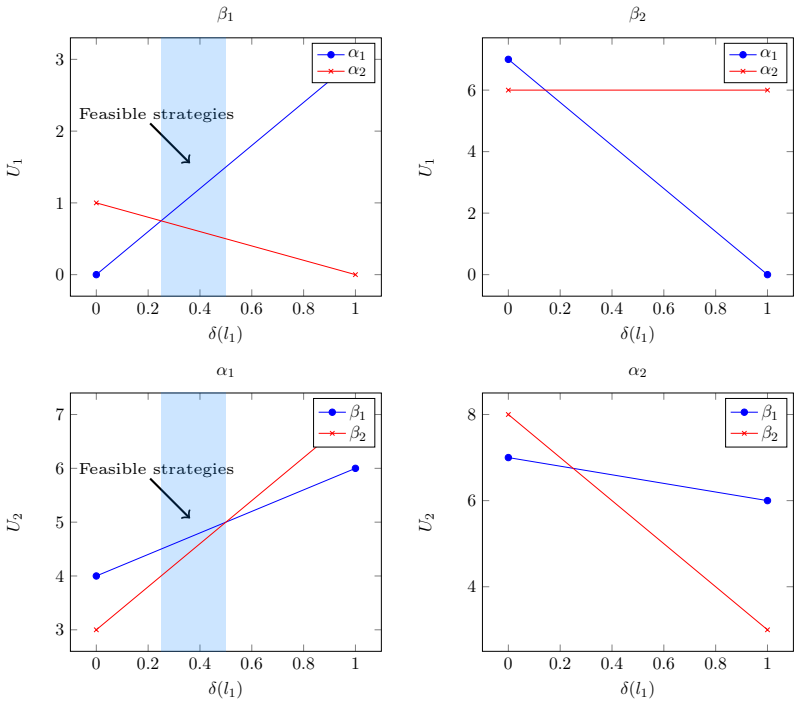


Figure 6.5: Plots showing how the followers' utilities change with respect to leader strategy in game 6.4. The region highlighted is the only feasible region of the leader strategy domain.

followers' pure strategies Nash equilibrium in a single game once the leader committed to a mixed strategy is a simple task, the hardness of solving this problem is due to the infinite number of followers' subgames that the leader can generate by committing to a mixed strategy. However, we can look at the problem from another perspective. In fact, once fixed a pure Nash equilibrium for the followers, the problem of maximizing the leader's utility subject to constraints of this equilibrium is a LP.

In this context, the problem of finding the LMFP Stackelberg equilibrium is reduced to the solution of a finite number of LPs in the order of $|A_a| \cdot |A_b|$.

For each couple of followers' strategies, the leader will search a mixed strategy such that the followers' strategy profile is a Nash equilibrium and then he tries to maximize his own utility. Then, fixed the pure followers' strategies $\bar{j}$ and $\bar{k}$, each LP would be in the form of problem 6.68. To speed up the convergence of this MULTI-LP, a branch and bound algorithm similar to that used to solve LPFM can be applied in order to choose which strategy profiles to analyze and which to exclude. A good enough upper bound for real world security games could be simply determined by observing the best possible leader's payoff for each pure strategy profile of the followers, while the lower bound can be progressively refined by resolving LP 6.68 for each profile.

Problem 16: LMFP LP

$$\max_{\delta} \quad \sum_{i \in D} \delta_i U_d(i, \bar{j}, \bar{k}) \tag{6.68}$$

$$\text{s.t.} \quad \sum_{i \in D} \delta_i U_a(i, \bar{j}, \bar{k}) \geq \sum_{i \in D} \delta_i U_a(i, j, \bar{k}) \tag{6.69}$$

$$\sum_{i \in D} \delta_i U_b(i, \bar{j}, \bar{k}) \geq \sum_{i \in D} \delta_i U_b(i, \bar{j}, k) \tag{6.70}$$

$$\sum_{i \in D} \delta_i = 1 \tag{6.71}$$

$$\delta_i \geq 0 \tag{6.72}$$

Another way to calculate the equilibrium, exploiting the game structure without solving a LP for each pure followers' strategy, is to formulate the problem as the following MIP. We start by defining the followers strategies ρ_j^1 and ρ_k^2 as the probability of playing a certain action, which, being pure strategies, can be only 0 or 1. The leader strategy, instead, is a probability distribution over all its possible actions. In this context we define the leader's expected utility as the sum of the payoffs corresponding to the followers' couple of strategies weighted on the leader strategy. The constraint 6.75 allows to consider only the payoffs corresponding to the followers' chosen strategies. The remaining constraints 6.78, 6.79 and 6.80 are all needed to determine ρ_a and ρ_b as mutual best responses, such that the followers play a Nash equilibrium.

Problem 17: LMFP MIP

$$\max_{\delta,\rho_1,\rho_2} \quad \sum_{j\in A_a}\sum_{k\in A_b} u^d_{jk} \tag{6.73}$$

$$\text{s.t.} \quad u^d_{jk} \le \sum_{i\in D} \delta_i U_d(i,j,k) \qquad \forall j \in A_a, \forall k \in A_b \tag{6.74}$$

$$u^d_{jk} \le M\rho^a_j \qquad \forall j \in A_a, \forall k \in A_b \tag{6.75}$$

$$\delta_i \in [0,1] \qquad \forall i \in D \tag{6.76}$$

$$\sum_{i\in D} \delta_i = 1 \tag{6.77}$$

$$u^a_{jk} \le \sum_{i\in D} \delta_i U_a(i,j,k) \qquad \forall i \in D, \forall k \in A_b \tag{6.78}$$

$$u^a_{jk} \ge v_a - M(2-\rho^a_j-\rho^b_k) \qquad \forall j \in A_a, k \in A_b \tag{6.79}$$

$$v_a \ge \sum_{i\in D} \delta_i U_a(i,j,k) - M(1-\rho^a_j) \qquad \forall j \in A_a, k \in A_b \tag{6.80}$$

$$\sum_{j\in A_a} \rho^a_j = 1 \tag{6.81}$$

$$\rho^a_j \in \{0,1\} \qquad \forall \sigma_{\psi_1} \in \Sigma_{\psi_1} \tag{6.82}$$

6.3.4 LMFM: Leader Mixed Followers Mixed

Finally, we analyze the most general problem, where all the players are allowed to play in mixed strategies. As known from theory, the problem of finding a Stackelberg equilibrium in mixed strategies in games with more than two players is NP-hard [23]. We provide two formulations of this problem: NLP and MINLP. The first calculates a feasible good solution more quickly, using a nonlinear optimizer like SNOPT, however the optimality of the solution provided is not granted. The second formulation allows to always find the optimal solution in a finite time, using a mixed-integer optimizer like COUENNE or SCIP, but requires much more time to calculate it. Then, we provide an analysis on the solution times as well as value of the solution provided, in order to give the reader an idea of which approach to choose, depending on the application domain.

LMFM NLP

The formulation of this problem as a NLP can be easily obtained by restricting the general multi-follower formulation presented at the start of

this chapter, to only two followers. The resulting formulation is reported as Problem 18.

Problem 18: LMFM NLP

$$\max_{\delta,\rho_1,\rho_2} \quad \sum_{i\in D}\sum_{j\in A_a}\sum_{k\in A_b} \delta_i \rho_j^a \rho_k^b U_d(i,j,k) \tag{6.83}$$

$$\text{s.t.} \quad \sum_{i\in D}\sum_{k\in A_b} v_a - \delta_i \rho_k^b U_a(i,j,k) \geq 0 \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.84}$$

$$\sum_{i\in D}\sum_{k\in A_b} \rho_j^a v_a - \delta_i \rho_j^a \rho_k^b U_a(i,j,k) = 0 \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.85}$$

$$\sum_{j\in A_a} \rho_j^a = 1 \qquad \forall a \in \Psi \tag{6.86}$$

$$v_a \in \mathbb{R} \qquad \forall a \in \Psi \tag{6.87}$$

$$\sum_{i\in D} \delta_i = 1 \tag{6.88}$$

Nevertheless its simplicity this formulation is extremely hard to solve to optimum due to nonconvex constraint 6.85. Instead, we solve this problem using SNOPT, a large scale optimizer based on Sequential Quadratic Programming (SQP) [14]. This optimizer explores feasible solutions trying to maximize the solution value in a very small time although it is almost never able to provide the optimal solution to this problem.

A way to improve the value of the best solution found is to exploit the extreme speed of this optimizer by running the optimization several times, each time starting from a different random strategy, using the best known feasible solution as a lower bound to the problem. A good randomization algorithm is crucial for the success of this method, in fact, a simple uniform randomization would make the algorithm start always almost from the same point when increasing the number of players' strategies. If each single strategy is chosen using a uniform distribution and then the result is divided for the sum of all variables, the sum of all the variables will be 1, as required, but the value of each variable will tend to zero when increasing the number of actions.

A better way to choose the starting point is to randomize uniformly over the leader strategy and then calculate the Nash equilibrium in the resulting followers subgame using MIP Nash or Lemke-Howson. In this way SNOPT always provides a feasible solution as the starting point is a feasible solution itself. Moreover, the nonlinear nature of this problem causes the followers' Nash equilibria to be very different from each other as the leader strategy

acts as a parameter in the nonlinear system whose equilibria are all the feasible solutions to the multi-follower problem. In other words, a small perturbation on the leader strategy could significantly change the number and the topology of the followers' equilibria. In this way SNOPT can explore different portions of the domain increasing the probability of finding a good solution. However, with the increasing the number of the players' actions, presolving either with MIP Nash or Lemke-Howson becomes too expensive, while the solution provided remains far from being optimal. Actually, the best starting point can be easily determined by presolving with LPFP. In a leader-follower normal form game with uniform random payoffs, the value of the solution of the LPFP problem tends to LMFM optimum with the number of players' actions tending to infinity. In fact, it is intuitive to see that, with an increasing number of actions of the players, the probability of finding a good LPFP solution tends to 1. In this way it is no more necessary to rerun the LMFM_NLP algorithm several times as the first solution provided is equal to or better than the LPFP solution, which is very close to optimum, allowing to solve large scale security problems in a small time.

LMFM MINLP

Now we propose another formulation obtained starting from the MINLP Nash problem 6.3.2 and adding the leader strategy in the calculation of the followers' utilities u_j^a as well as in the objective.

Problem 19: LMFM MINLP

$$\max_{\delta,\rho_1,\rho_2} \quad \sum_{i\in D}\sum_{j\in A_a}\sum_{k\in A_b} \delta_i \rho_j^a \rho_k^b U_d(i,j,k) \qquad \forall a,b \in \Psi, b \neq a \quad (6.89)$$

$$\text{s.t.} \quad \sum_{i\in I} \delta_i = 1 \quad (6.90)$$

$$\sum_{j\in A_a} \rho_j^a = 1 \qquad \forall a \in \Psi \quad (6.91)$$

$$u_j^a = \sum_{i\in I}\sum_{k\in A_b} \delta_i \rho_k^b U_a(i,j,k) \qquad \forall j \in A_a,\ \forall a,b \in \Psi, b \neq a \quad (6.92)$$

$$v_a \geq u_j^a \qquad \forall j \in A_a, \forall a \in \Psi \quad (6.93)$$

$$r_j^a = v_a - u_j^a \qquad \forall j \in A_a, \forall a \in \Psi \quad (6.94)$$

$$\rho_j^a \leq 1 - s_j^a \qquad \forall j \in A_a, \forall a \in \Psi \quad (6.95)$$

$$r_j^a \leq M s_j^a \qquad \forall j \in A_a, \forall a \in \Psi \quad (6.96)$$

$$s_j^a \in \{0,1\} \qquad \forall j \in A_a, \forall a \in \Psi \quad (6.97)$$

$$\rho_j^a \geq 0 \qquad \forall j \in A_a, \forall a \in \Psi \quad (6.98)$$

$$\delta_i \geq 0 \qquad \forall i \in D \quad (6.99)$$

(6.100)

This mixed-integer formulation of the multi-follower problem can be solved to optimum using a mixed-integer optimizer like COUENNE or SCIP. In order to obtain a faster convergence to the solution we introduce some additional variables and constraints to the LMFM MINLP:

$$\begin{aligned} z_{ij}^a &= \delta_i \cdot \rho_j^a \qquad \forall i \in D, \forall j \in A_a, \forall a \in \Psi \\ z_{ijk} &= z_{ij}^a \cdot \rho_k^b \quad \forall i \in D, \forall j \in A_a, \forall k \in A_b \end{aligned} \quad (6.101)$$

The variables here introduced represent the joint probabilities that two pure strategies are in the support of the mixed strategy at the same time. Because these variables are probabilities too, we need to introduce the following set of constraints:

$$0 \leq z_{ij}^a \leq 1 \qquad \forall i \in D, \forall j \in A_a, \forall a \in \Psi \quad (6.102)$$

$$0 \leq z_{ijk} \leq 1 \quad (6.103)$$

$$\sum_{i\in D}\sum_{j\in A_a} z_{ij}^a = 1 \quad (6.104)$$

$$\sum_{i\in D}\sum_{j\in A_a}\sum_{k\in A_b} z_{ij}^a = 1 \quad (6.105)$$

Finally, we rewrite the problem 19 as the following problem:

Problem 20: LMFM MINLP with z variables

$$\max_{\delta,\rho_1,\rho_2} \quad \sum_{i\in D}\sum_{j\in A_a}\sum_{k\in A_b} z_{ijk}U_d(i,j,k) \tag{6.106}$$

$$\text{s.t.} \quad \sum_{i\in I}\delta_i = 1 \tag{6.107}$$

$$\sum_{j\in A_a}\rho_j^a = 1 \qquad \forall a\in\Psi \tag{6.108}$$

$$\sum_{i\in D}\sum_{j\in A_a} z_{ij}^a = 1 \qquad \forall a\in\Psi \tag{6.109}$$

$$\sum_{i\in D}\sum_{j\in A_a}\sum_{k\in A_b} z_{ijk} = 1 \qquad \forall a,b\in\Psi, b\neq a \tag{6.110}$$

$$u_j^a = \sum_{i\in I}\sum_{k\in A_b} z_{ik}^b U_a(i,j,k) \qquad \forall j\in A_a,\ \forall a,b\in\Psi, b\neq a \tag{6.111}$$

$$v_a \geq u_j^a \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.112}$$

$$r_j^a = v_a - u_j^a \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.113}$$

$$\rho_j^a \leq 1 - s_j^a \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.114}$$

$$r_j^a \leq M s_j^a \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.115}$$

$$s_j^a \in \{0,1\} \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.116}$$

$$\rho_j^a \geq 0 \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.117}$$

$$z_{ij}^a \geq 0 \qquad \forall i\in D, \forall j\in A_a, \forall a\in\Psi \tag{6.118}$$

$$z_{ijk} \geq 0 \qquad \forall i\in D, \forall j\in A_a, \forall k\in A_b \tag{6.119}$$

$$\delta_i \geq 0 \qquad \forall i\in D \tag{6.120}$$

6.4 Polymatrix security games with two attackers

Normal form multiplayer games are described using a multidimensional multi-matrix whose size increases exponentially with the number of players. In some games, however, it could be reasonable to decompose the multiplayer game into many two-player games. These games are called polymatrix games or multimatrix games. In these games, each player plays a different two-player game against each other player. Thanks to such structure, the size of these games grows quadratically in the number of players' actions. The expected utility of each player is calculated as the sum of the utilities he gets in all the games he plays, then, polymatrix games can suit well to multi-follower security problems where the outcomes of two players are not affected by the other players' behavior.

Papadimitriou et al. recently demonstrated that the problem of finding a Nash equilibrium in general-sum succinct games, to which the class of polymatrix games belongs, is PPAD-complete even in two-player games,

while under the zero-sum hypothesis the complexity is PPAD for games with more than two players [11]. Like in the classic leader-follower case, in a leader-follower polymatrix game the leader first commits to a strategy, then the followers calculate their best response in the consequent simultaneous game, however, unlike the previous case, the bimatrix of the followers' game is not modulated by the leader strategy, which simply adds a certain outcome to their expected utility. In this context, the aim of the leader is to commit to a strategy such that the best response of the followers maximizes his own utility. The expected utility of the leader, $\mathcal{V}_d$, can be defined as

$$\mathcal{V}_d = \sum_{a \in \Psi} \sum_{i \in D} \sum_{j \in A_a} \delta_i \rho_j^a U_d^a(i,j) \tag{6.121}$$

where U_d^a is the payoffs matrix of the leader d against follower a. In the same way we define, for each follower $a \in \Psi$ his expected utility as

$$\mathcal{V}_a = \sum_{j \in A_a} \sum_{i \in D} \delta_i \rho_j^b U_a^d(i,j) + \sum_{j \in A_a} \sum_{k \in A_b} \rho_j^a \rho_k^b U_a^b(j,k) \tag{6.122}$$

where U_a^d is the payoffs matrix of the follower a playing against the leader and U_a^b is his payoffs matrix when playing against other follower. Again, as done for the previous problem, we propose four formulations, depending on whether the players are allowed to play in mixed strategies or not.

6.4.1 Polymatrix LPFP

The *pure-pure* formulation of the problem is very similar to the LPFP algorithm described before. For each combination of pure strategies, for each follower we check if a follower can improve his utility by deviating individually from that strategy. If both the followers benefit in playing that strategy, it is an equilibrium. If it is the equilibrium that up to now provides the best utility to the leader, it is temporarily labeled as the leader-follower solution of the game. Finally, when all the combinations of pure strategies have been enumerated, the temporary solution is flagged as optimal and the algorithm ends. If no optimal solution has been found, the game is declared infeasible. To show an example of application we apply the algorithm to the problem in Table 6.4.1.

L/F_1	α_1	α_2
l_1	3 , 3	8 , 0
l_2	8 , 0	2 , 5

L/F_2	β_1	β_2
l_1	4 , 0	3 , 7
l_2	7 , 1	2 , 6

F_1/F_2	β_1	β_2
α_1	2 , 1	1 , 3
α_2	5 , 8	2 , 3

First, we suppose the players playing (l_1, α_1, β_1) with players' outcomes being $(7, 5, 1)$. Such strategy is not an equilibrium as F_2 is better to defect to α_2,

obtaining an utility of 10. Next, the algorithm analyzes (l_1, α_1, β_2). None of the followers prefers to defect, then this strategy is temporarily flagged as the leader-follower solution in which the expected payoff of the leader is 6. The next strategy to analyze is (l_1, α_2, β_1) whose expected outcome is $(12, 5, 2)$. Again, F_2 prefers to defect. Later, proceeding in this order, the algorithm finds (l_2, α_2, β_1) as a possible leader-follower equilibrium, with leader's outcome of 9. Being this solution better than the previous, the best solution is updated. Finally the algorithm checks (l_2, α_2, β_2) which is not an equilibrium because F_1 is better to play α_2.

It is important to note that the solution found is not a Nash equilibrium, because under that solution concept the leader would prefer to play (l_2, α_2, β_1) which has an outlook of 12. The time complexity of this algorithm, fixed the number of players, is polynomial in $\mathcal{O}(n^p)$ where n is the number of actions of each of the p players.

6.4.2 Polymatrix LMFP

Now we extend the LPFP problem by allowing the leader to play in mixed strategies. In this way, the leader can possibly obtain a better outcome than that obtained in pure strategies. For example the value of the LPFP solution of the game in Table 6.4.1 is 9. This value can be improved by committing to a mixed strategy such that the followers benefit from staying in a pure equilibrium, while the leader maximizes his own utility until the followers are indifferent. Compliance principle will assure that they will not move away from the equilibrium. As shown for the LMFP multi-follower problem, a feasible LMFP strategy not always exists. A trivial example of infeasible problem is shown in table 6.4.2.

L/F_1	α_1	α_2
l_1	3 , 0	8 , 0
l_2	8 , 0	2 , 0

L/F_2	β_1	β_2
l_1	4 , 0	3 , 0
l_2	7 , 0	2 , 0

F_1/F_2	β_1	β_2
α_1	0 , 1	1 , 0
α_2	1 , 0	0 , 1

In this game the followers are not affected by the leader strategy, then an equilibrium of the game can only exist if a Nash equilibrium exists in the followers' game. Being the followers' utilities indifferent to leader strategy and not existing a pure Nash equilibrium in the followers' game, a LMFP equilibrium does not exist. In the game reported in Table 6.4.1, instead, the leader could commit to a strategy such that the followers will respond with their pure polymatrix Nash equilibrium (α_2, β_2). Such strategy is feasible if both followers are not better to defect.

By calculating followers utility using formula 6.122, we find that follower F_1 prefers to stay in the equilibrium only if

$$u_1 = 5 \cdot l_2 + 2 \geq 3 \cdot l_1 + 1 \tag{6.123}$$

This forces the leader to play l_2 with a probability of at least 0.25. Follower F_2, instead, will always stay in the equilibrium for any leader strategy.

$$u_2 = 7 \cdot l_1 + 6 \cdot l_2 + 3 \geq l_2 + 8 \tag{6.124}$$

The leader would prefer to play l_1, which gives to him an higher utility, however, in order to maintain follower a compliant, he will commit to the optimal strategy $(0.75, 0.25)$, which gives to him an utility of 9.25.

A mathematical formulation of this problem can be obtained by adapting the concept of regret to the polymatrix scenario. As done for LMFM MINLP, we impose that followers' strategies are part of the equilibrium only if the associated regret r is zero:

$$0 \leq r_j^a \leq M(1 - \rho_j^a) \qquad \forall a \in \Psi, \forall j \in A_a \tag{6.125}$$

Then we calculate the regret associated with j-th action as the difference between best response utility $\hat{\mathbf{u}}_a$ and the payoff u_j^a obtained by playing ρ_j^a, that is

$$r_j^a = v_a - u_j^a \qquad \forall a \in \Psi, \forall j \in A_a \tag{6.126}$$

The best response v_a of follower a to adversaries' strategy is simply the greatest utility obtained in all possible strategy

$$v_a \geq u_j^a \qquad \forall a \in \Psi, \forall j \in A_a \tag{6.127}$$

Constraints 6.127 and 6.126 combined implies the non-negativity of regrets. We define u_d as a matrix such that each element is allowed to be greater than zero only if both followers play the corresponding pure strategy. If the followers play j-th and k-th strategy respectively, which means $\rho_j^a = 1$ and $\rho_k^b = 1$, then the element u_{jk}^d is upper bounded by the definition of leader's utility 6.121. Under these constraints, which are reported as 6.131 and 6.132, maximizing the sum of all the elements of u_d is the same as calculating the leader's utility in the leader-follower equilibrium. The resulting MIP is below reported as Problem 21.

Problem 21: LMFP Polymatrix

$$\max_{\delta,\rho_a,\rho_b} \quad \sum_{j\in A_a}\sum_{k\in A_b} u^d_{jk} \tag{6.128}$$

$$\text{s.t.} \quad \sum_{i\in D} \delta_i = 1 \tag{6.129}$$

$$\sum_{j\in A_a} \rho_a = 1 \qquad \forall a \in \Psi \tag{6.130}$$

$$u^d_{jk} \leq \sum_{i\in D} \delta_i (U^a_d(i,j) + U^b_d(i,k)) + M(2-\rho^a_j-\rho^b_k) \qquad \forall j \in A_a, \forall k \in A_b \tag{6.131}$$

$$u^d_{jk} \leq M\rho^a_j \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.132}$$

$$u^a_j = \sum_{i\in D} \delta_i U^d_a(i,j) + \sum_{k\in A_b} \rho^b_k U^b_a(j,k) \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.133}$$

$$v_a \geq u^a_j \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.134}$$

$$r^a_j = v_a - u^a_j \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.135}$$

$$r^a_j \leq M(1-\rho^a_j) \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.136}$$

$$\rho^a_j \in \{0,1\} \qquad \forall j \in A_a, \forall a \in \Psi \tag{6.137}$$

$$\delta_i \geq 0 \qquad \forall i \in D \tag{6.138}$$

6.4.3 Polymatrix LPFM

Now we consider the case in which the followers are allowed to play in mixed strategies while the leader is restricted to play in pure strategies. This problem can be formulate both as a single MIP or a MULTI-MIP by applying a branch-and-bound over leader's strategies, as done for LPFM in normal form.

Single MIP

By committing to a pure strategy, the leader simply introduces a constant offset to the followers' utilities. Then, the followers, aware of the commitment, calculate their strategies as mutual best responses. Being the leader strategy pure, the objective can be linearized by defining for each follower a auxiliary variables z^a_{ij} which represent the joint probability that leader plays pure strategy δ_i and follower a assigns probability ρ^a_j to action j. Variables z^a_{ij} are upper bounded by leader strategy δ_i (6.141) and lower bounded by follower's strategy ρ^a_j only when the leader plays pure strategy δ_i(6.140). The followers' utilities (6.142) and regrets (6.144 and 6.145) are defined as done in the LMFP problem but now we need to reintroduce binary variables s^a_j needed to switch strategies on and off. The resulting MIP is shown as Problem 22.

Problem 22: LPFM Polymatrix single MIP

$$\max_{\delta,\rho_a,\rho_b} \quad \sum_{a\in\Psi}\sum_{i\in D}\sum_{j\in A_a} z_{ij}^a U_d^a(i,j) \tag{6.139}$$

$$\text{s.t.} \quad z_{ij}^a \geq \rho_j^a - (1-\delta_i) \qquad \forall i\in D, \forall j\in A_a, \forall a\in\Psi \tag{6.140}$$

$$z_{ij}^a \leq \delta_i \qquad \forall i\in D, \forall j\in A_a, \forall a\in\Psi \tag{6.141}$$

$$u_j^a = \sum_{i\in D}\delta_i U_a^d(i,j) + \sum_{k\in A_b}\rho_k^b U_a^b(j,k) \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.142}$$

$$v_a \geq u_j^a \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.143}$$

$$r_j^a = v_a - u_j^a \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.144}$$

$$r_j^a \leq M s_j^a \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.145}$$

$$s_j^a \in \{0,1\} \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.146}$$

$$\sum_{i\in D}\delta_i = 1 \tag{6.147}$$

$$\sum_{j\in A_a}\rho_a = 1 \qquad \forall a\in\Psi \tag{6.148}$$

$$0 \leq \rho_j^a \leq 1 - s_j^a \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.149}$$

$$\delta_i \geq 0 \qquad \forall i\in D \tag{6.150}$$

MULTI-MIP

The second approach consists in decomposing the whole problem into many easier subproblems as done for the normal form LPFM problem. Also in this case we can analyze three different followers' behaviors:

- The followers mutually try to maximize the leader's utility.
- The followers mutually try to minimize the leader's utility.
- The followers simply choose a Nash equilibrium.

The branch-and-bound algorithm is the same of that used in the normal form case (Algorithm 10) with exception of upper bound and lower bound calculations as well as subproblems construction. Also in this case, for each leader's pure strategy i, we build a problem in which the followers plan their strategy given the leader's commitment. Thanks to the polymatrix structure, in each subproblem the leader's utility can now be expressed linearly as:

$$\mathcal{V}_d = \sum_{a\in\Psi}\sum_{j\in A_a}\rho_a U_d^a(j) \tag{6.151}$$

The two followers' utility bimatrices can now be reduced to a single bimatrix $(\hat{\mathbf{U}}_a^b, \hat{\mathbf{U}}_b^a)$ in which, to every payoff of the original follower's game, we add

the constant offset introduced by the leader's commitment:

$$\hat{\mathbf{U}}_a^b(j,k) = U_a^b(j,k) + U_a^d(i,j) \qquad \forall j \in A_a, \forall k \in A_b, \forall a, b \in \Psi, a \neq b \quad (6.152)$$

This bimatrix represents the payoffs of a two-player normal form game which encloses all the information needed to calculate followers' equilibrium. Then, an upper bound for each leader's pure strategy can be calculated by replacing objective 6.30 with leader's utility 6.151 and calculating followers' mutual best response on $(\hat{\mathbf{U}}_a^b, \hat{\mathbf{U}}_b^a)$. By replacing the objective, two additional variables need to be added to the problem, representing the strategies of the followers. Being $p(j,k)$ the joint probability of followers playing j and k we define ρ_j^a for each follower a as the marginal probability of follower a playing strategy j:

$$\rho_j^a = \sum_{k \in A_b} p(j,k) \qquad \forall j \in A_a, \forall a \in \Psi, a \neq b \quad (6.153)$$

$$(6.154)$$

Because of the objective maximization the equation 6.153 can be relaxed to an inequality, then the upper bound of each leader's pure strategy can be obtained by solving the following LP:

Problem 23: Correlated equilibrium Polymatrix LP

$$\max_{p} \quad \sum_{a \in \Psi} \sum_{j \in A_a} \rho_a U_d^a(j) \quad (6.155)$$

$$\text{s.t.} \quad \rho_j^a \leq \sum_{k \in A_b} p(j,k) \qquad \forall j \in A_a, \forall a, b \in \Psi, a \neq b \quad (6.156)$$

$$\sum_{k \in A_b} p(j,k)\hat{\mathbf{U}}_a(j',k) \leq \sum_{k \in A_b} p(j,k)\hat{\mathbf{U}}_a(j,k) \qquad \forall j, j' \in A_a \quad (6.157)$$

$$\sum_{j \in A_a} p(j,k)\hat{\mathbf{U}}_b(j,k') \leq \sum_{j \in A_a} p(j,k)\hat{\mathbf{U}}_b(j,k) \qquad \forall k, k' \in A_b \quad (6.158)$$

$$\sum_{j \in A_a} \sum_{k \in A_b} p(j,k) = 1 \quad (6.159)$$

$$p(j,k) \geq 0 \qquad \forall j \in A_a, \forall k \in A_b \quad (6.160)$$

After calculating an upper bound for each action, they are sorted in decreasing order. A first sweep to remove suboptimal strategies is performed by calculating a lower bound solving for LPFP, once fixed the leader strategy. Note that solving normal form LPFP on the subproblem followers' bimatrix, maximizing 6.151 is indifferent of solving LPFP polymatrix problem with leader strategy fixed. Finally, each subgame in order, is solved to optimum until the solution calculated is greater of or equal to next strategy upper

bound. To calculate a solution for each subproblem we apply definitions 6.151 and 6.152 to MINLP 6.3.2 thus obtaining the following MIP:

Problem 24: LPFM Polymatrix MIP

$$
\begin{aligned}
\max_{\rho_1,\rho_2} \quad & \sum_{a\in\Psi}\sum_{j\in A_a} \rho_a U_d^a(j) & & \\
\text{s.t.} \quad & \sum_{j\in A_a} \rho_j^a = 1 & \forall a \in \Psi & \\
& u_j^a = \sum_{k\in C} \rho_k^b \hat{\mathbf{U}}_a^b(j,k) & \forall j \in A_a, \forall a \in \Psi, b \neq a & \\
& v_a \geq u_j^a & \forall j \in A_a, \forall a \in \Psi & (6.161)\\
& r_j^a = v_a - u_j^a & \forall j \in A_a, \forall a \in \Psi & \\
& \rho_j^a \leq 1 - s_j^a & \forall j \in A_a, \forall a \in \Psi & \\
& r_j^a \leq M s_j^a & \forall j \in A_a, \forall a \in \Psi & \\
& s_j^a \in \{0,1\} & \forall j \in A_a, \forall a \in \Psi & \\
& \rho_j^a \geq 0 & \forall j \in A_a, \forall a \in \Psi &
\end{aligned}
$$

Under the hypothesis that followers act to minimize leader's utility the only difference with this formulation is the minimization of the objective. In the case that followers only care to play any Nash equilibrium, the subproblem can be solved using both MIP-Nash or Lemke-Howson, but the second generally provides a quicker convergence.

6.4.4 Polymatrix LMFM

The general *mixed-mixed* formulation of leader-follower equilibrium in polymatrix games can be derived starting from the multi-follower LMFM problem by modifying the definitions of players' utilities. The objective of this problem is to maximize the expected utility of the leader defined in equation 6.121. Starting from 6.122, for each follower a, we define the expected utility u_j^a of each of his actions, given the other players strategy. The remaining constraints of the following MINLP come unchanged from the LMFM problem.

Problem 25: LMFM Polymatrix MINLP

$$\max_{\delta,\rho_a,\rho_b} \quad \sum_{i\in D}\sum_{j\in A_a}\delta_i\rho_j^a U_d(i,j) + \sum_{i\in D}\sum_{k\in A_b}\delta_i\rho_k^b U_d(i,k) \tag{6.162}$$

$$\text{s.t.} \quad \sum_{i\in D}\delta_i = 1 \tag{6.163}$$

$$\sum_{j\in A_a}\rho_a = 1 \qquad \forall a\in\Psi \tag{6.164}$$

$$u_j^a = \sum_{i\in D}\delta_i U_a^d(i,j) + \sum_{k\in A_b}\rho_k^b U_a^b(j,k) \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.165}$$

$$v_a \geq u_j^a \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.166}$$

$$r_j^a = v_a - u_j^a \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.167}$$

$$r_j^a \leq M s_j^a \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.168}$$

$$\rho_j^a \leq 1 - s_j^a \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.169}$$

$$\rho_j^a \geq 0 \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.170}$$

$$s_j^a \in \{0,1\} \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.171}$$

$$\delta_i \geq 0 \qquad \forall i\in D \tag{6.172}$$

Again, as in the LMFM multi-follower problem, the maximization of the leader's utility introduces a nonlinear objective, however, in this case all the constraints are linear. To speed up the convergence, as done for the previous problem, we introduce for each follower a, for each pure strategy profile (i,j) of its leader-follower game, a variable z_{ij}^a to represent the joint probability that he and the leader choose the pair of actions (i,j).

$$z_{ij}^a = \delta_i\rho_j^a \qquad \forall i\in D, \forall j\in A_a, \forall a\in\Psi \tag{6.173}$$

Being z_a a matrix of joint probabilities, every of its elements must be bounded between 0 and 1 and the sum of all its elements must be one. Although these requirements are yet satisfied having defined constraints over δ and ρ_a, those probability constraints over z_a seem to actually help in providing a faster convergence. Having specified these redundant constraints we are now allowed to relax the equality constraint 6.173 by replacing it with the inequality constraint 6.185, which also implies the non-negativity of z_{ij}^a. Putting it all together we obtain the following MINLP:

Problem 26: LMFM Polymatrix MINLP with z variables

$$\max_{\delta,\rho_1,\rho_2} \quad \sum_{i\in D}\sum_{j\in A_a} z^a_{ij}U_d(i,j) + \sum_{i\in D}\sum_{k\in A_b} z^b_{ik}U_d(i,k) \tag{6.174}$$

$$\text{s.t.} \quad \sum_{i\in D}\delta_i = 1 \tag{6.175}$$

$$\sum_{j\in A_a}\rho_a = 1 \qquad \forall a\in\Psi \tag{6.176}$$

$$\sum_{i\in D}\sum_{j\in A_a} z^a_{ij} = 1 \qquad \forall a\in\Psi \tag{6.177}$$

$$u^a_j = \sum_{i\in D}\delta_i U^d_a(i,j) + \sum_{k\in A_b}\rho^b_k U^b_a(j,k) \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.178}$$

$$v_a \geq u^a_j \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.179}$$

$$r^a_j = v_a - u^a_j \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.180}$$

$$r^a_j \leq M s^a_j \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.181}$$

$$\rho^a_j \leq 1 - s^a_j \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.182}$$

$$\rho^a_j \geq 0 \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.183}$$

$$s^a_j \in \{0,1\} \qquad \forall j\in A_a, \forall a\in\Psi \tag{6.184}$$

$$\delta_i\rho^a_j \leq z^a_{ij} \leq 1 \qquad \forall i\in D, \forall j\in A_a, \forall a\in\Psi \tag{6.185}$$

$$\delta_i \geq 0 \qquad \forall i\in D \tag{6.186}$$

6.5 Experimental results

In this section we provide some analysis on the performance of the multi-follower algorithms proposed. All the tests have been executed on a 64 bit, 2.33 GHz Intel(R) Xeon(R) CPU with 16GB of RAM. We used CPLEX 11.0.1 to solve LPs and MIPs, SNOPT 7.2.4 to solve NLPs and SCIP 3.0.0 to solve MINLPs. Multiple-LPs and Multiple-MINLPs algorithms rely on a branch and bound procedure whose overhead is not considered in this work as irrelevant if compared to the total time. For the evaluation of the problems we generated a set of normal form games grouped by dimension. The dimension of a game is denoted by the number of players' actions. For simplicity we suppose all players having the same number of pure strategies. In most of the experiments, we generated games starting from 5 actions per player up to 50 actions per player, except for problems exceeding time limit before and for each dimension we generated 50 games using GAMUT [32]. In order to keep the analysis as general as possible we generated payoff matrices as uniform random matrices assuming integer values from 0 to 100. When a different payoff structure is used we will say it explicitly. Note that in polymatrix cases, although the payoffs are still limited from 0 to 100, the

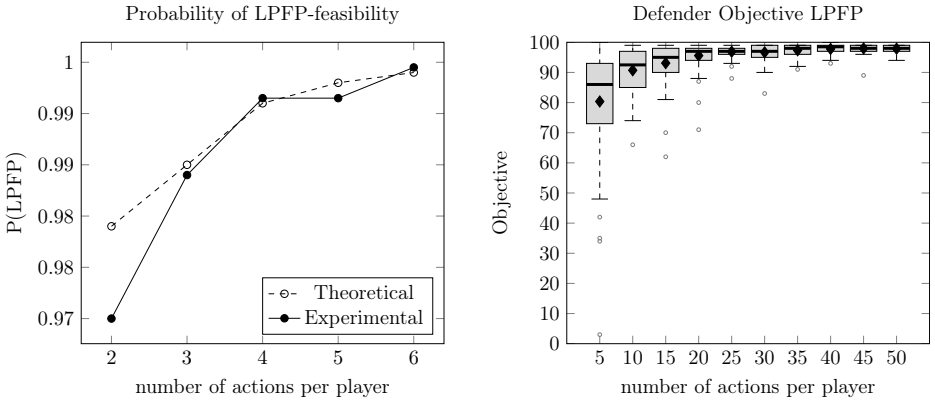


Figure 6.6: The left plot shows the relative frequency of LPFP-feasible instances compared to the theoretical probability. The right plot shows how the LPFP solution value grows with the number of players' actions, in RandomGame instances.

leader's objective will range from 0 to 200 as the sum of the outcomes of two games. On the x-axis we show the number of actions of the game set, while the y-axis shows the CPU time on logarithmic scale. A time limit of 3600 seconds has been set for each problem.

In order to entirely depict the whole information about the tests we chose to represent data series using box plots. Each box gives information about the sampling distribution of the measured quantity of each set of game instances. The median is represented with a thick black line while the black diamond represents the mean value. The gray boxes contains all the samples between first quartile and third quartile, or, in other words, one half of the population is contained in the box. Samples with a value too different from median[2] are considered outliers and they are represented with a small circle outside whiskers range.

In Figure 6.6, on the left, we show the relative frequency of feasible instances from our experimental results when solving for LPFP, compared to theoretical probability reported in Table 6.1. In our experiment we set the same number of strategies to all players, generating 2000 instances per dimension with uniform random integer payoffs ranging from 0 to 100. On the right figure we can see how the value of the LPFP solution grows with respect to the number of strategies while the right image shows the the value of the LPFP solution.

Although the average solution value tends to the maximum when solving bigger instances of uniform random games, this behavior is not valid in

[2]naming q_1 the 25^{th} percentile and q_3 the 75^{th} percentile, in these plots we consider outliers all samples whose value is larger than $q_3 + \frac{3}{2}(q_3 - q_1)$ or smaller than $q_1 - \frac{3}{2}(q_3 - q_1)$

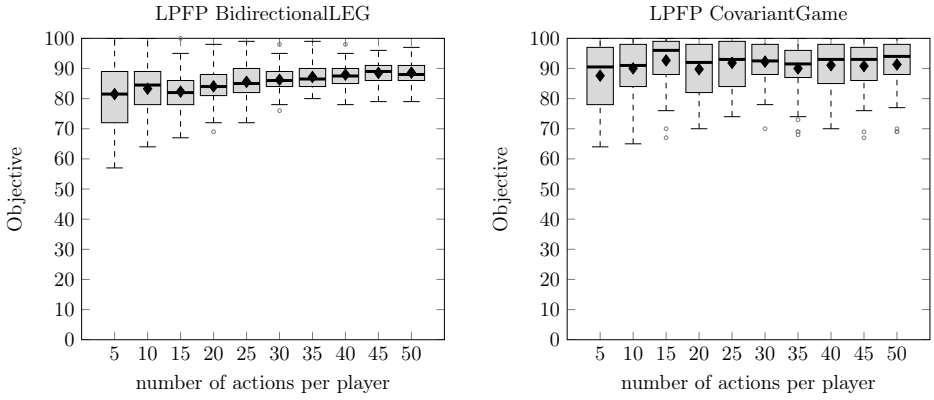


Figure 6.7: The left plot shows the defender objective in BidirectionalLEG instances while the right figure show the same result over CovariantGame instances.

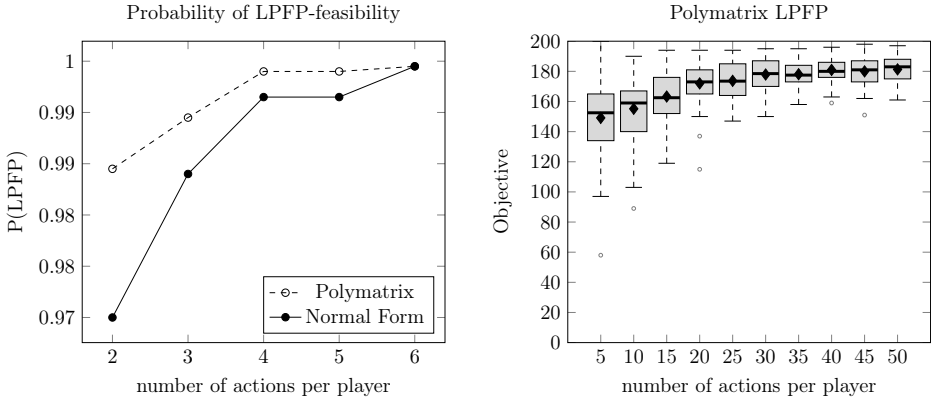


Figure 6.8: The left plot shows the relative frequency of feasible instances in polymatrix uniform random games. The right plot shows the distribution of the objective values.

general. In figure 6.7 we show the value of the LPFP equilibrium in games with different payoff structures. Both these experiments and that in Figure 6.6 on the right were conducted over 50 instances per game dimension. In Figure 6.8, on the left we see the relative frequency of feasible instances over uniform random instances in the polymatrix case while on the right we can see the leaders' utility. Again, for the feasibility analysis we generated 2000 instances per dimension in order to better extimate the relative frequency of infeasible instances, while in the objective analysis we generated 50 instances per dimension.

Figure 6.9 shows the performance of the MULTI-MINLP and MULTI-LCP algorithms which cover all the three hypotheses about followers' behaviour in LPFM games, discussed in Section 6.3.2. We generated 50

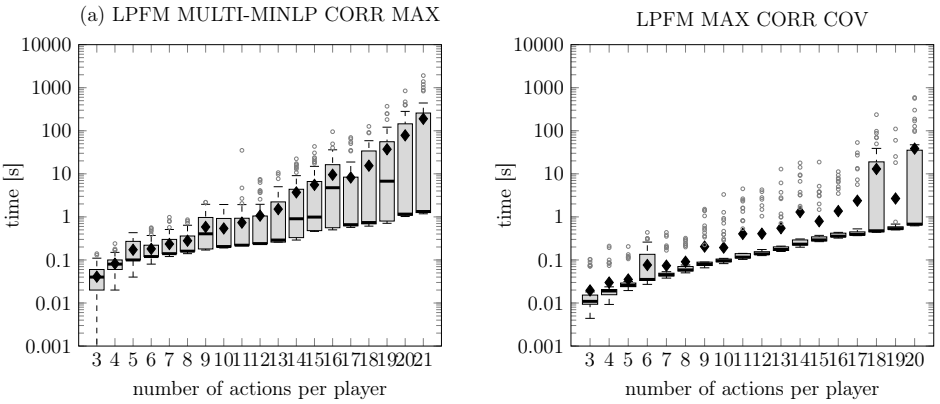


Figure 6.9: Figure (a) shows the solution times of LPFM MULTI-MINLP algorithm applied on RandomGame instances. Figure (b) shows the performance of the same algorithm over CovariantGame instances.

instances for each game dimension until 20 actions per player or until the execution time exceeded the threshold of 3600 seconds. In Figure 6.9.a we can see the solving times of MULTI-MINLP under the maximization hypothesis on RandomGame instances. In this experiment the upper bound was calculated using correlated equilibrium, which is solved in polynomial time using CPLEX. The core MINLP was solved using SCIP. We note that, although in the worst cases the branch and bound algorithm requires an exponential time to find the solution, about half of the instances were solved efficiently.

The same experiment was repeated on CovariantGame instances whose result is reported in Figure 6.9.b. As we expected, the advantage of branch and bound is more evident in games where payoffs follow some kind of schema, as in the CovariantGame case, where most of the instances were solved efficiently, while the benefit is less evident but still significant in RandomGame case. In Figure 6.10, we see how the use of correlated equilibrium as upper bound is a bad choice when solving MULTI-LCP as the overhead introduced to compute it is much higher than the time needed to execute rrLH. In the minimization case of MULTI-MINLP the branch and bound used is less effective and too many subgames had to be solved to determine the optimal solution as shown in Figure 6.11.a. The upper bound used in this experiment was the minimum among the lowest pure followers' Nash equilibrium and the rrLHsolution. The LPFM problem with maximization of leader's utility has also been solved using a single MINLP formulation which has been computed with SCIP, whose bad result shown in Figure 6.11.b.

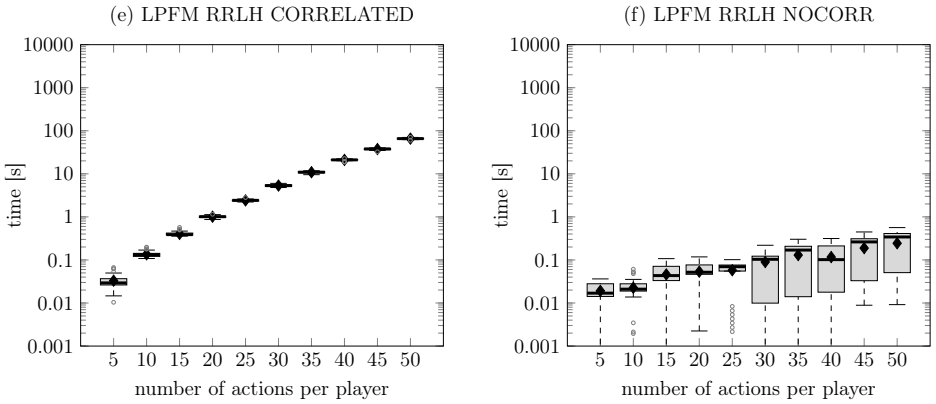


Figure 6.10: On the left figure we show the performance of MULTI-LCP with correlated equilibrium as upper bound. On the right figure the upper bound used is the best leader's payoff in the followers subgame. Both the experiments were conducted over RandomGame instances.

The solutions proposed for the LPFM polymatrix case are a single MIP and a MULTI-MIP or MULTI-LCP. The performance of the single MIP is presented in Figure 6.11.c. Although slightly more performant than the normal form formulation, it is outperformed by the MULTI-MIP polymatrix formulation, Figure 6.12.a, thus providing the same solution, given the maximization of leader's utility. Figure 6.11.c The minimization case was also able to solve all games up to 50 actions per player within the time limit. The MULTI-LCP performance are still good as we expected, see Figure 6.12.c.

The performances of LMFP algorithms for the normal form case and for the polymatrix case are shown in Figure 6.13. For both cases we compared the performance of the MULTI-LP algorithm against the MIP algorithm. The LP at the core of the branch and bound are solved using CPLEX as well as the MIP problem does. We note that the MULTI-LP formulation is quite faster than the MIP thanks to the branch and bound which allows to prune infeasible strategies before the corresponding LP has been evaluated. Moreover the variance of solution times is very low for larger game instances, which could be an interesting property for some applications. In the bottom of the figure we can see the same experiments run on polymatrix games, showing similar results.

Figure **??** shows how SNOPT solving NLP outperforms SCIP solving MINLP, however SNOPT solutions, while being feasible, are not guaranteed to be optimal. The MINLP formulation without z variables and redundant onstraints could not even solve the first 3x3x3 instance, reaching a gap between primal bound and dual bound of 4.74%. Polymatrix formulation

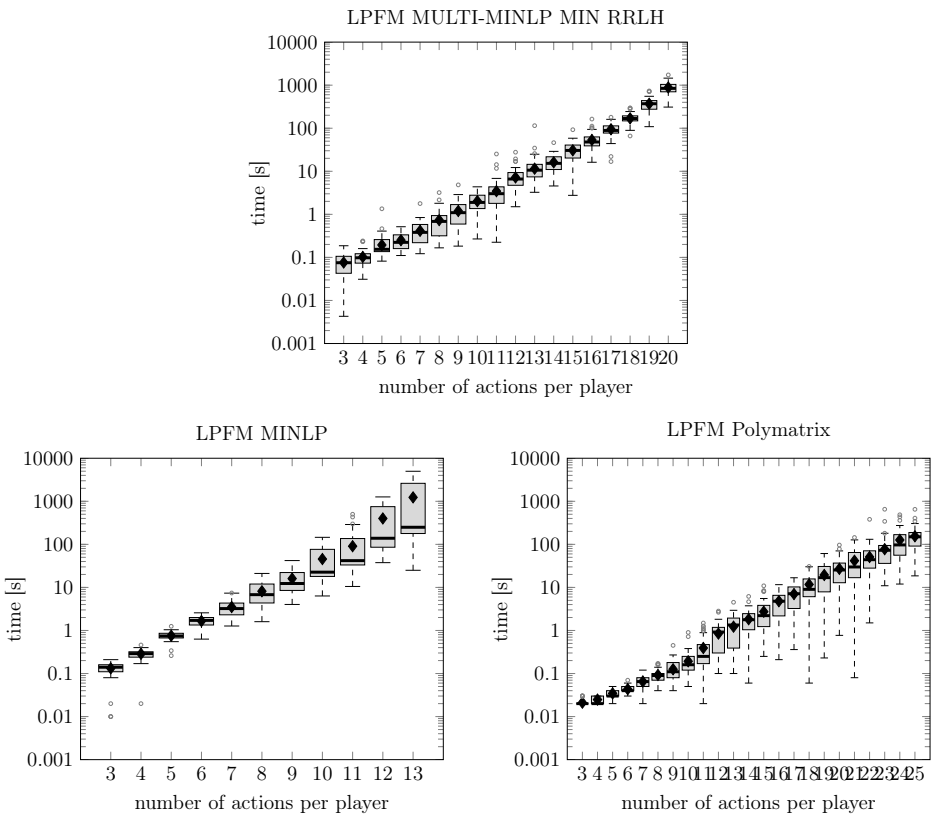


Figure 6.11: (a) Performance of LPFM MULTI-MINLP with minimization of leader's utility. (b) Performance of single MINLP formulation with maximization of leader's utility. (c) Performance of single MINLP in polymatrix games. All these experiments were conducted over RandomGame instances.

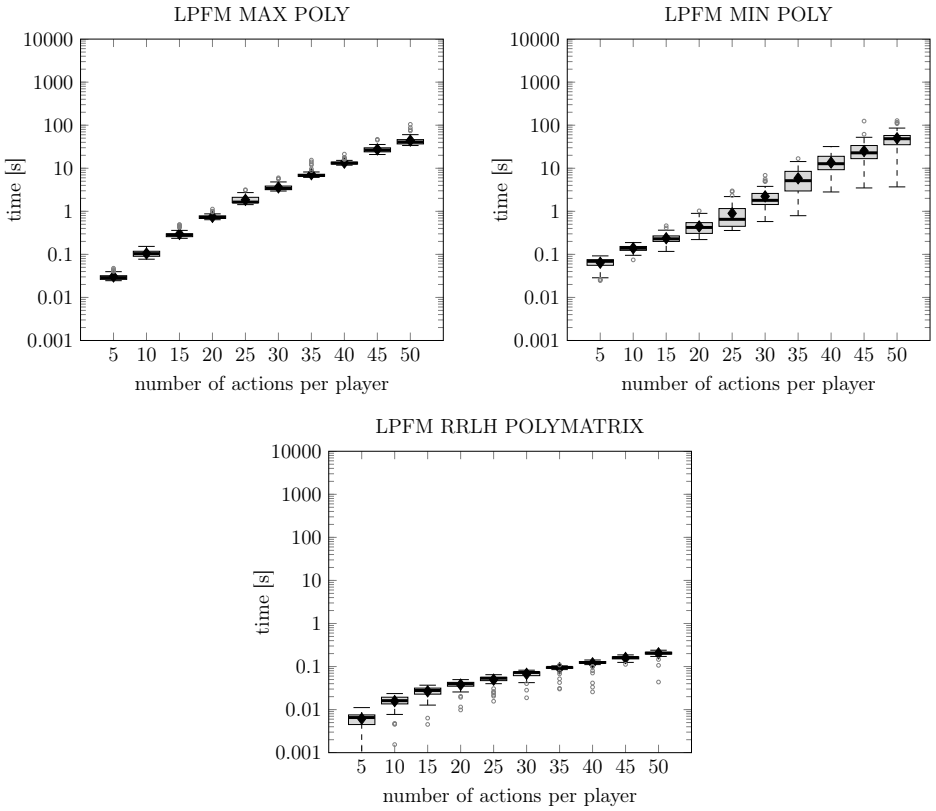


Figure 6.12: (a) Performance of LPFM MULTI-MINLP with minimization of leader's utility. (b) Performance of single MINLP formulation with maximization of leader's utility. (c) Performance of single MINLP in polymatrix games. All these experiments were conducted over RandomGame instances.

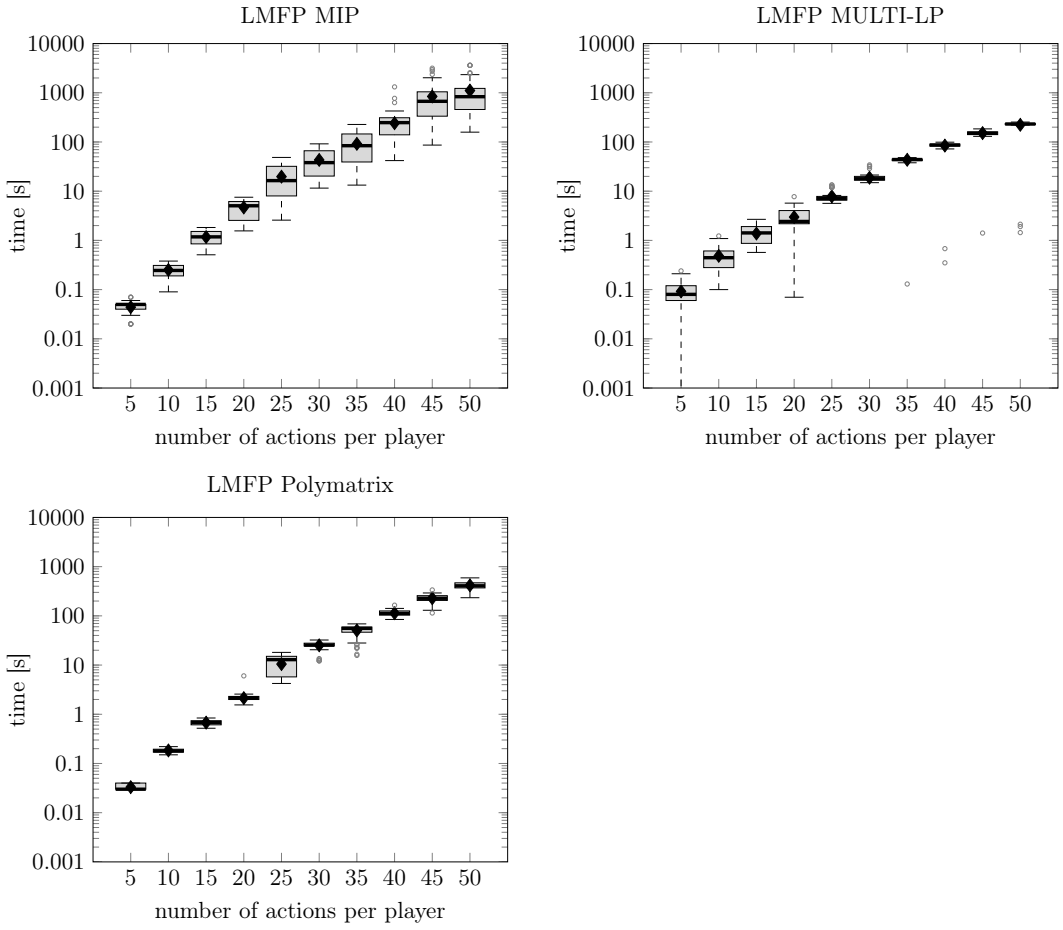
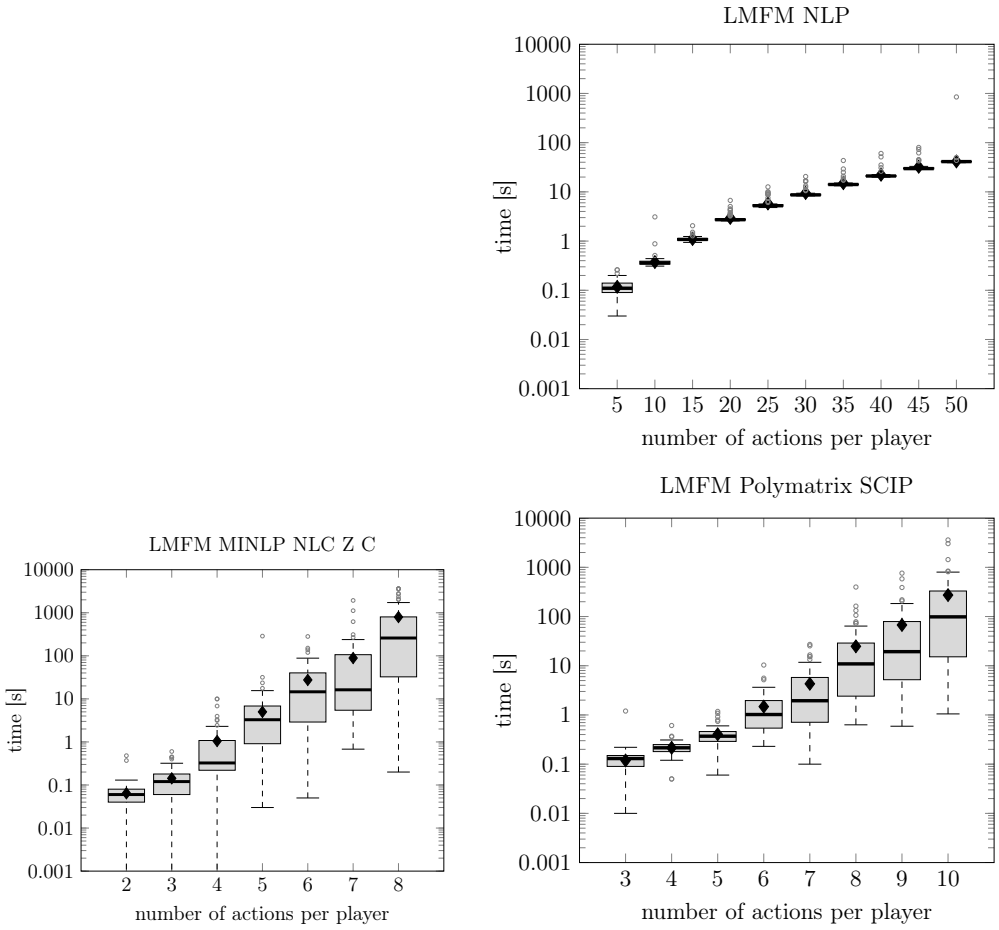
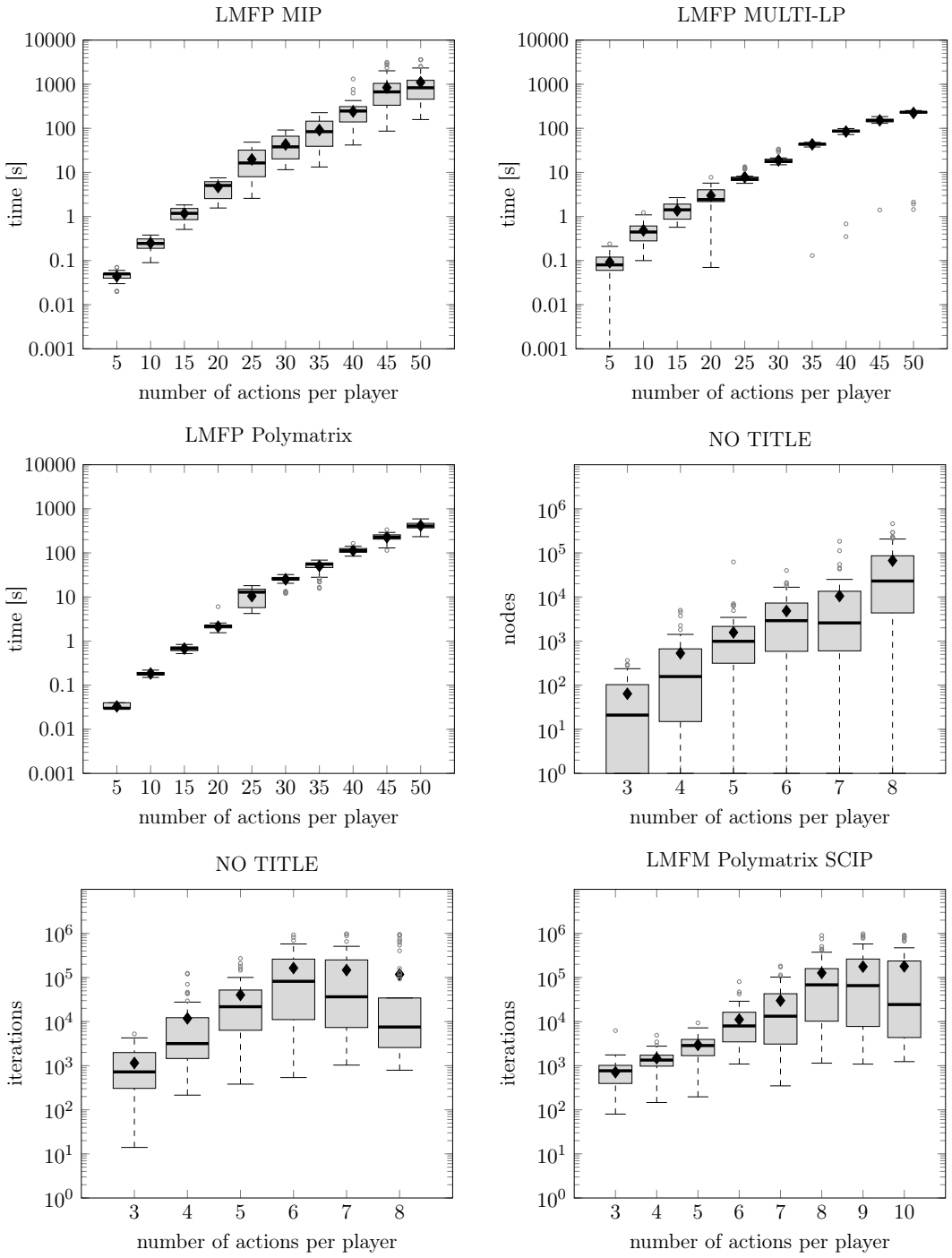
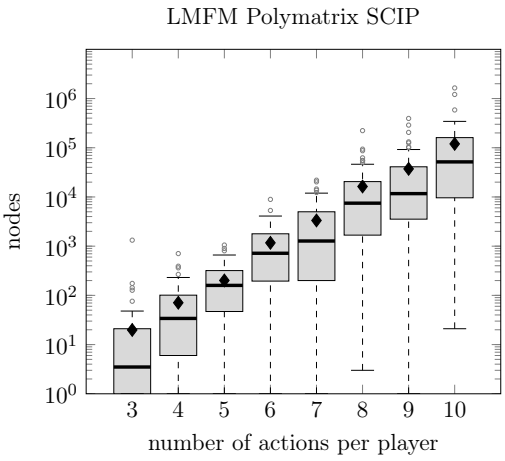


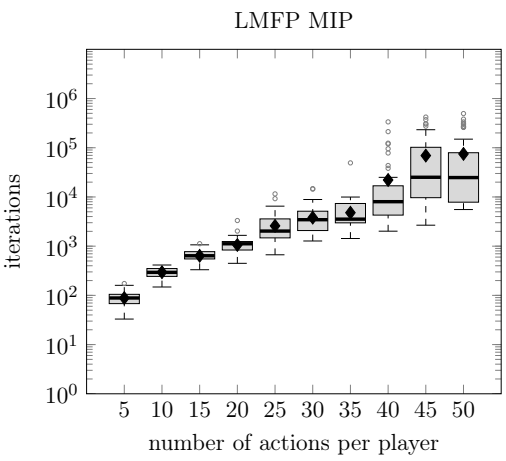
Figure 6.13: Solution times of LMFP algorithms.

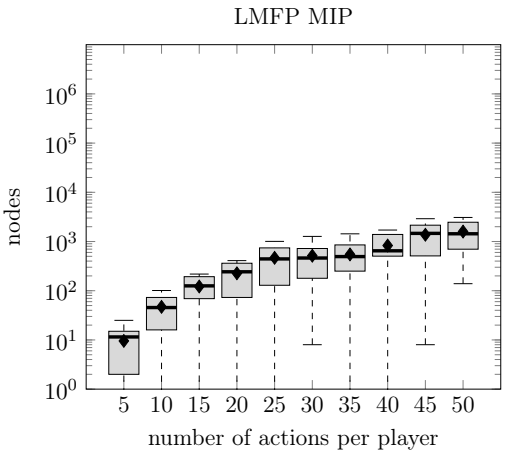
allows to solve slightly larger games but cannot solve all instances with 10 actions per player without exceeding time limit. LPFM problems are easier to solve than mixed-mixed case but neither MINLP formulation nor MULTI-MINLP formulation could solve all instance efficiently. Looking to Figure **??** we note that MULTI-MINLP formulation is some faster than the MINLP formulation, while polymatrix formulation is slightly better than MULTI-MINLP. However all formulations fail to solve larger instances. Moreover we note from center figure that about half of the generated instance could be solved quite efficiently using MULTI-MINLP, while in the worst case the computational time grows exponentially. Finally, we analyzed the LMFP case, for which we provided three polynomial-time formulations. MULTI-LP formulation and polymatrix formulation allow CPLEX to solve all games within time limit while MIP formulation




LMFP MIP
time [s]
number of actions per player
LMFP MULTI-LP
time [s]
number of actions per player
LMFP Polymatrix
time [s]
number of actions per player
NO TITLE
nodes
number of actions per player
NO TITLE
iterations
number of actions per player
LMFM Polymatrix SCIP
iterations
number of actions per player


LMFM Polymatrix SCIP
nodes
10^0
10^1
10^2
10^3
10^4
10^5
10^6
3 4 5 6 7 8 9 10
number of actions per player


LMFP MIP
iterations
10^0
10^1
10^2
10^3
10^4
10^5
10^6
5 10 15 20 25 30 35 40 45 50
number of actions per player


LMFP MIP
nodes
10^0
10^1
10^2
10^3
10^4
10^5
10^6
5 10 15 20 25 30 35 40 45 50
number of actions per player

Bibliography

[1] Robert J. Aumann. Subjectivity and correlation in randomized strategies. April 1973.

[2] Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. Protect – a deployed game theoretic system for strategic security allocation for the united states coast guard. 2012.

[3] Nicola Basilico, Nicola Gatti, and Federico Villa. Asynchronous multi-robot patrolling against intrusion in arbitrary topologies. 2010.

[4] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, 2004.

[5] M. Breton, A. Alg, and A. Haurie. Sequential stackelberg equilibria in two-person games. *Optimization Theory and Applications*, pages 71–97, 1988.

[6] Gerald Brown and Matthew Carlyle. A two-sided optimization for theater ballistic missile defense. *Annals of Mathematics*, pages 745–663, September-October 2005.

[7] Gerald Brown, Matthew Carlyle, Javier Salmeron, and Kevin Wood. Defending critical infrastructure. *Interfaces*, pages 530–544, November-December 2006.

[8] Vincent Conitzer and Dmytro Korzhyk. Commitment to correlated strategies. 2011.

[9] Vincent Conitzer and Tuomas Sandholm. Computing optimal strategies to commit to. *EC*, June 2006.

[10] Vincent Conitzer and Tuomas Sandholm. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. *EC*, June 2006.

[11] Constantinos Daskalakis, Alex Fabrikant, and Christos H. Papadimitriou. The game world is flat: The complexity of nash equilibria in succint games. *Automata, Languages and Programming*, pages 513–524, 2006.

[12] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing nash equilibrium. 2008.

[13] N. Gatti, G. Patrini, M. Rocco, and T. Sandholm. Combining local search techniques and path following for bimatrix games. *CoRR abs/1210.4858*, 2012.

[14] Philip E. Gill, Walter Murray, and Michael A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Journal on Optimization, Volume 12*, pages 979,1006, November 2002.

[15] K. Goldberg, A. J. Goldman, and M. Newman. The probability of an equilibrium point. *JOURNAL OF RESEARCH of the National Bureau of Standars - B. Mathematical Sciences Vol. 72B, No. 2*, 1968.

[16] John C. Harsanyi. Games with incomplete information played by "bayesian" players i-iii. *Management Science*, pages 159–182, November 1967.

[17] Rufus Isaacs. *Differential Games.* Dover Publications, 1965.

[18] Manish Jain, Christopher Kiekintveld, and Milind Tambe. Quality-bounded solutions for finite bayesian stackelberg games: Scaling up. *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, pages 997–1004, May 2011.

[19] Manish Jain, Dmytro Korzhyk, Ondrej Vanek, Vincent Conitzer, Michal Pechoucek, and Milind Tambe. A double oracle algorihtm for zero-sum security games on graphs. *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, pages 327–334, May 2011.

[20] Manish Jain and Fernando Ordonez. Security games with arbitrary schedules: A branch and prize approach. 2010.

[21] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordonez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pages 689–696, May 2009.

[22] Don N. Kleinmuntz and Henry Willis. Risk-based allocation of resources to counter terrorism. *RAND*, 2009.

[23] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. *www.aaai.org*, 2010.

[24] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Solving stackelberg games with uncertain observability. *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, pages XXX–XXX, May 2011.

[25] C.E. Lemke and J. J. T. Howson. Equilibrium points of bimatrix games. *SIAM J APPL MATH*, pages 413–423, 1964.

[26] Joshua Letchford and Vincent Conitzer. An efficient heuristic for security against multiple adversaries in stackelberg games. *www.aaai.org*, 2007.

[27] Joshua Letchford and Vincent Conitzer. Computing optimal strategies to commit to in extensive-form games. June 2010.

[28] Maschler M. A price leadership method for solving the inspector's non-constant-sum game. *Naval Research Logistic Quarterly*, pages 11–33, 2009.

[29] Richard D. McKelvey and Thomas R. Palfrey. Quantal response equilibria for normal form games. March 1994.

[30] John Nash. Non-cooperative games. *Annals of Mathematics*, pages 286–295, September 1951.

[31] N. Nisan, T. Roughgarden, E. Tardos, and V.V. Vazirani. Algorithmic game theory. *Cambridge University Press*, 2007.

[32] Eugene Nudelman, Jennifer Wortman, Yoav Shoham, and Kevin Leyton-Brown. Run the gamut: A comprehensive approach to evaluating game-theoretic algorithms. *AAMAS*, 2004.

[33] T. D. Parsons. *Pursuit-evasion in a graph.* Springer, 1976.

[34] Praveen Paruchuri, Jonathan P. Pearce, and Sarit Kraus. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems, 2008*, pages XXX–XXX, May 2008.

[35] James Pita, Manish Jain, Janusz Marecki, Fernando Ordonez, 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. *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 125–132, May 2008.

[36] James Pita, Milind Tambe, Chris Kiekintveld, Shane Cullen, and Erin Steigerwald. Guards - game theoretic security allocation on a national scale. *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems - Innovative Applications Tracks (AAMAS 2011)*, pages XXX–XXX, May 2011.

[37] R. Porter, E. Nudelman, and Yoav Shoham. Simple search methods for finding a nash equilibrium. *www.aaai.org*, 2004.

[38] Tuomas Sandholm, Andrew Gilpin, and Vincent Conitzer. Mixed-integer protramming methods for finding nash equilibria. 2005.

[39] Yoav Shoam and Kevin Leyton-Brown. *Multiagent Systems*. 2009.

[40] Jason Tsai, Shyamsunder Rathi, Christopher Kiekintveld, Fernando Ordonez, and Milind Tambe. Iris - a tool for strategic security allocation in transportation networks. *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pages 37–44, May 2009.

[41] J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, page 100:295–320, 1927.

[42] H. von Stackelberg. *Marktform und Gleichgewitch*. Springer, 1934.

[43] Bernhard von Stengel and Shmuel Zamir. Leadership with commitment to mixed strategies. pages XXX–XXX, February 2004.

[44] Bernhard von Stengel and Shmuel Zamir. Leadership games with convex strategy sets. *Games and Economic Behavior*, August 2010.